DEPARTMENT OF TRANSPORTATION

Development of a Travel-Time Reliability Measurement System

Eil Kwon, Principal Investigator
Department of Civil Engineering
University of Minnesota Duluth

September 2018

Research Report
Final Report 2018-28

To request this document in an alternative format, such as braille or large print, call 651-366-4718 or 1-800-657-3774 (Greater Minnesota) or email your request to ADArequest.dot@state.mn.us. Please request at least one week in advance.

Technical Report Documentation Page

1. Report No.	2.	3. Recipients Accession No.
MN/RC 2018-28		

4. Title and Subtitle	5. Report Date
Development of a Travel-Time Reliability Measurement System	September 2018
	6.
7. Author(s) Eil Kwon, Chongmyung Park	8. Performing Organization Report No.
9. Performing Organization Name and Address University of Minnesota Duluth 252 SCiv, 1405 University Dr. Duluth, MN 55812	10. Project/Task/Work Unit No. CTS#2016005 11. Contract (C) or Grant (G) No. (C) 99008 (WO) 188
12. Sponsoring Organization Name and Address Minnesota Department of Transportation Research Services & Library 395 John Ireland Boulevard, MS 330 St. Paul, Minnesota 55155-1899	13. Type of Report and Period Covered Final Report 14. Sponsoring Agency Code

15. Supplementary Notes
http://mndot.gov/research/reports/2018/201828.pdf

16. Abstract (Limit: 250 words)

This study has developed a computerized Travel-Time Reliability Measurement System (TTRMS), which can automate the time-consuming process of gathering and managing data from multiple sources and calculating various types of reliability measures under user-specified conditions for given corridors. The TTRMS adopts a server and client structure, where the main database and computational engines reside in the server, while the user-clients are designed for entering the data and generating the output files. In particular, most of the external data, such as traffic and weather datasets, can be remotely downloaded following predefined time schedules. Further, the travel-time calculation process developed in this study can explicitly reflect various lane-configurations at work zones for correctly calculating travel times of the routes with work zones. The map-based user interfaces provide users with a flexible environment, where the route selection and specification of operating conditions for reliability estimation can be efficiently performed. The integrated TTRMS was tested in the Twin Cities' metro freeway network by estimating the reliability measures of selected corridors with real data for a two-year period, 2012-13. The test results indicate that the TTRMS can substantially reduce the time and effort in estimating various types of reliability measures under different operating conditions for predefined corridors.

17. Document Analysis/Descriptors	18. Availability Statement
Transportation operations, Weather conditions, Snow, Performance measurement, Traffic flow, Travel time, Traffic volume, Stopped time delays	No restrictions. Document available from: National Technical Information Services, Alexandria, Virginia 22312

19. Security Class (this report)	20. Security Class (this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	232	

DEVELOPMENT OF A TRAVEL-TIME RELIABILITY MEASUREMENT SYSTEM

FINAL REPORT

Prepared by:

Eil Kwon
Chongmyung Park
Department of Civil Engineering
University of Minnesota Duluth

SEPTEMBER 2018

Published by:

Minnesota Department of Transportation
Research Services & Library
395 John Ireland Boulevard, MS 336
St. Paul, Minnesota 55155-1899

ACKNOWLEDGEMENTS

This research was financially supported by the Minnesota Department of Transportation. The authors greatly appreciate the technical guidance and data support from the staff at the Regional Traffic Management Center, in particular, Brian Kary and Garrett Shreiner. Also, the administrative support from Nelson Cruz and Thomas Johnson-Kaiser is very much appreciated.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1

1.1 Background and Research Objectives 1

1.2 Report Organization 1

CHAPTER 2: DESIGN OF THE TRAVEL-TIME RELIABILITY MEASUREMENT SYSTEM 3

2.1 Introduction 3

2.2 Overview of the TTRMS Architecture 3

2.3 Design of TTRMS Server 6

2.4 Design of User Client 23

2.5 Design of the Admin Client 27

CHAPTER 3: ENHANCEMENT OF THE TRAVEL TIME ESTIMATION MODULE FOR WORK ZONE SITES 32

3.1 Introduction 32

3.2 Design of the Route-Configuration Module for Work-Zones 33

3.3 Enhancement of the Travel-Time Calculation Module for Work-Zones 37

3.3.1 Basic Methodology for Travel-time Estimation 37

3.3.2 Design of the Modules for Calculating Work-Zone Route Travel-Time 37

3.4 Development and Testing of the Route-Configuration and Travel-Time Calculation Modules 43

CHAPTER 4: DEVELOPMENT OF A DATA-CONVERSION AND ROUTE-CONFIGURATION MODULE 52

4.1 Introduction 52

4.2 Development of an Internal Database for Managing and Storing Reliability-Related Data 53

4.3 Development of Conversion Modules for External Non-Traffic Data 66

4.3.1 Design of Data-Conversion Process 66

4.3.2 Development of Data-Access Layer 70

4.3.3 Development of API Server 73

4.3.4 Development of External-Data Reader 76

4.4 Development of Administration-Client Module 80

4.5 Development of Travel-Time Reliability-Route Configuration Modules 90

CHAPTER 5: DEVELOPMENT OF THE TRAVEL TIME RELIABILITY COMPUTATION MODULE 96

5.1 Introduction 96

5.2 Development of Weather and Incident Data Reader Modules 98

5.2.1 Development of Additional Data Reader module for NOAA Weather Data 98

5.2.2 Development of Incident Data Reader module 100

5.3 Development of Travel-Time Processing Module 102

5.4 Development of Data Categorization Module 103

5.4.1 Design of Data Categorization module 103

5.4.2 Development of Sub Modules for Categorizing Each Data Type 106

5.5 Development of Reliability Calculation Module 117

5.5.1 Structure of Reliability Calculation module 117

5.5.2 Development of Data Extraction module 118

5.5.3 Development of Reliability Calculation module 121

5.5.4 Example of Reliability Calculation 123

CHAPTER 6: DEVELOPMENT OF A TRAVEL-TIME INFORMATION MODULE 127

6.1 Introduction 127

6.2 Development of Travel-Time Information and Public Service API Modules 129

6.3 Development of an Example Webpage for MnDOT Driver-Information System 137

CHAPTER 7: DEVELOPMENT OF THE USER-INTERFACE AND REPORT-GENERATION MODULE 141

7.1 Introduction 141

7.2 Development of the User-Interface and User-Service Modules 144

7.2.1 Overview of the User-Interface and User-Service modules 144
7.2.2 Development of the User-Interface module 145
7.2.3 Development of the User-Service module 153
7.3 Development of the Reliability-Estimation and Report-Generation Module 155
7.3.1 Overview of the Reliability Estimation and Report module 155
7.3.2 Development of the Reliability-Estimation Process module 155
7.3.3 Development of the Report-Generation module 158
CHAPTER 8: SYSTEM INTEGRATION AND TESTING 167
8.1 Introduction 167
8.2 Integration of the Entire System 170
8.2.1 Enhancement of the Admin-Client and Admin-Service module 170
8.2.2 Development of the Periodic Data-Processing module 174
8.2.3 Development of the Task-Processing module 177
8.3 Operating Process of the Integrated System 181
8.4 Testing of the Integrated System 185
CHAPTER 9: CONCLUSIONS 211
REFERENCES 213

LIST OF FIGURES

Figure 2.2.1 User Groups of TTRMS 4

Figure 2.2.2 Main Modules of TTRMS 5

Figure 2.3.1 Components of TTRMS Server 6

Figure 2.3.2 Database Model Diagram 13

Figure 2.4.1 Components of User Client 23

Figure 2.5.1 Components of Admin Client 28

Figure 3.1.1 Examples of Work-Zone Lane-configurations 32

Figure 3.2.1 Framework for Travel-Time Reliability Measurement System 33

Figure 3.2.2 ‘RouteInfo’ class for storing lane-configuration data 35

Figure 3.2.3 Class-diagram for Route-Configuration 36

Figure 3.2.4 Process Diagram for Configuration of Travel-Routes 36

Figure 3.3.1 Vehicle-trajectories leaving the first station in the beginning of each time interval 38

Figure 3.3.2 Pseudo-code of the travel-time function 40

Figure 3.3.3 Class Diagram of travel-time calculation process 40

Figure 3.3.4 Sequence-diagram of travel-time calculation process 41

Figure 3.3.5 Sequence-diagram for developing station list 41

Figure 3.3.6 Pseudo code of *get_stations()* function in *RouteInfoHelper* 42

Figure 3.3.7 Sequence-diagram for checking detectors 42

Figure 3.3.8 Pseudo-code of the *check_detector()* function in *RouteInfoHelper* 43

Figure 3.4.1 Main script for operating TTRMS server 44

Figure 3.4.2 Graphical User-Interface for Work-Zone Route/Lane-Configuration 44

Figure 3.4.3 Lane configuration of I-694 EB case 45

Figure 3.4.4 Screenshot of the I-694 EB route created in the client program 46

Figure 3.4.5 Lane configuration of I-35E NB case (June 18th, 2013) 47

Figure 3.4.6 I-35E NB case (June 18th, 2013) created in client program 48

Figure 3.4.7 Lane configuration of I-35E NB work-zone (9th July 2013) 49

Figure 3.4.8 I-35E NB work-zone (9th July 2013) created in client program 50

Figure 4.2.1 The New Modules developed in Chapter4 in TTRMS Architecture 54

Figure 4.2.2 Relationship Diagram of Database 55

Figure 4.2.3 Types of External Data in Client and Server 63

Figure 4.2.4 Sample Database-Table Class for Special Event Data 64

Figure 4.2.5 Sample Python Data-Class for Special Event Data 64

Figure 4.2.6 Sample Java Data Class for Special Event 65

Figure 4.3.1 Overview of Data-Conversion Process for External Data 66

Figure 4.3.2 Data-Conversion Modules for Type 1 External Data 67

Figure 4.3.3 An Example JSON String 68

Figure 4.3.4 An Example Sequence Diagram to Add a New Data 69

Figure 4.3.5 Data-Conversion Process and Modules for Type 2 Data 70

Figure 4.3.6 Structure of Data-Access Layer 70

Figure 4.3.7 Data-Access Module for Special Event Data 72

Figure 4.3.8 Data-Access-Base module 73

Figure 4.3.9 DB Access-Module Test Results with an Incident Data 73

Figure 4.3.10 Structure of API Server and Data Service Modules 74

Figure 4.3.11 Special-Event Data-Service module 74

Figure 4.3.12 TeTRES-API Module 75

Figure 4.3.13 Structure of RWIS Data Reader 76

Figure 4.3.14 RWIS Data Reader module 76

Figure 4.3.15 Detector station S73 and nearby RWIS site locations 77

Figure 4.3.16 Output Screen of Test Program 78

Figure 4.3.17 Precipitation History Page in SCANWeb for the Same Duration with Test Code 79

Figure 4.3.18 Surface history page for the same duration with test code in SCANWeb site 79

Figure 4.4.1 Structure of Administration-Client Module for Each Data Type 80

Figure 4.4.2 DataAPI module for Special-Event data 81

Figure 4.4.3 APIClient module 82

Figure 4.4.4 Work-Zone List Display Panel 84

Figure 4.4.5 Dialog to Add a New Work-Zone 84

Figure 4.4.6 Lane Configuration Edit Dialog 85

Figure 4.4.7 Spreadsheet for Lane Configuration 85

Figure 4.4.8 Main Window of Special-Event User Interface 86

Figure 4.4.9 Dialog to add a New Special Event 87

Figure 4.4.10 Snow Management Information panel 88

Figure 4.4.11 Dialog for New Snow-Event and Management Data 88

Figure 4.4.12 Snow-Management Section List panel 89

Figure 4.4.13 Dialog for New Snow-Management Section 89

Figure 4.5.1 Travel-Time Route Configuration Process 90

Figure 4.5.2 Reliability-Route Configuration Main User Interface 91

Figure 4.5.3 Dialog for New Reliability-Route 91

Figure 4.5.4 Confirmation Dialog to Create Opposite Direction Route 92

Figure 4.5.5 Class definition for Reliability-Route in Client 93

Figure 4.5.6 Data API module for Reliability-Route in Client 93

Figure 4.5.7 Reliability-Route Service module in Server 94

Figure 4.5.8 Data-Access module for Reliability-Route Data in Server..95
Figure 4.5.9 Sample Reliability-Route Data stored in Database 95
Figure 4.5.10 "route" Field Data of Sample Reliability-Route Database 95
Figure 5.1.1 The Modules developed in Chapter 5 in TTRMS Architecture...97
Figure 5.2.1 Data Flow and Structure of Weather Data Reader Module 99
Figure 5.2.2 Weather Stations in the Twin Cities Metro Area 100
Figure 5.2.3 Incident Data Reader module 101
Figure 5.3.1 Structure of Travel-Time Processing module...102
Figure 5.4.1 Data flows of Data Categorization module 103
Figure 5.4.2 Relationship diagram of the database 105
Figure 5.4.3 Example junction table between travel-time and weather tables 106
Figure 5.4.4 Flow charts of Weather Categorization module 107
Figure 5.4.5 Distance and offset-distance in incident data categorization 108
Figure 5.4.6 Flow charts of Incident Categorization module 109
Figure 5.4.7 Location type, distance and offset-distance in work zone data categorization 111
Figure 5.4.8 Flow chart of Work Zone Categorization module 112
Figure 5.4.9 Arrival and Departure Time Window in special event data categorization..........................113
Figure 5.4.10 Flow charts of Special Event Categorization module...114
Figure 5.4.11 Flow chart of Snow-Management Categorization module...116
Figure 5.5.1 Structure and Data flow of the Reliability Calculation module 117
Figure 5.5.2 Flow chart of Reliability Calculation process 118
Figure 5.5.3 Class diagram for data filter..119
Figure 5.5.4 Class diagram of filter generation functions for each data type 120
Figure 5.5.5 Flow chart of reliability calculation process..122

Figure 5.5.6 Example travel time route on I-35E (NB) 123

Figure 5.5.7 Example program to perform travel time reliability calculation 125

Figure 5.5.8 Output from example application 126

Figure 6.1.1 TTI and PS-APT Modules in the TTRMS Architecture 128

Figure 6.2.1 The Relationship between Travel-Time Information and other relevant modules 130

Figure 6.2.2 Process to Calculate Reliability Measures using Historical Data 132

Figure 6.2.3 Source code for Reliability Measure Calculation using Historical Data 134

Figure 6.2.4 A Sequence diagram of travel-time information service process 135

Figure 6.2.5 Source code of the travel-time information service function in the TTI Module 136

Figure 6.3.1 An Example web page for travel-time information service 138

Figure 6.3.2 Example application results of the Travel-Time Information Module for a route on I-35E (NB) 139

Figure 6.3.3 Example Application Results of the Travel-Time Information Module for a route on I-35W (NB) 140

Figure 7.1.1 Architecture of TTRMS and the User-Interface Modules 143

Figure 7.2.1 Overview of User-Client and User-Service module structure 144

Figure 7.2.2 Estimation Panel of the User Interface 146

Figure 7.2.3 Route-Selection Panel of the User Client 147

Figure 7.2.4 Operating Condition-Configuration Panel of the User Interface 149

Figure 7.2.5 Combinations of the Operating Sub-Conditions 150

Figure 7.2.6 Dialog to Add a Weather sub-Condition 151

Figure 7.2.7 API Client module structure 151

Figure 7.2.8 HttpClient class structure 152

Figure 7.2.9 Data Types used in API Client module 152

Figure 7.2.10 User-Service module structure 153

Figure 7.2.11 Source code of User-Service module 154

Figure 7.3.1 Estimation and Report module structure 155
Figure 7.3.2 Reliability-Estimation module structure 156
Figure 7.3.3 Sequence Diagram of Reliability-Estimation Process 157
Figure 7.3.4 Sequence Diagram of Reliability Estimation Process 158
Figure 7.3.5 Output File Example of Whole-Time-Reliability Writer module 159
Figure 7.3.6 Output File Example of Whole Time Period Reliability Writer by Operating Conditions 160
Figure 7.3.7 Output File Example of Time of Day Reliability Writer module 161
Figure 7.3.8 Output File Example of Travel Time Data Writer module 161
Figure 7.3.9 An Example Cumulative-Probability graph of Travel Time Rate 162
Figure 7.3.10 An Example Buffer-Index graph depending on Operating Conditions 163
Figure 7.3.11 An Example Graph for Yearly Buffer Index Variations 163
Figure 7.3.12 An Example Graph for Yearly Multiple Indices Comparison 163
Figure 7.3.13 An Example Graph for Monthly Buffer Index Variations 164
Figure 7.3.14 An Example Graph for Monthly Multiple Indices Comparison 164
Figure 7.3.15 An Example Graph for Daily Buffer Index Variations 164
Figure 7.3.16 Variations of Daily Travel-Time Rate vs. Buffer-Index 165
Figure 7.3.17 An Example graph for Travel-Time Variations by Time of Day 165
Figure 7.3.18 An Example graph for Yearly Planning-Time Index by Time of Day 166
Figure 7.3.19 An Example graph for Monthly Planning-Time Index by Time of Day 166
Figure 8.1.1 The Modules developed in Chapter 8 in the TTRMS Architecture 169
Figure 8.2.1 Updated and Added Modules for the System Integration 170
Figure 8.2.2 User Interface of the Admin Client to Manage Non-Traffic Data 171
Figure 8.2.3 Data Change Log Management User Interface 172
Figure 8.2.4 System-Configuration User Interface 173

Figure 8.2.5 Periodic Data Processing module 174

Figure 8.2.6 Sub-modules in the Daily-Tasks module in Periodic Data Processing 175

Figure 8.2.7 Structure and Sequence for Weekly-Task processing in Periodic Data Processing module. 176

Figure 8.2.8 Structure and Operational Sequence of the Monthly-Task module in Periodic Data Processing 176

Figure 8.2.9 Structure of the Task-Processing module 177

Figure 8.2.10 Sub-modules in the Initial-Data Mark module.. 178

Figure 8.2.11 Flow Chart of Data Processor module 179

Figure 8.2.12 Flow Chart of Travel Time Data Checker module 180

Figure 8.3.1 Operating Process of the Integrated System 181

Figure 8.3.2 Screen Shot of Admin Client 182

Figure 8.3.3 Screen Shot of User Client 183

Figure 8.3.4 Example Web Page for Travel-Time Information Service 184

Figure 8.4.1 Freeway Sections used for defining Travel-Time Routes... 185

Figure 8.4.2 Buffer Index (95^{th}-ile) by Operating Condition Types for I-35E NB..................................... 190

Figure 8.4.3 Yearly Buffer Index (95^{th}-ile) Variations for Operating Condition Type “All” of I-35E NB 190

Figure 8.4.4 Monthly Buffer Index (95^{th}-ile) Variations for Operating Condition Type “All” of I-35E NB. 191

Figure 8.4.5 Cumulative Probability of Travel-Time Rate of I-35E NB 191

Figure 8.4.6 Daily Buffer Index (95^{th}-ile) vs. Travel-Time Rate for Operating Condition Type “All” of I-35E NB.. 191

Figure 8.4.7 Time-of-Day Travel-Time Distribution of I-35E NB 192

Figure 8.4.8 Monthly TOD Buffer Index (95%-ile) of I-35E NB.. 192

Figure 8.4.9 Buffer Index (95^{th}-ile) by Operating Conditions of I-35W NB 196

Figure 8.4.10 Yearly Buffer Index (95^{th}-ile) Variations for Operating Condition Type “All” of I-35W NB. 197

Figure 8.4.11 Monthly Buffer Index (95^{th}-ile) Variations for Operating Condition Type “All” of I-35W NB 197

Figure 8.4.12 Cumulative Probability of Travel Time Rate of I-35W NB 197

Figure 8.4.13 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for “All operating condition” of I-35W NB 198

Figure 8.4.14 Time-of-Day Travel Time Distribution of I-35W NB 198

Figure 8.4.15 Monthly TOD Buffer Index (95%-ile) of I-35W NB 198

Figure 8.4.16 Buffer Index (95th-ile) by Operating Conditions of U.S.169 NB 203

Figure 8.4.17 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type “All” of U.S.169 NB 203

Figure 8.4.18 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type “All” of U.S.169 NB 203

Figure 8.4.19 Cumulative Probability of Travel Time Rate of U.S.169 NB 204

Figure 8.4.20 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for “All operating condition” of U.S.169 NB 204

Figure 8.4.21 Time-of-Day Travel Time Distribution of U.S.169 NB 204

Figure 8.4.22 Monthly TOD Buffer Index (95%-ile) of U.S.169 NB 204

Figure 8.4.23 Buffer Index (95th-ile) by Operating Condition of T.H.100 NB 209

Figure 8.4.24 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type “All” of T.H.100 NB 209

Figure 8.4.25 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type “All” of T.H.100 NB 209

Figure 8.4.26 Cumulative Probability of Travel Time Rate of T.H.100 NB 210

Figure 8.4.27 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for “All operating condition” of T.H.100 NB 210

Figure 8.4.28 Time-of-Day Travel Time Distribution of T.H.100 NB 210

Figure 8.4.29 Monthly TOD Buffer Index (95%-ile) of T.H.100 NB 210

LIST OF TABLES

Table 2.3.1 Components of TTRMS Server 7

Table 2.3.2 Database Schema 14

Table 2.4.1 Components of User Client 24

Table 2.5.1 Components of Admin Client 28

Table 3.4.1 Selected Detectors and Speed Data on the Travel-route in I-694 EB work-zone 46

Table 3.4.2 Calculated travel-time of I-694 EB Work-Zone 47

Table 3.4.3 Selected Detectors and Speed Data on the travel-lanes of I-35E NB case (June 18th, 2013) ... 48

Table 3.4.4 Calculated travel time of I-35E NB case (June 18th, 2013) 49

Table 3.4.5 Selected Detectors and Speed Data on the travel-lanes of I-35E NB work-zone (July 9th, 2013) 50

Table 3.4.6 Calculated travel time of I-35E NB work-zone (July 9th, 2013) 51

Table 4.2.1 Database Tables 56

Table 4.2.2 Database Schema 57

Table 4.4.1 Symbols in Lane-Configuration Spreadsheet File 83

Table 5.2.1 Weather Data Table 98

Table 5.2.2 Incident Data Tables 101

Table 5.3.1Travel-time table schema 103

Table 5.4.1 Weather data categorization scheme 106

Table 5.4.2 Junction-table schema between travel-time and weather tables 107

Table 5.4.3 Incident data categorization scheme 108

Table 5.4.4 Junction table schema between travel time and incident tables 108

Table 5.4.5 Work zone data categorization scheme 110

Table 5.4.6 Junction table schema between travel time and work zone tables 110

Table 5.4.7 Special event data categorization scheme 113

Table 5.4.8 Junction table schema between travel time and work zone tables 113

Table 5.4.9 Snow management data categorization scheme 115

Table 5.4.10 Junction table schema between travel time and snow management tables 115

Table 6.2.1 Regimes used in the Travel-Time Information Module 129

Table 6.2.2 Database Table for Travel Time Information module 130

Table 8.4.1 Operating-Condition Types used in the Test 186

Table 8.4.2 Estimated Reliability Measures by Operating Conditions of I-35E NB Route 186

Table 8.4.3 Yearly Reliability Measures of Operating Condition Type "All" of I-35E NB 187

Table 8.4.4 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of I-35E NB 187

Table 8.4.5 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of I-35E NB 189

Table 8.4.6 Reliability Measures by Operating Condition Type of I-35W NB 192

Table 8.4.7 Yearly Reliability Measures for Operating Condition Type "All" of I-35W NB 193

Table 8.4.8 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of I-35W NB 194

Table 8.4.9 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of I-35W NB 195

Table 8.4.10 Reliability Measures by Operating Conditions of U.S.169 NB 198

Table 8.4.11 Yearly Reliability Measures for Operating Condition Type "All" of U.S.169 NB 199

Table 8.4.12 Monthly Reliability Measures in 2012 of Operating Condition Type "All" of U.S.169 NB ... 200

Table 8.4.13 Monthly Reliability Measures in 2013 of Operating Condition Type "All" of U.S.169 NB ... 201

Table 8.4.14 Reliability Measures by Operating Condition Types of T.H.100 NB 205

Table 8.4.15 Yearly Reliability Measures for Operating Condition Type "All" of T.H.100 NB 205

Table 8.4.16 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of T.H.100 NB .. 206

Table 8.4.17 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of T.H.100 NB .. 207

EXECUTIVE SUMMARY

Travel-time reliability has been emerging as one of the major measures in quantifying the operational effectiveness of transportation networks. While the importance of travel time reliability in measuring the performance of transportation systems has been well recognized by transportation professionals, the current state of the practice has not reached the point where various types of reliability measures under different operating conditions can be automatically generated using data from multiple sources. This study developed a computerized Travel-Time Reliability Measurement System (TTRMS), which can automate the time-consuming process of gathering and managing data from multiple sources and calculating various reliability measures under user-specified conditions for given corridors. The TTRMS adopts a server and client structure, where the main database and computational engines reside in the server, while the user-clients are designed for entering the data and generating the output files and reports. In particular, most of the external data, such as traffic and weather datasets, can be remotely downloaded following predefined time schedules. Further, the travel-time calculation process developed in this study can explicitly reflect the various lane-configurations at work zones for correctly calculating the travel times of the routes with work zones. The map-based user interfaces provide the users of TTRMS with a flexible environment, where the route selection and specification of operating conditions for reliability estimation can be efficiently performed. The integrated TTRMS was tested with real corridors in the metro freeway network in the Twin Cities, and the reliability measures for the selected corridors were estimated for a two-year period, 2012-13. The test results indicated that the TTRMS developed in this study can substantially reduce the time and effort in estimating the various types of the reliability measures under different operating conditions for the predefined corridors. Future research needs include the application of reliability measures in identifying and prioritizing the bottlenecks in the metro freeway network. The extension of reliability to new measures, which can quantify the vulnerability and resilience levels of the existing corridors in dealing with large-scale incidents and natural events, is also recommended. Such measures can be directly applicable for effectively allocating the operational resources to the priority routes and also for developing short- and long-term plans for freeway-network improvements.

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND AND RESEARCH OBJECTIVES

Travel time reliability is formally defined as the consistency or dependability in travel times, as measured from day-to-day and/or across different times of the day (1). While the importance of travel time reliability in measuring the performance of transportation systems has been well recognized by transportation professionals, its measurement and application is still an emerging practice. Recently a series of research projects under the SHRP2 program produced a set of the guidelines in measuring and applying travel-time reliability measures (2-4). However, the current state of the practice has not reached the point where various types of reliability measures under different operating conditions can be automatically generated using data from multiple sources. To be sure, most reliability estimations performed in the SHRP2 studies to date have employed spreadsheet-based, project-specific processes, which require extensive efforts for gathering and managing a large amount of data from various sources, such as traffic, weather, incident and work-zone databases. Such a labor-intensive process in estimating reliability measures has restricted the scope of the reliability applications.

This study develops a computerized Travel-Time Reliability Measurement System (TTRMS), which can automate the time-consuming process for gathering and managing data from multiple sources and calculating the various types of reliability measures under user-specified conditions for corridors in the metro freeway network in the Twin Cities. The specific objectives of this research include:

- Development of a data management system for incorporating different types of data from multiple sources,
- Development of a travel-time reliability computation module for the selected corridors under various operational conditions, e.g., weather, incidents and construction sites, etc.
- Development of a set of user interfaces that can facilitate the input and output processes for reliability estimation.

Further, a reliability-based, time-of-day travel-time estimation module was developed and its connectivity to the existing driver-information system of MnDOT was examined. The resulting TTRMS was tested with real data from the metro freeway network.

1.2 REPORT ORGANIZATION

Chapter 2 develops a detailed design of the TTRMS architecture, where a set of the main modules, their functionalities and interrelationships are identified. In Chapter 3, the existing travel-time estimation functions in TICAS (5), Traffic Information and Condition Analysis System, developed at the University of Minnesota Duluth, will be enhanced to be able to handle the travel-time estimation of the work zones with various types of lane configurations. Chapter 4 develops the travel-time reliability calculation module, which is the main engine of the TTRMS. A reliability-based, time-of-day travel-time estimation module is developed in Chapter 5 to examine its connectivity to MnDOT's driver information system. Chapter 6 develops the user interfaces and report-generation modules for the system administrator and the general users. All the individual modules developed in this study are integrated in Chapter 7 and the resulting system is tested by estimating the reliability measures for the selected corridors with real data. Finally Chapter 8 includes the conclusions and future research needs.

CHAPTER 2: DESIGN OF THE TRAVEL-TIME RELIABILITY MEASUREMENT SYSTEM

2.1 INTRODUCTION

In this chapter, the detailed design of the Travel-Time Reliability Estimation System (TTRMS) is developed. The main output from the TTRMS includes the estimates of travel-time reliability indices, such as travel time index, buffer index and semi-variances, for predefined corridors and time periods. The input to the system consists of a set of traffic and non-traffic data. The traffic data mainly contains the traffic flow data collected from detectors on the metro freeway network, while the non-traffic data includes the types of data indicating freeway operating conditions, such as weather, incident and work zones. Both traffic and non-traffic data are combined in the TTRMS and the reliability measures under different operating conditions are estimated for given corridors. In the current version of TTRMS, the reliability measures are estimated for a set of the fixed routes, which are pre-defined and stored by the system administrator in the server. The rest of this chapter summarizes the detailed architecture of TTRMS developed with a top-down design approach.

2.2 OVERVIEW OF THE TTRMS ARCHITECTURE

TTRMS provides users the travel time and reliability information for the freeway routes predefined by the users, who can be categorized into two group, as shown in Figure 2.2.1, i.e., the general users at the Minnesota Department of Transportation (MnDOT) and the TTRMS administrator. The system administrator manages the server and the interfaces of the TTRMS to the external systems to collect the data necessary for estimating the reliability measures. The traffic data needed to estimate travel times are collected from the traffic-data archive of IRIS, while the non-traffic data, such as weather, incident, work zone, etc., are obtained from external databases.

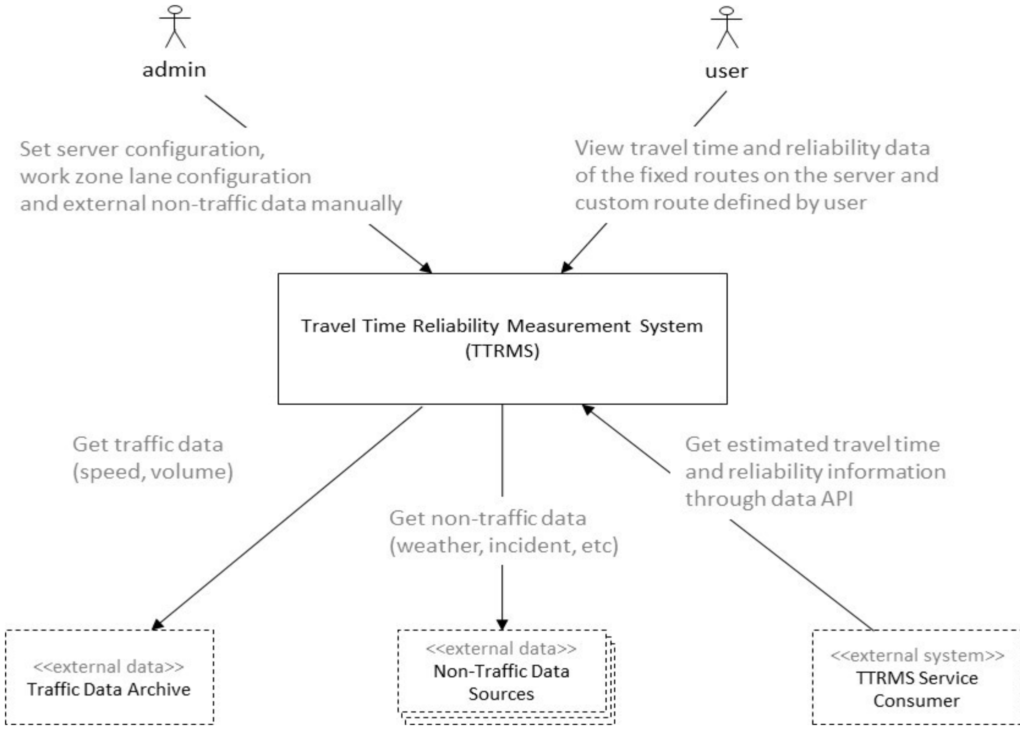


Figure 2.2.1 User Groups of TTRMS

Figure 2.2.2 shows the main modules of the TTRMS, which are consisted with three executable programs or containers, i.e., TTRMS Server, User Client and Admin Client. The travel time and reliability information, which are to be provided to the clients and the external services, are periodically stored into the database of the TTRMS Server. The server configuration and the non-traffic data sources are managed by the Admin Client. The MnDOT users can access the travel time reliability information and obtain the reliability reports by using the User Client, which can be used to configure freeway routes, calculate travel times and estimate the impacts of external factors on the travel-time reliability. The main functions of the major modules in TTRMS are as follows:

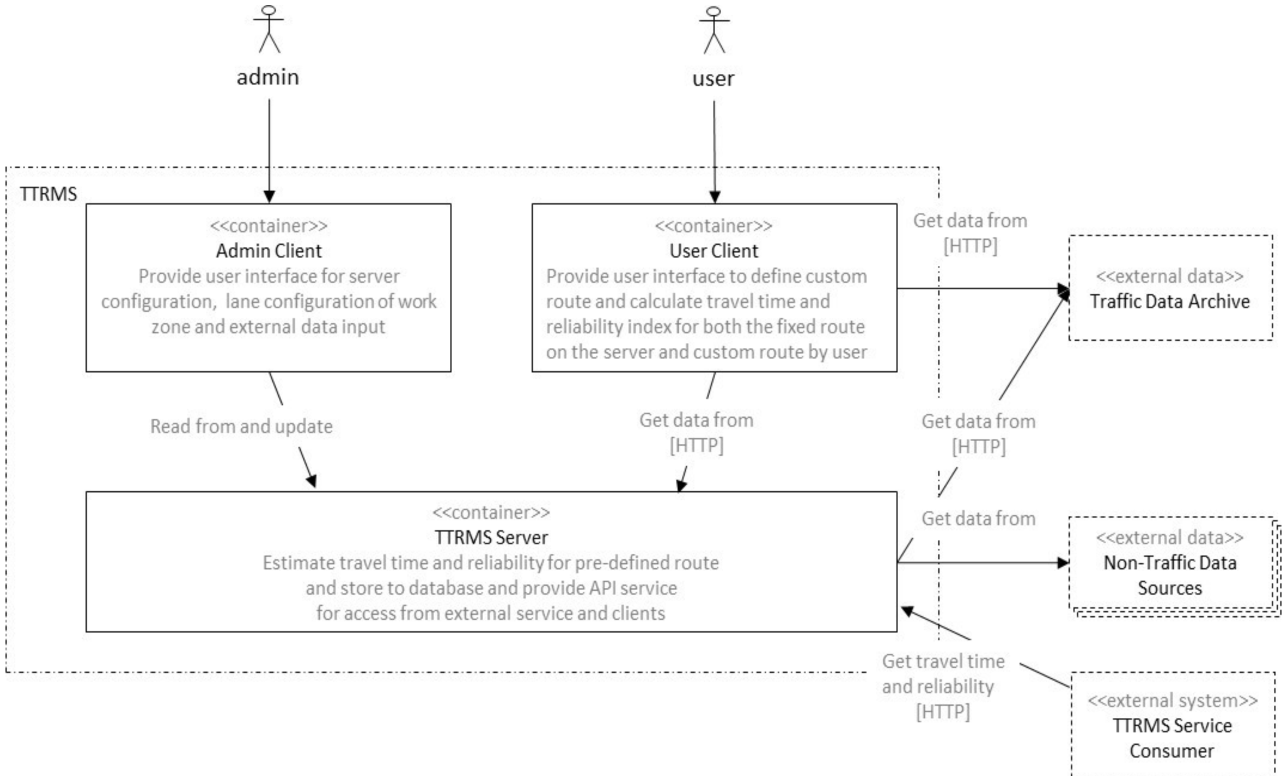


Figure 2.2.2 Main Modules of TTRMS

TTRMS Server

- Estimation of the travel times and the reliability measures for the pre-defined routes.
- Storage of the estimated information to the database.
- Provision of the API service to the external services and clients.

User Client

- Selection of the freeway routes, time periods and operating conditions for estimating reliability measures.
- Calculation of the travel times and reliability indices for the selected routes on the server.
- Estimation of the impacts of the non-traffic data on the travel time reliability for selected routes.

Admin Client

- Server configuration, lane configuration of work zones and the management of the external data.

2.3 DESIGN OF TTRMS SERVER

The TTRMS Server estimates the travel times and the reliability measures for predefined routes using historical data and provides the clients with the estimated information through the API services. Figure 2.3.1 shows the components of the server developed in this study. The Administrator sets the operational server configuration, such as target freeway routes, job schedule of the estimation process and non-traffic data source information. The non-traffic data are imported manually by the Administrator. All the requests from the Admin client are handled by the *Server-Configuration* component. The *Periodic-Job* component conducts the scheduled-estimation process by using the *Reliability-Engine* following the pre-defined schedule by the administrator. The User client and external service can access the stored information through the *Data-API* component via HTTP. The rest of this section describes the details of each component.

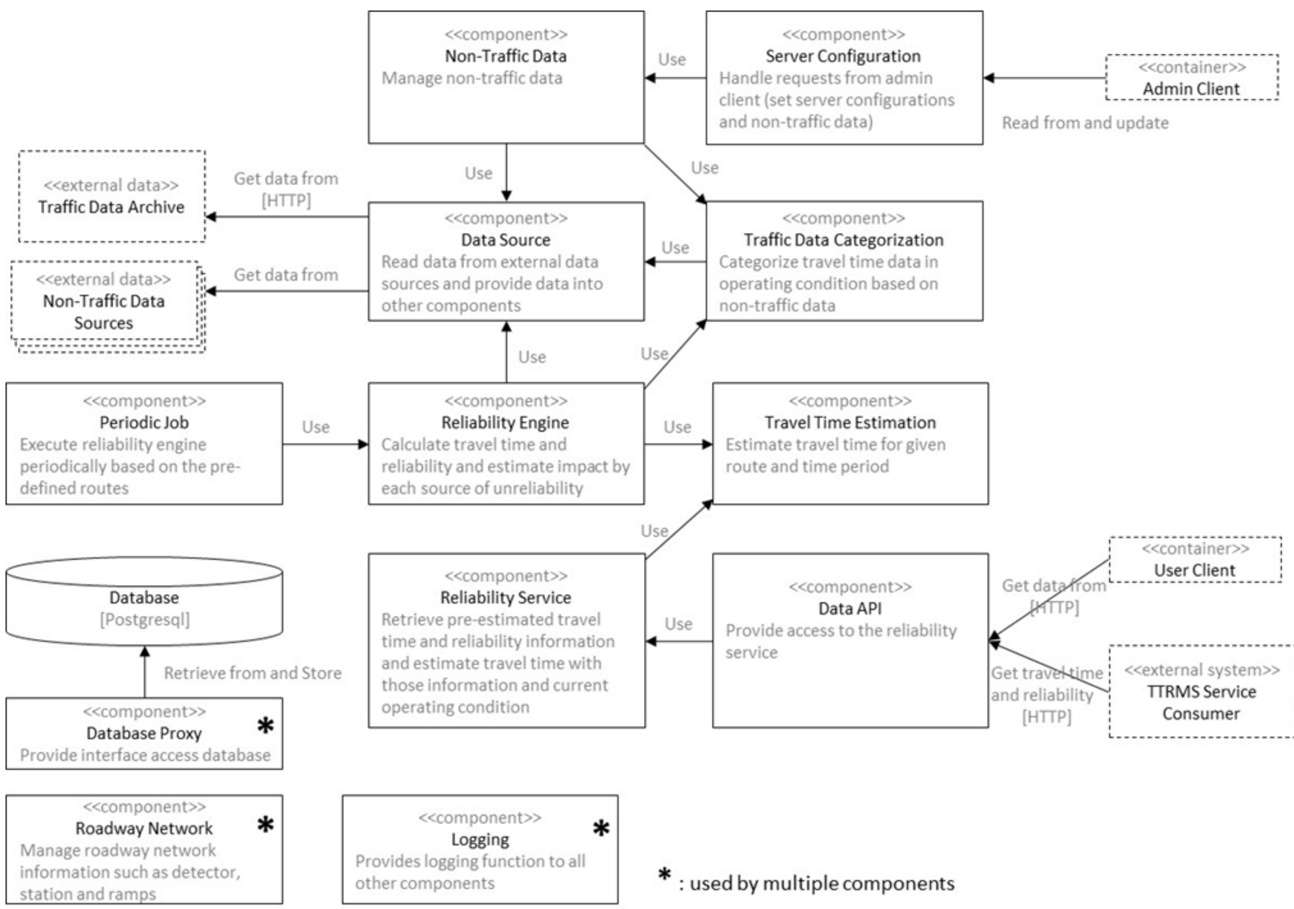


Figure 2.3.1 Components of TTRMS Server

Table 2.3.1 Components of TTRMS Server

Reliability Engine Component

Responsibilities	*Collect traffic and non-traffic data, Estimate and Categorize travel times* *Analyze relation between operating conditions and travel-time reliability*
Collaborators	*Periodic Job, Data Source, Traffic Data Categorization, Travel Time Estimation*
Input	*Freeway route, Time period*
Output	*Impacts of different regimes, which are combination of various operational conditions such as demand, weather, incident and work zones, etc., on the reliability measures*

Travel Time Estimation Component

Responsibilities	*Estimate travel time*
Collaborators	*Reliability Engine, Reliability Service*
Notes	*Lane configuration information of work zones should be considered in the estimation process*
Input	*freeway route, time period, active detector list*
Output	*travel time list for each time interval*

Traffic Data Categorization Component

Responsibilities	*categorize each travel time data based on operational conditions*
Collaborators	*Non-Traffic Data, Data Source, Reliability Engine*

Notes	*must be used after traffic and non-traffic data are saved and prepared*
Input	*freeway route, time period*
Output	*Linkage between travel time data and operation conditions in database*

Data Source Component

Responsibilities	read data from external data sources
Collaborators	Reliability Engine, Non-Traffic Data, Traffic Data Categorization
Notes	use asynchronous call with thread safe way due to delay by remote data access should consider using proxy server
Input	time period
Output	data

Periodic Job Component

Responsibilities	manage and execute periodic jobs
Collaborators	Reliability Engine
Notes	generalize job scheduling mechanism
Input	job that needs to be conducted with predetermined schedules
Output	N/A

Data API Component

Responsibilities	provide access mechanism to the reliability service component
Collaborators	Reliability Service, User Client, TTRMS Service Consumer
Notes	service method issue: 1. create a service thread for each request 2. one service thread and respond in FIFO or other scheduling
Input	request from clients
Output	corresponding output from the service component

Reliability Service Component

Responsibilities	retrieve pre-estimated travel time, reliability information and non-traffic data estimate travel time with the retrieved information and operational conditions
Collaborators	Data API, Travel Time Estimation
Input	freeway route, time period
Output	travel time, travel-time reliability indices, non-traffic data

Server Configuration Component

Responsibilities	handle requests from admin client, set server configurations
Collaborators	Admin Client, Non-Traffic Data
Input	requests (add, update, delete, etc.) configurations, non-traffic data
Output	updated configuration table of the database

Non-Traffic Data Component

Responsibilities	manage non-traffic data has data management modules for each non-traffic data type
Collaborators	Server Configuration, Non-Traffic Data
Input	non-traffic data
Output	update non-traffic data tables

Roadway Network Component

Responsibilities	load roadway network configurations from IRIS manage roadway network information such as detector, station and ramp
Collaborators	all other components in the system
Notes	network configuration change issue : freeway network configuration on IRIS changes - keep metro network information daily - roadway network information should be loaded by time period, not loaded at system booting sequence
Input	roadway node name
Output	roadway node information such as lanes, speed limit and location of station

Logging Component

Responsibilities	provide logging functions

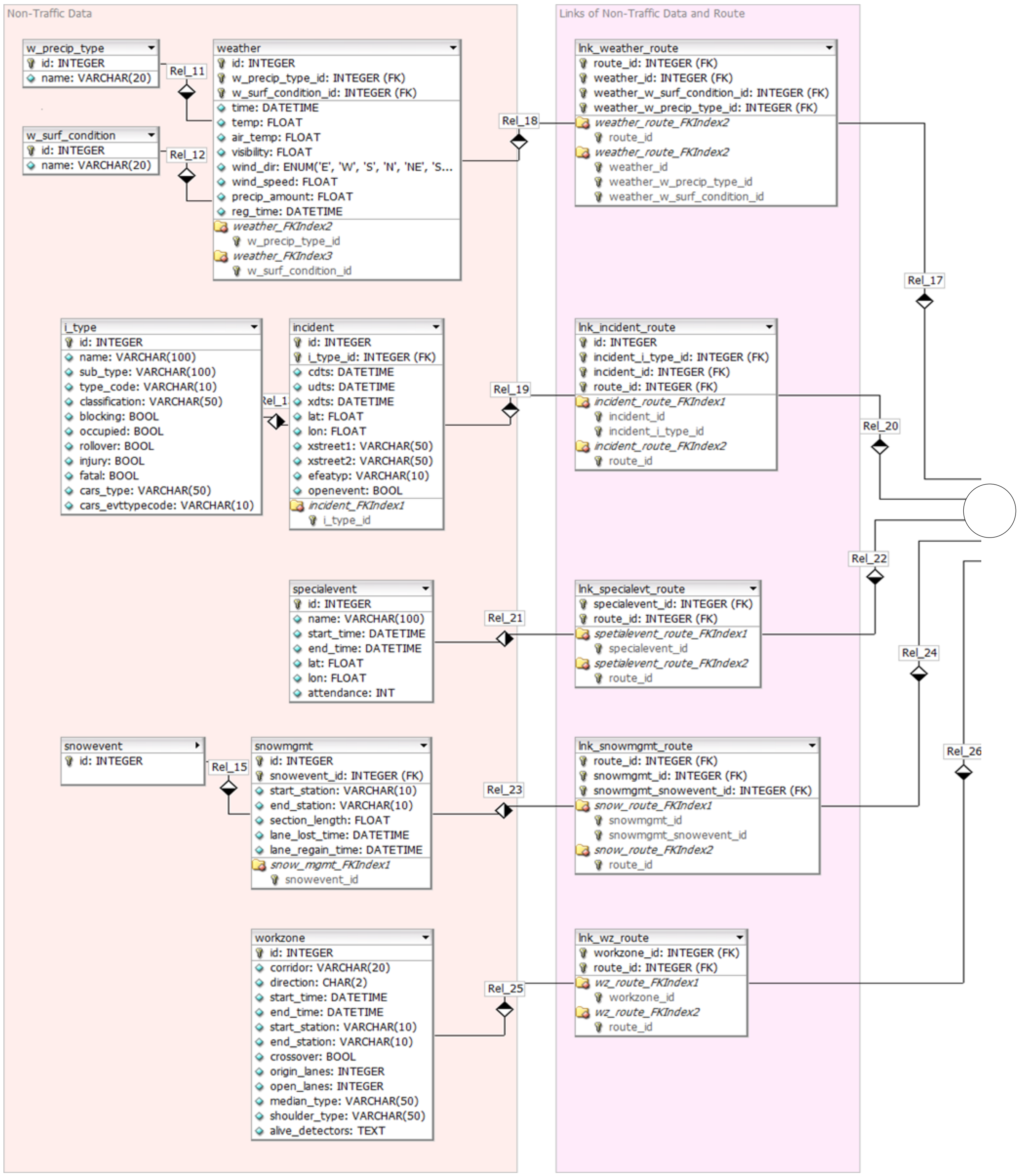
Collaborators	all other components in the system
Input	message
Output	write log message

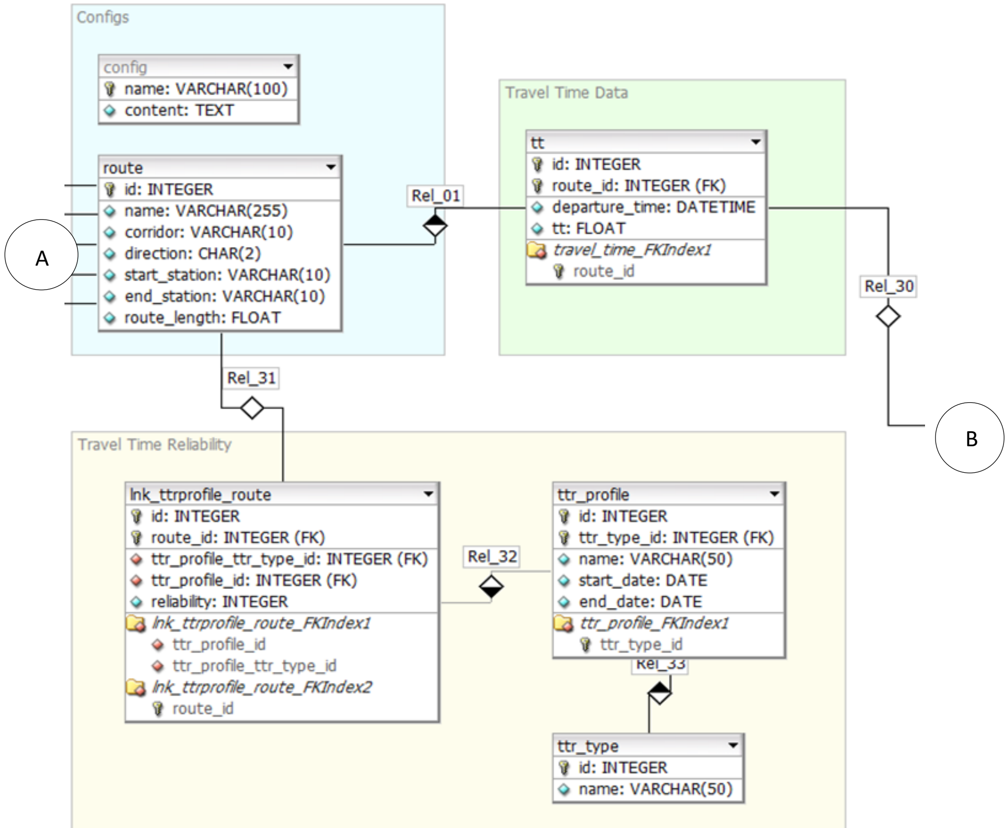
Database proxy Component

Responsibilities	provide database access interface
Collaborators	all other components in the system
Input	query
Output	update database

Design of TTRMS Database

In this research, Postgresql is used as the database engine to store the travel-time data for predefined routes, non-traffic data, estimated data and system configurations. Figure 2.3.2 and Table 2.3.2 show the database-model diagram and schema respectively.


Non-Traffic Data
Links of Non-Traffic Data and Route
w_precip_type
id: INTEGER
name: VARCHAR(20)
w_surf_condition
id: INTEGER
name: VARCHAR(20)
weather
id: INTEGER
w_precip_type_id: INTEGER (FK)
w_surf_condition_id: INTEGER (FK)
time: DATETIME
temp: FLOAT
air_temp: FLOAT
visibility: FLOAT
wind_dir: ENUM('E', 'W', 'S', 'N', 'NE', 'S...
wind_speed: FLOAT
precip_amount: FLOAT
reg_time: DATETIME
weather_FKIndex2
w_precip_type_id
weather_FKIndex3
w_surf_condition_id
lnk_weather_route
route_id: INTEGER (FK)
weather_id: INTEGER (FK)
weather_w_surf_condition_id: INTEGER (FK)
weather_w_precip_type_id: INTEGER (FK)
weather_route_FKIndex2
route_id
weather_route_FKIndex2
weather_id
weather_w_precip_type_id
weather_w_surf_condition_id
Rel_11
Rel_12
Rel_18
Rel_17
i_type
id: INTEGER
name: VARCHAR(100)
sub_type: VARCHAR(100)
type_code: VARCHAR(10)
classification: VARCHAR(50)
blocking: BOOL
occupied: BOOL
rollover: BOOL
injury: BOOL
fatal: BOOL
cars_type: VARCHAR(50)
cars_evttypecode: VARCHAR(10)
incident
id: INTEGER
i_type_id: INTEGER (FK)
cdts: DATETIME
udts: DATETIME
xdts: DATETIME
lat: FLOAT
lon: FLOAT
xstreet1: VARCHAR(50)
xstreet2: VARCHAR(50)
efeatyp: VARCHAR(10)
openevent: BOOL
incident_FKIndex1
i_type_id
lnk_incident_route
id: INTEGER
incident_i_type_id: INTEGER (FK)
incident_id: INTEGER (FK)
route_id: INTEGER (FK)
incident_route_FKIndex1
incident_id
incident_i_type_id
incident_route_FKIndex2
route_id
Rel_1
Rel_19
Rel_20
specialevent
id: INTEGER
name: VARCHAR(100)
start_time: DATETIME
end_time: DATETIME
lat: FLOAT
lon: FLOAT
attendance: INT
lnk_specialevt_route
specialevent_id: INTEGER (FK)
route_id: INTEGER (FK)
spetialevent_route_FKIndex1
specialevent_id
spetialevent_route_FKIndex2
route_id
Rel_21
Rel_22
snowevent
id: INTEGER
snowmgmt
id: INTEGER
snowevent_id: INTEGER (FK)
start_station: VARCHAR(10)
end_station: VARCHAR(10)
section_length: FLOAT
lane_lost_time: DATETIME
lane_regain_time: DATETIME
snow_mgmt_FKIndex1
snowevent_id
lnk_snowmgmt_route
route_id: INTEGER (FK)
snowmgmt_id: INTEGER (FK)
snowmgmt_snowevent_id: INTEGER (FK)
snow_route_FKIndex1
snowmgmt_id
snowmgmt_snowevent_id
snow_route_FKIndex2
route_id
Rel_15
Rel_23
Rel_24
workzone
id: INTEGER
corridor: VARCHAR(20)
direction: CHAR(2)
start_time: DATETIME
end_time: DATETIME
start_station: VARCHAR(10)
end_station: VARCHAR(10)
crossover: BOOL
origin_lanes: INTEGER
open_lanes: INTEGER
median_type: VARCHAR(50)
shoulder_type: VARCHAR(50)
alive_detectors: TEXT
lnk_wz_route
workzone_id: INTEGER (FK)
route_id: INTEGER (FK)
wz_route_FKIndex1
workzone_id
wz_route_FKIndex2
route_id
Rel_25
Rel_26



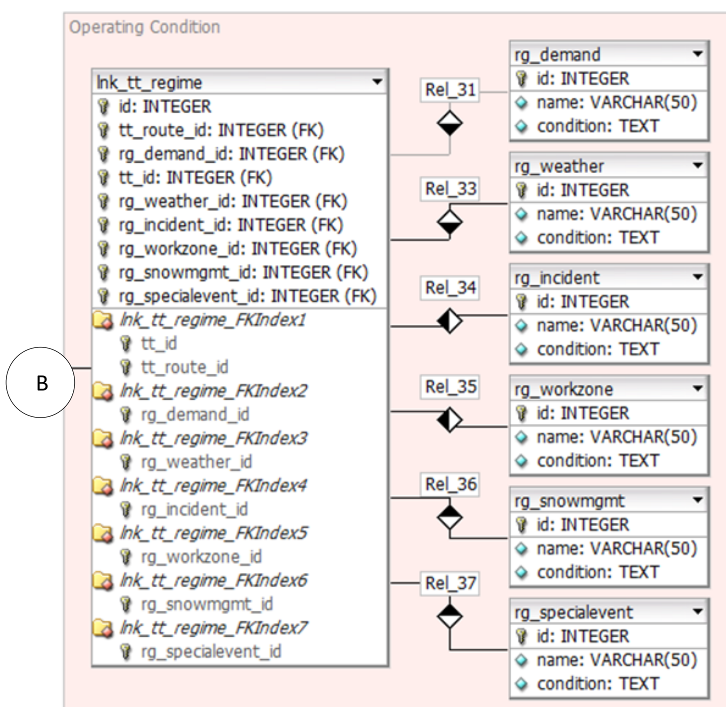


Figure 2.3.2 Database Model Diagram

Table 2.3.2 Database Schema

Configs

config

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
name	**VARCHAR(100)**	PK	NN				AI
content	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	name

route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(255)		NN				
corridor	VARCHAR(10)		NN				
direction	CHAR(2)		NN				
start_station	VARCHAR(10)		NN				
end_station	VARCHAR(10)		NN				
route_length	FLOAT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Non-Traffic Data

incident

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
i_type_id	**INTEGER**	PK	NN	UNSIGNED			
cdts	DATETIME		NN				
udts	DATETIME						
xdts	DATETIME						
lat	FLOAT		NN				
lon	FLOAT		NN				
xstreet1	VARCHAR(50)						
xstreet2	VARCHAR(50)						
efeatyp	VARCHAR(10)						
openevent	BOOL						

IndexName	IndexType	Columns

PRIMARY	PRIMARY	id i_type_id
incident_FKIndex1	Index	i_type_id

i_type

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(100)		NN				
sub_type	VARCHAR(100)						
type_code	VARCHAR(10)						
classification	VARCHAR(50)						
blocking	BOOL						
occupied	BOOL						
rollover	BOOL						
injury	BOOL						
fatal	BOOL						
cars_type	VARCHAR(50)						
cars_evttypecode	VARCHAR(10)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

snowevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
start_time	DATETIME		NN				
end_time	DATETIME		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

snowmgmt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
snowevent_id	**INTEGER**	PK	NN	UNSIGNED			
start_station	VARCHAR(10)						
end_station	VARCHAR(10)						
section_length	FLOAT						
lane_lost_time	DATETIME						
lane_regain_time	DATETIME						

IndexName	IndexType	Columns

PRIMARY	PRIMARY	id snowevent_id
snow_mgmt_FKIndex1	Index	snowevent_id

specialevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(100)		NN				
start_time	DATETIME		NN				
end_time	DATETIME		NN				
lat	FLOAT						
lon	FLOAT						
attendance	INT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

weather

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
w_precip_type_id	**INTEGER**	PK	NN	UNSIGNED			
w_surf_condition_id	**INTEGER**	PK	NN	UNSIGNED			
time	DATETIME		NN				
temp	FLOAT						
air_temp	FLOAT						
visibility	FLOAT						
wind_dir	ENUM('E', 'W', 'S', 'N', 'NE', 'SE', 'SW', 'NW')						
wind_speed	FLOAT						
precip_amount	FLOAT						
reg_time	DATETIME		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id w_precip_type_id w_surf_condition_id
weather_FKIndex2	Index	w_precip_type_id
weather_FKIndex3	Index	w_surf_condition_id

workzone

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
corridor	VARCHAR(20)		NN				
direction	CHAR(2)		NN				
start_time	DATETIME		NN				
end_time	DATETIME		NN				
start_station	VARCHAR(10)		NN				
end_station	VARCHAR(10)		NN				
crossover	BOOL		NN				
origin_lanes	INTEGER		NN	UNSIGNED			
open_lanes	INTEGER		NN	UNSIGNED			
median_type	VARCHAR(50)						
shoulder_type	VARCHAR(50)						
alive_detectors	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

w_precip_type

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(20)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

w_surf_condition

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(20)		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Links of Non-Traffic Data and Route

lnk_incident_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
incident_i_type_id	**INTEGER**	PK	NN	UNSIGNED			
incident_id	**INTEGER**	PK	NN	UNSIGNED			
route_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns

PRIMARY	PRIMARY	id incident_i_type_id incident_id route_id
incident_route_FKIndex1	Index	incident_id incident_i_type_id
incident_route_FKIndex2	Index	route_id

lnk_snowmgmt_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
route_id	**INTEGER**	PK	NN	UNSIGNED			
snowmgmt_id	**INTEGER**	PK	NN	UNSIGNED			
snowmgmt_snowevent_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	route_id snowmgmt_id snowmgmt_snowevent_id
snow_route_FKIndex1	Index	snowmgmt_id snowmgmt_snowevent_id
snow_route_FKIndex2	Index	route_id

lnk_specialevt_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
specialevent_id	**INTEGER**	PK	NN	UNSIGNED			
route_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	specialevent_id route_id
spetialevent_route_FKIndex1	Index	specialevent_id
spetialevent_route_FKIndex2	Index	route_id

lnk_weather_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
route_id	**INTEGER**	PK	NN	UNSIGNED			
weather_id	**INTEGER**	PK	NN	UNSIGNED			
weather_w_surf_condition_id	**INTEGER**	PK	NN	UNSIGNED			
weather_w_precip_type_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	route_id weather_id

		weather_w_surf_condition_id weather_w_precip_type_id
weather_route_FKIndex2	Index	route_id
weather_route_FKIndex2	Index	weather_id weather_w_precip_type_id weather_w_surf_condition_id

lnk_wz_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
workzone_id	**INTEGER**	PK	NN	UNSIGNED			
route_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	workzone_id route_id
wz_route_FKIndex1	Index	workzone_id
wz_route_FKIndex2	Index	route_id

Operating Condition

lnk_tt_regime

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
tt_route_id	**INTEGER**	PK	NN	UNSIGNED			
rg_demand_id	**INTEGER**	PK	NN	UNSIGNED			
tt_id	**INTEGER**	PK	NN	UNSIGNED			
rg_weather_id	**INTEGER**	PK	NN	UNSIGNED			
rg_incident_id	**INTEGER**	PK	NN	UNSIGNED			
rg_workzone_id	**INTEGER**	PK	NN	UNSIGNED			
rg_snowmgmt_id	**INTEGER**	PK	NN	UNSIGNED			
rg_specialevent_id	**INTEGER**	PK	NN	UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id tt_route_id rg_demand_id tt_id rg_weather_id rg_incident_id rg_workzone_id rg_snowmgmt_id rg_specialevent_id
lnk_tt_regime_FKIndex1	Index	tt_id tt_route_id
lnk_tt_regime_FKIndex2	Index	rg_demand_id

lnk_tt_regime_FKIndex3	Index	rg_weather_id
lnk_tt_regime_FKIndex4	Index	rg_incident_id
lnk_tt_regime_FKIndex5	Index	rg_workzone_id
lnk_tt_regime_FKIndex6	Index	rg_snowmgmt_id
lnk_tt_regime_FKIndex7	Index	rg_specialevent_id

rg_demand

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)		NN			Uncongested, Low, Moderate, High	
condition	TEXT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

rg_incident

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)		NN				
condition	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

rg_snowmgmt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)		NN				
condition	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

rg_specialevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)		NN				
condition	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

rg_weather

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI

name	VARCHAR(50)		NN				
condition	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

rg_workzone

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)		NN				
condition	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Travel Time Data

tt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
route_id	**INTEGER**	PK	NN	UNSIGNED			
departure_time	DATETIME		NN				
tt	FLOAT		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id route_id
travel_time_FKIndex1	Index	route_id

Travel Time Reliability

lnk_ttrprofile_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
route_id	**INTEGER**	PK	NN	UNSIGNED			
ttr_profile_ttr_type_id	INTEGER		NN	UNSIGNED			
ttr_profile_id	INTEGER		NN	UNSIGNED			
reliability	INTEGER			UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id route_id
lnk_ttrprofile_route_FKIndex1	Index	ttr_profile_id ttr_profile_ttr_type_id

lnk_ttrprofile_route_FKIndex2	Index	route_id

ttr_profile

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
ttr_type_id	**INTEGER**	PK	NN	UNSIGNED			
name	VARCHAR(50)		NN				
start_date	DATE		NN				
end_date	DATE		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id ttr_type_id
ttr_profile_FKIndex1	Index	ttr_type_id

ttr_type

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(50)						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

2.4 DESIGN OF USER CLIENT

The User Client retrieves the travel-time and reliability information from the server for predefined routes and generates a set of reports. In particular, the impacts of external operating factors, such as weather, incidents, etc., on reliability measures can be analyzed with the User Client for predefined routes. In the current version of TTRMS, the non-traffic data is stored only in the server. Therefore the client needs to use the Data API of the server to access the non-traffic data, while the traffic flow data can be directly accessible via HTTP. Figure 2.4.1 shows the components of the User Client, which reuses some modules developed for the server, such as the Reliability Engine, Traffic Data Categorization, Travel Time Estimation, Roadway Network and Logging components. In what follows, the detailed functionalities of each component in the User Client are described.

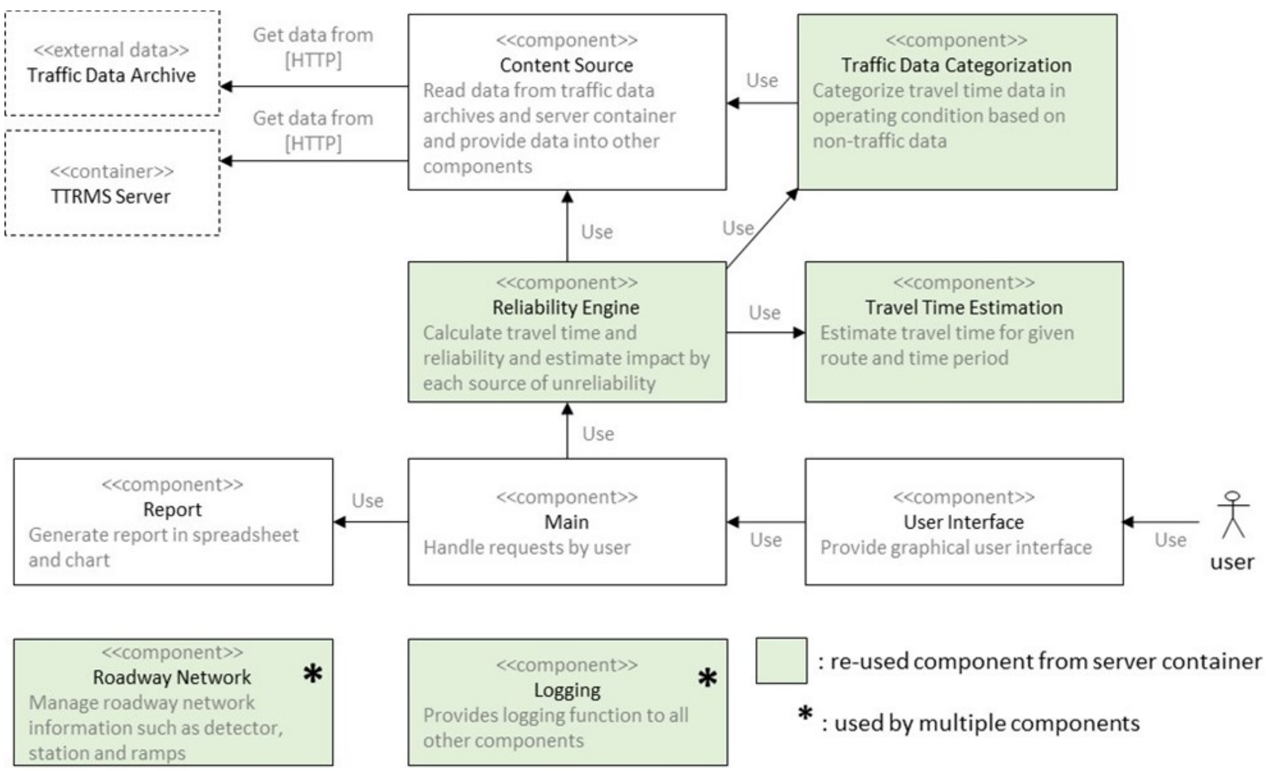


Figure 2.4.1 Components of User Client

Table 2.4.1 Components of User Client

User Interface Component

Responsibilities	provide graphical user interface
Collaborators	Main, user
Notes	support map-based freeway route definition
Input	request from user
Output	N/A

Main Component

Responsibilities	handle request from user
Collaborators	User Interface, Reliability Engine, Report
Notes	should be thread-safe
Input	request from user interface component
Output	N/A

Content Source Component

Responsibilities	read data from traffic data archives of IRIS read non-traffic data from server through data API
Collaborators	Reliability Engine, Traffic Data Categorization
Input	time period

Output	corresponding data to type

Report Component

Responsibilities	generate report in spreadsheet and chart
Collaborators	Main
Notes	use open source library to make spreadsheet and chart
Input	report type, results
Output	report file

Reliability Engine Component

Responsibilities	collect traffic and non-traffic data estimate travel time and categorize it analyze relation between unreliability sources and travel time reliability
Collaborators	Main, Content Source, Traffic Data Categorization, Travel Time Estimation
Input	freeway route, time period
Output	impact of regimes that are combination of operational conditions such as demand, weather, incident and so on

Travel Time Estimation Component

Responsibilities	estimate travel time
Collaborators	Reliability Engine

Notes	lane configuration information of work zones to be considered in estimation process get lane configuration information from the server
Input	freeway route, time period, active detector list
Output	travel time list for each time interval

Traffic Data Categorization Component

Responsibilities	categorize each travel time data based on operational conditions
Collaborators	Content Source, Reliability Engine
Notes	must be used after traffic and non-traffic data loaded
Input	freeway route, time period
Output	make link between travel time data and operation conditions in database

Roadway Network Component

Responsibilities	load roadway network configurations from IRIS manage roadway network information such as detector, station and ramp
Collaborators	all other components in the system
Notes	network configuration change issue : freeway network configuration on IRIS changes - keep metro network information daily - roadway network information should be loaded by time period,

	not loaded at system booting sequence
Input	roadway node name
Output	roadway node information such as lanes, speed limit and location of station

Logging Component

Responsibilities	provide logging functions
Collaborators	all other components in the system
Input	message
Output	write log message

2.5 DESIGN OF THE ADMIN CLIENT

The Admin Client of the TTRMS manages the location information of the external-data sources, such as server IP, port and protocol, so that the information from the external servers can be automatically collected by the server. In addition, the data-import function is provided by the Non-Traffic Data Configuration component for manually updating the data. In particular, the user-interface for the work zones on freeways is also developed to configure the lane-layout of each work-zone, so that the travel times with work zones can be estimated correctly. Figure 2.5.1 shows the components of the Admin Client, which reuses the Roadway Network and Logging components developed for the sever container. The rest of this section describes the details of each component.

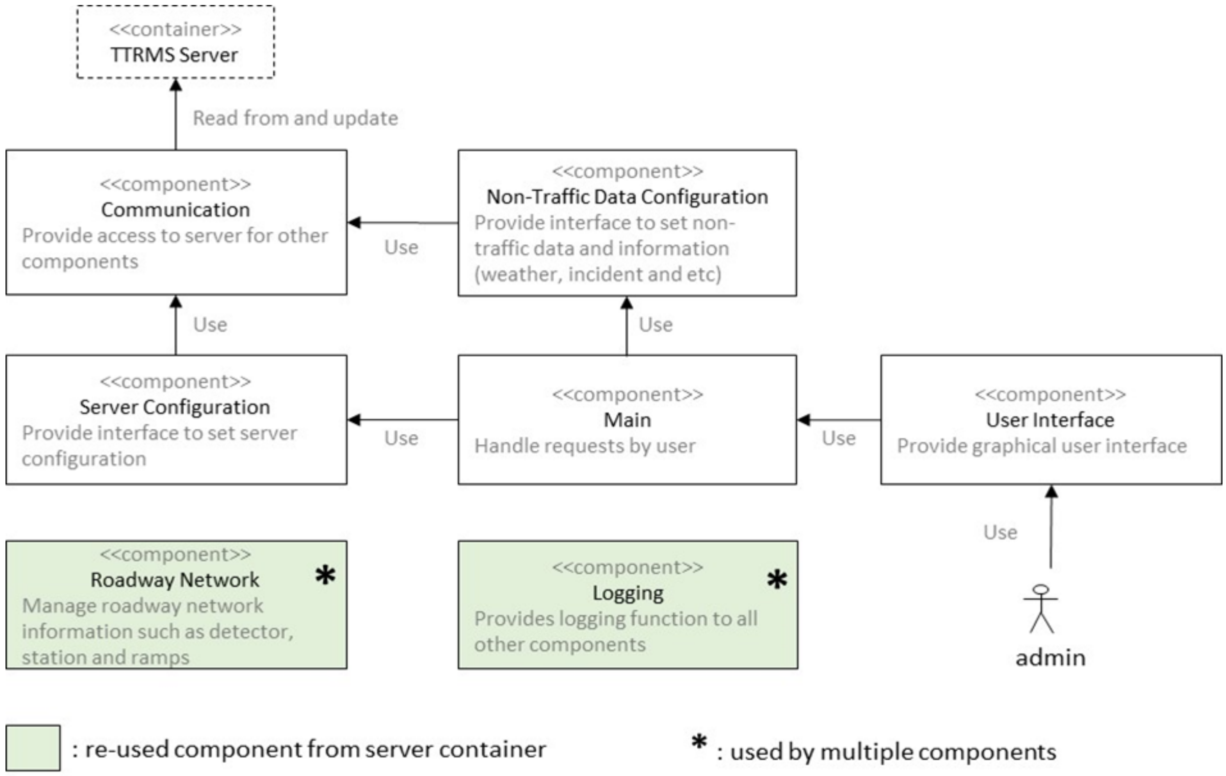


Figure 2.5.1 Components of Admin Client

Table 2.5.1 Components of Admin Client

User Interface Component

Responsibilities	provide graphical user interface
Collaborators	Main, admin
Input	request from user
Output	N/A

Main Component

Responsibilities	handle request from user

Collaborators	User Interface, Server Configuration, Non-Traffic Data Configuration
Input	request from user interface component
Output	N/A

Non-Traffic Data Configuration Component

Responsibilities	provide interface to set non-traffic data and information update database on server container
Collaborators	Main, Communication
Input	non-traffic data
Output	N/A

Server Configuration Component

Responsibilities	provide interface to set server configuration view server system log update database on server container
Collaborators	Main, Communication
Input	configurations
Output	view of data and logs

Communication Component

Responsibilities	provide access mechanism to server

Collaborators	generalize communication mechanism should consider using proxy server
Input	method (add, update and delete) data
Output	N/A

Roadway Network Component

Responsibilities	load roadway network configurations from IRIS manage roadway network information such as detector, station and ramp
Collaborators	all other components in the system
Notes	network configuration change issue : freeway network configuration on IRIS changes - keep metro network information daily - roadway network information should be loaded by time period, not loaded at system booting sequence
Input	roadway node name
Output	roadway node information such as lanes, speed limit and location of station

Logging Component

Responsibilities	provide logging functions
Collaborators	all other components in the system

Input	message
Output	write log message

CHAPTER 3: ENHANCEMENT OF THE TRAVEL TIME ESTIMATION MODULE FOR WORK ZONE SITES

3.1 INTRODUCTION

In this chapter, the existing travel-time module in TICAS, Traffic Information and Condition Analysis System developed at the University of Minnesota Duluth, is enhanced to be able to calculate the travel-times of the freeway work-zones, where various types of lane-configurations are implemented through time. Figure 3.1.1 shows the common examples of work-zone lane-configurations, which include a lane-closure, lane-shift and crossovers to opposing lanes. Further, multiple types of lane-configurations can be combined in a single work-zone, e.g., a lane-closure and a crossover, etc.

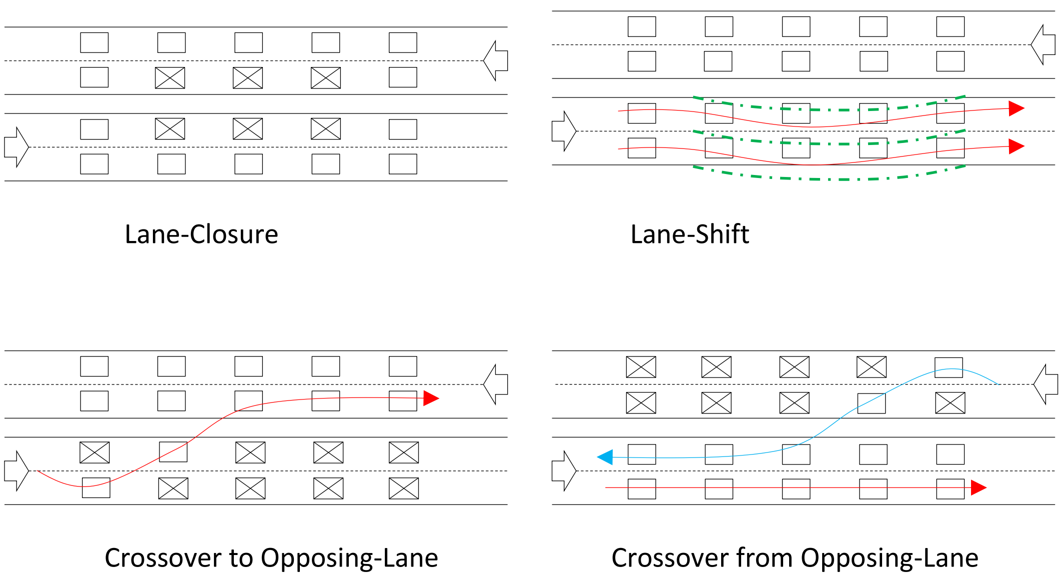


Figure 3.1.1 Examples of Work-Zone Lane-configurations

In this study, the following modules are developed to configure the travel-time routes with work-zones and to calculate the travel-times of those work-zone routes:

1) Work-zone Route-Configuration Module to construct the travel-time routes for given work-zones by identifying a list of detectors on the open-lanes for each direction,

2) Travel-time Calculation Module for new work-zone routes.

Figure 3.2.1 shows the framework of the TTRMS incorporating the above modules developed in this chapter. The geometric information for given work zones are entered through the user-interface client and the calculated travel-times are stored in the database to be used by the reliability estimation module. The rest of this chapter summarizes the details of the new work-zone route and travel-time modules along with their test results.

3.2 DESIGN OF THE ROUTE-CONFIGURATION MODULE FOR WORK-ZONES

In this research, the route-configuration module is first developed to identify a list of the detectors on the travel-route in each direction for a given work-zone.

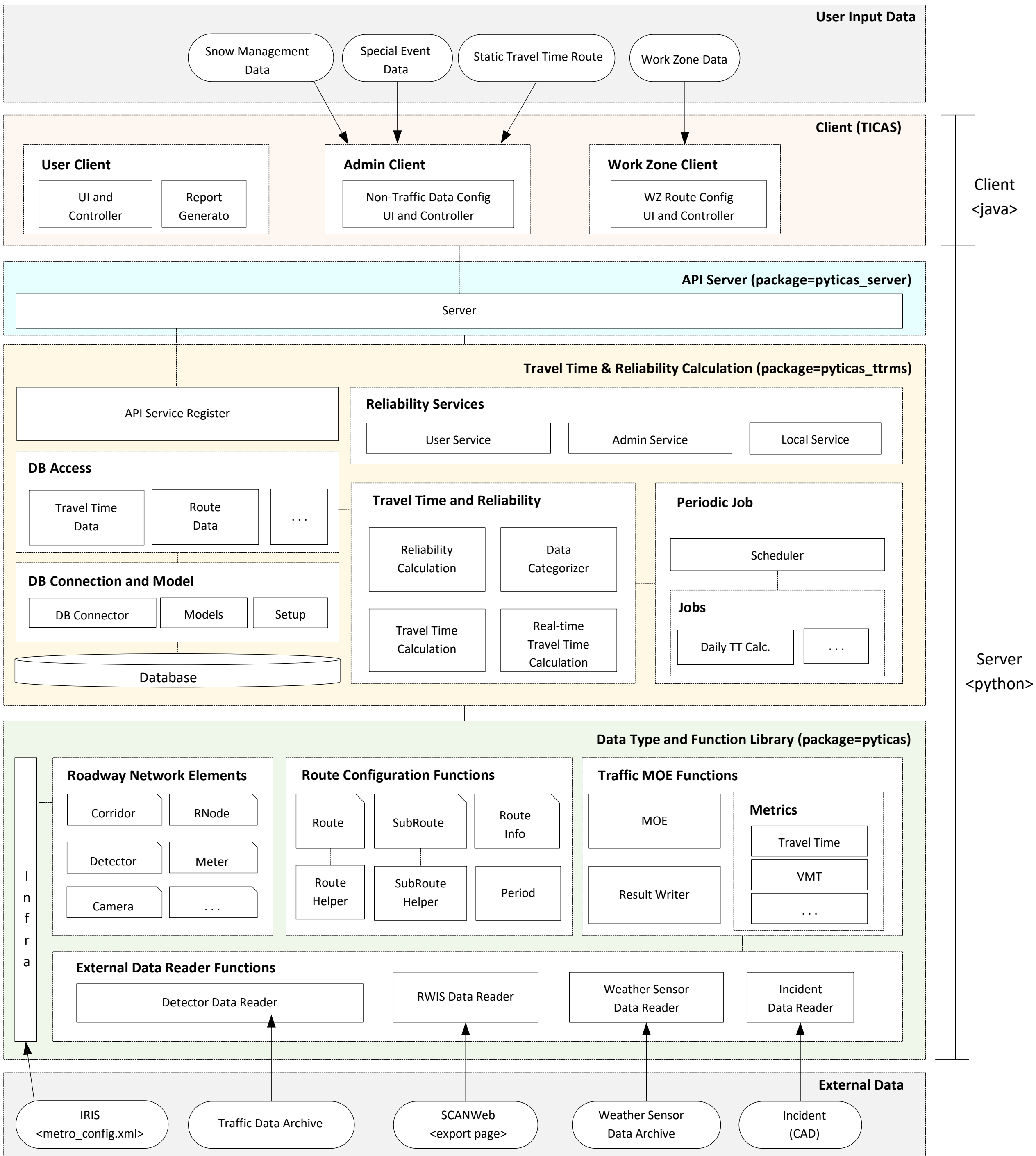


Figure 3.2.1 Framework for Travel-Time Reliability Measurement System

Required Data for Work-Zone Route Configuration

The following types of the geometry data are needed for configuring the travel-time routes for a given work-zone. These data will be entered through the user-interface client and stored in JSON format for each work-zone.

- Location/IDs of lane-closure sections,
- Location/IDs of lane-shifting and shifted-lanes,
- Location of crossover points,
- Work-zone geometry data including type/width of Median and shoulder, speed limit,
- Ramp-closure information within a work-zone.

Design of Data Structure

To store the above data and configure the travel-time route for a given work-zone, the following classes were developed:

- *'Infra'*: a class containing all the geometry and detector information of the metro freeway-network, such as the ID information for each corridor, detector, station, entrance/exit ramp, meter and camera.
- *'Route'*: a class representing a freeway route,
- *'RouteInfo'*: a class storing the lane-configuration data for each work-zone,
- *'SubRoute'*: a class representing a sub-route in a Route, as shown in Figure 2,
- *RouteHelper*, *SubRouteHelper* and *RouteInfoHelper* are the classes to process the above classes.

Figure 3.2.2 includes the details of the RouteInfo class. The class-diagram showing the relationships among the above classes are presented in Figure 3.2.3.

Route-Configuration Process

Figure 3.2.4 shows the process to configure a travel-route in a work-zone. The step-by-step procedure to create a travel-route is as follows:

- Step 1: Using the map-based interface in the Client program, the sub-routes in a given work-zone are defined by user.
- Step 2: For each sub-route defined in Step 1, user enters the lane-configuration information using the input dialog to be developed in this task.

- Step 3: The user-entered data regarding the sub-routes and their lane-configuration are converted to JSON format.

- Step 4: The route-data in JSON format are sent to the TTRMS server, where the data for each route are stored.

- Step 5: Using the JSON-format route-data, the TICAS service module in the TTRMS server creates the ‘Route’ instance, which is used by the Travel-Time calculation module.

```
class RouteInfo(object):
def __init__(self):
        self.has_crossover = None         """:type: bool """
        self.crossover_from = None        """:type: str """
        self.crossover_to = None          """:type: str """
        self.crossover_lanes = None       """:type: int """
        self.has_lane_close = None        """:type: bool """
        self.lane_close_from = None       """:type: str """
        self.closed_lanes = None          """:type: int """
        self.closed_lanes_list = []       """:type: list[int] """
        self.has_lane_shift = None        """:type: bool """
        self.lane_shift_from = None       """:type: str """
        self.shifted_lanes = None         """:type: int """
        self.shifted_lanes_list = []      """:type: list[int] """
        self.shift_direction = None       """:type: str """
        self.median_type = None           """:type: str """
        self.median_width = None          """:type: int """
        self.shoulder_type = None         """:type: str """
        self.shoulder_width = None        """:type: int """
        self.lane_width = None            """:type: int """
        self.speed_limit = None           """:type: int """
        self.closed_ramp_names = []       """:type: list[str] """
        self.route_start_offset = None    """:type: int  """
        self.route_end_offset = None      """:type: int  """
        self.rnode_names = []             """:type: list[str] """
        self.rnodes = []                  """:type: list[RNodeObject] """
```

Figure 3.2.2 ‘RouteInfo’ class for storing lane-configuration data

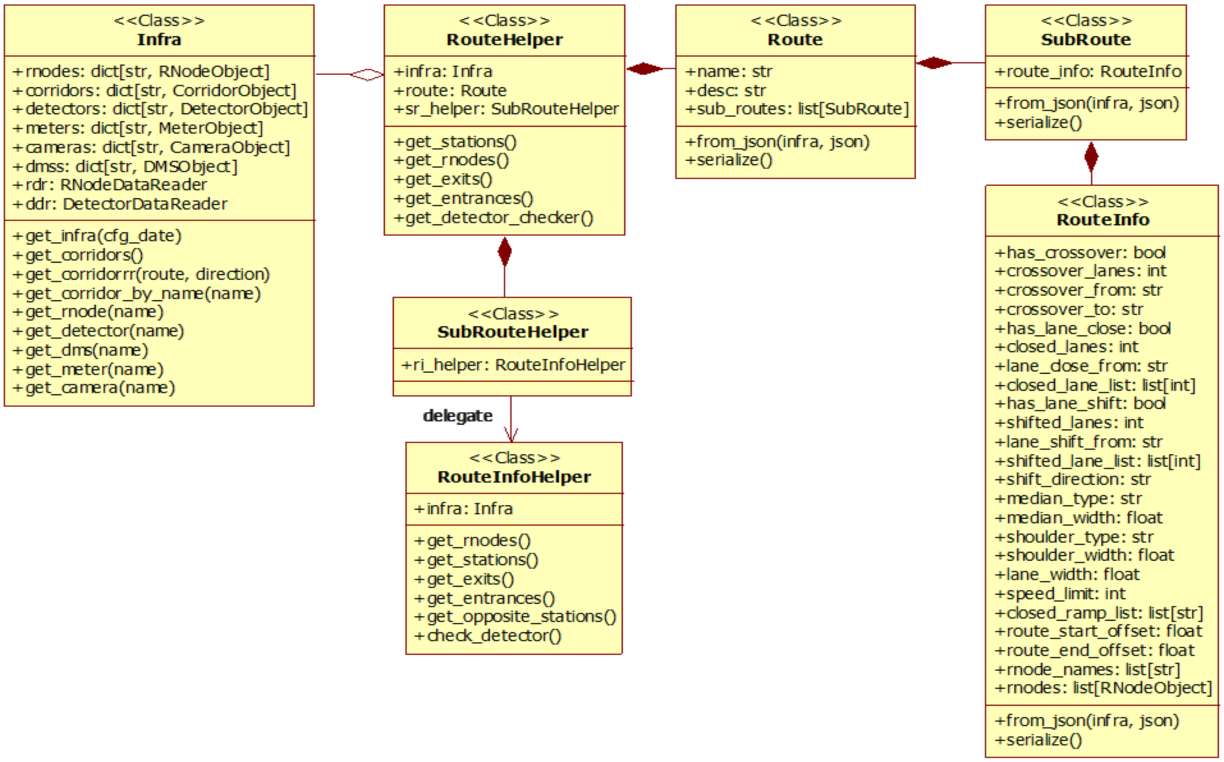


Figure 3.2.3 Class-diagram for Route-Configuration

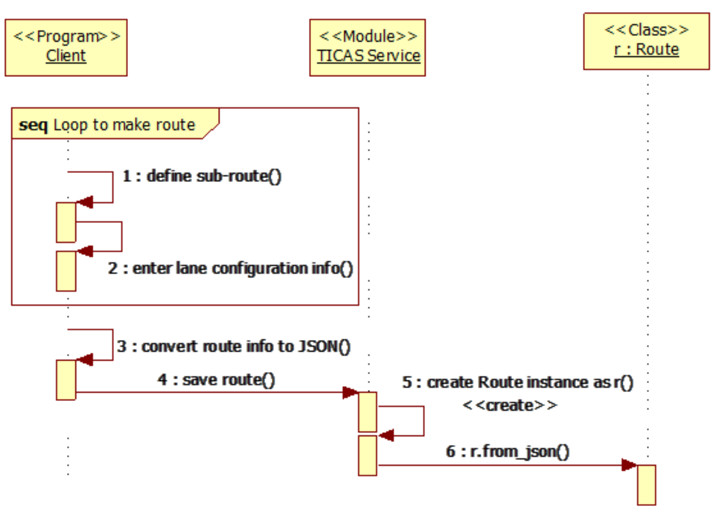


Figure 3.2.4 Process Diagram for Configuration of Travel-Routes

3.3 ENHANCEMENT OF THE TRAVEL-TIME CALCULATION MODULE FOR WORK-ZONES

3.3.1 Basic Methodology for Travel-time Estimation

Figure 3.3.1 illustrates the principles of estimating the travel-time for a route in TICAS. Using the speed data from each detector station located on a travel-route, the travel-time estimation procedure first determines the speed estimates of the equal-length subsections between two stations for each time interval, whose value is defined by user. The speed values of each subsection for each time interval are then applied to determine the travel-trajectory of a vehicle leaving the first station at the beginning of each time interval until it reaches the last station of a given route. The travel-time is calculated as the difference between the departure time at the first station and the arrival time at the last station of a given route. Figure 3.3.2 includes the pseudo-code of the travel-time calculation function, *estimate_travel_time()*, developed in this study to determine the travel-time of a route including work-zones.

3.3.2 Design of the Modules for Calculating Work-Zone Route Travel-Time

Figure 3.3.3 shows the classes developed for the travel-time calculation process, which is initiated by calling *estimate_travel_time()* in the Estimation module. The main functionalities of each module in this process are as follows:

- *Measure* module provides the functions to calculate the flow-measures, such as travel-time, speed, density, flow, and VMT, etc. Those measure-calculation functions are stored in *Measure* package.
- *MeasureHelper* module provides the data gathering methods to *Measure* module.
- *TravelTime* module calculates the travel-time of a given route.
- *RNodeData* is a class to organize travel- time calculation results. A *RNodeData* instance contains *RNode* instance, time-period, measurement-type, station data list and lane-by-lane data for a station. The results from *TravelTime* module are stored in an array (list) of *RNodeData* instance, which contains the travel-time value from the starting station to each downstream station along a route.
- *RNode* is a class representing a detector station.
- *ResultWriter* module stores the travel-time calculation results to a file in a spread- sheet format.

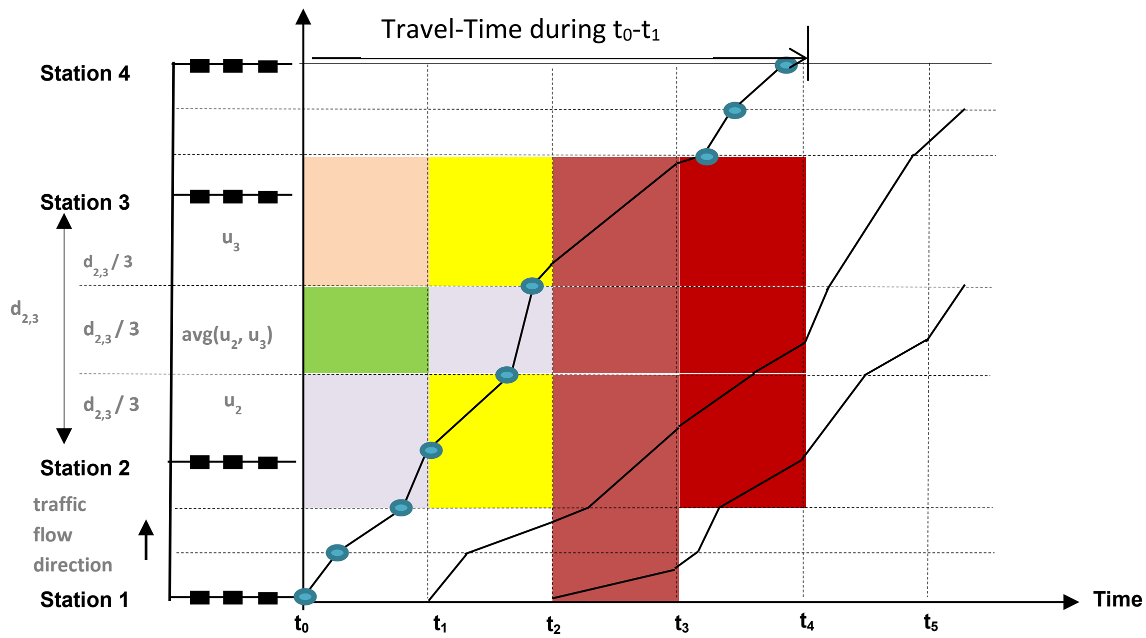


Figure 3.3.1 Vehicle-trajectories leaving the first station in the beginning of each time interval

```
FUNCTION estimate_travel_time(rnode_list, period):
    """
    estimate travel time with the given rnode (station) list and time period information
    * RNode represents station, entrance and exit
    """
    CALLget_speed() WITH rnode_list, period RETURN us_data
    SET tt_data TO list of float list
    # us_data and tt_data is list[ data list for a rnode ] (type: list[list[float]])
   # e.g.  us_data = [
   #      [ 65, 63, 50, 51, 42, 40.. ] ← speed list of station 1
   #      [ 70, 67, 48, 50, 45, 46.. ] ← speed list of station 2
   #      [ 68, 60, 54, 52, 41, 43.. ] ← speed list of station 3
  #      ...    # ]
    SET len_data TO length of data for a rnode
    FOR tidx TO len_data:
      SET partial_data TO data only after tidx
      CALL calculate_tt() WITH partial_data, period.interval RETURN tts
      FOR ridx TO lengh of tt_data:
         SET tt_data[ridx][tidx] TO tts[ridx]
      END FOR
   END FOR
   RETURN tt_data
END FUNCTION
```

```
FUNCTION calculate_tt(data, interval):
  """
  calculate travel times from the first station to each station
   when vehicle start to travel at the first time step"""
  SET tts To list of float
  SET p To (0, 0, tts, 0)
  # p[0] = traveled time
  # p[1] = traveled distance
  # p[2] = traveled time list for each rnode
  # p[3] = current rnode index
  WHILE True:
     Call tt_next() WITH data, interval p; RETURN p
     # p[0] = None, when time index or rnode index is over its limit
     # traveled distance (P[1]) could not exceed length of route
     IF NOT p[0] OR p[1] >= length of route:
        BREAK
     END IF
  END WHILE
  RETURN tts
END FUNCTION

FUNCTION tt_next(data, interval, p):
  """
  store the travel time and find the travel time, distance to next time step"""
  ASSIGN (tt, td, tts, cur_ridx) FROM p
  SET vd TO 0.1                    # distance in mile between rnodes
  SET max_ridx TO len(data)
  SET max_tidx TO len(data[0])
  SET ridx TO floor(td / vd)       # rnode index
  SET tidx TO floor(tt / interval)   # time index

  # check end condition
  IF it is over end of route or time data THEN
    RETURN (None, None, None, None)
  END IF

  # store travel time into the result list
  IF current rnode is not the end of route THEN
    tts[ridx] = tt / 60.0
  END IF

  # calculate remaining time and distance to next node
  SET remaining_interval TO ( interval * (tidx + 1) ) - tt
```

```
    SET remaining_distance TO ( vd * ( ridx + 1) ) - td

    # calculate travel time and distance to next node
    SET u TO data[ridx][tidx]
    SET tt_for_remaining_distance TO remaining_distance / u * seconds_per_hour
    SET tt_to_go TO min(remaining_interval, tt_for_remaining_distance)
    SET d_to_go TO u * tt_to_go / seconds_per_hour

    RETURN (tt + tt_to_go, td + d_to_go, tts, cur_ridx)
END FUNCTION
```

Figure 3.3.2 Pseudo-code of the travel-time function

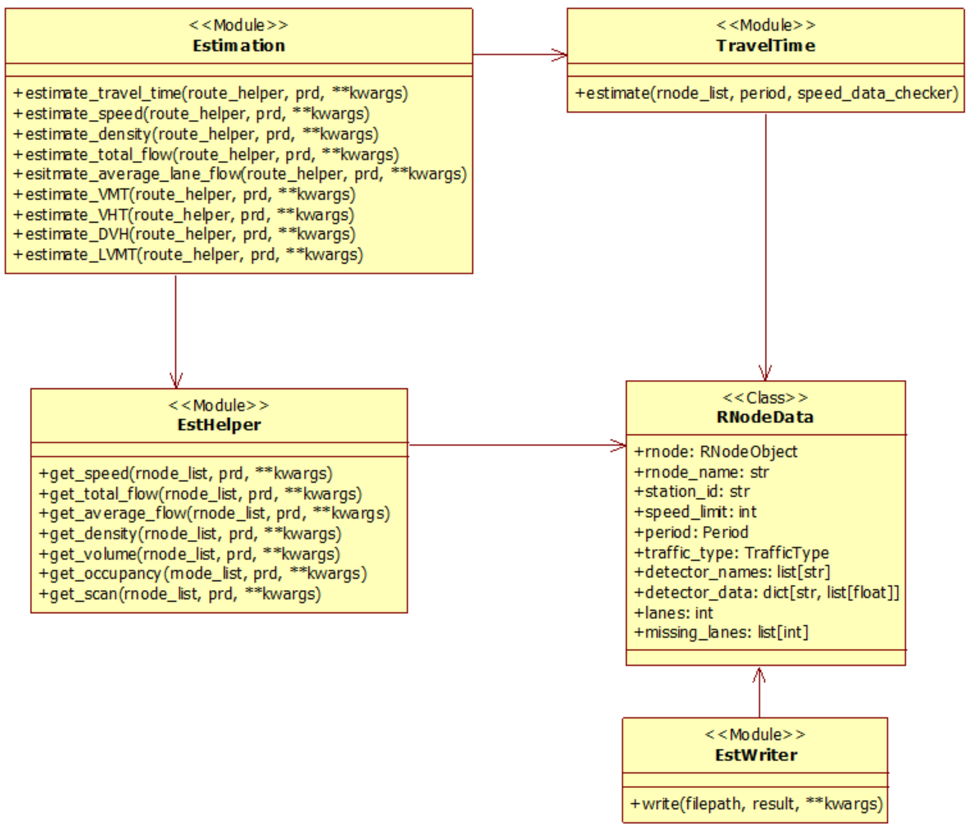


Figure 3.3.3 Class Diagram of travel-time calculation process

Travel-Time Calculation Process

Figure 3.3.4 shows the sequence-diagram of the travel-time calculation procedure, whose first step is to call the function, *travel_time(),* in the *Measure* module with the following parameters:

- *RouteHelper* instance: an object including *Route* instance and the functions necessary for configuring a route, such as developing a list of the stations for a given route and identifying the detectors on the traveling-lanes.

- *Period* instance: the time period information including the duration and data interval in seconds

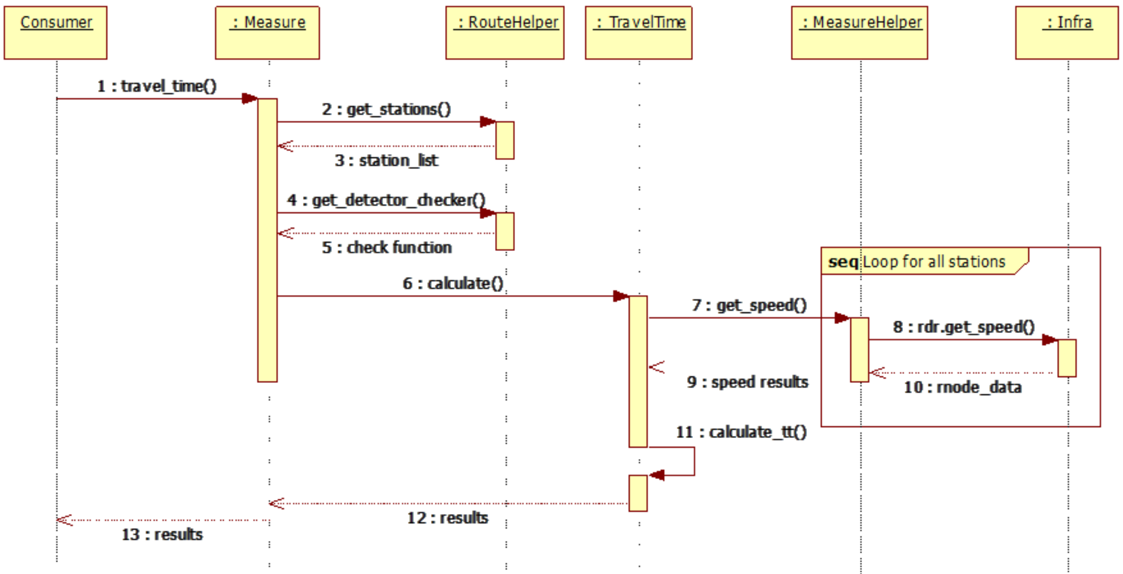


Figure 3.3.4 Sequence-diagram of travel-time calculation process

After *travel_time()* is called, the *Measure* module develops the station-list for a given route by calling *get_stations*(), which combines all the detector stations in the sub-routes of a work-zone. If there is a crossover to an opposite direction, the stations in the opposite lanes are returned. Figure 3.3.5 includes the detailed sequence-diagram for the *get_stations()* function, whose pseudo-code is also presented in Figure 3.3.6.

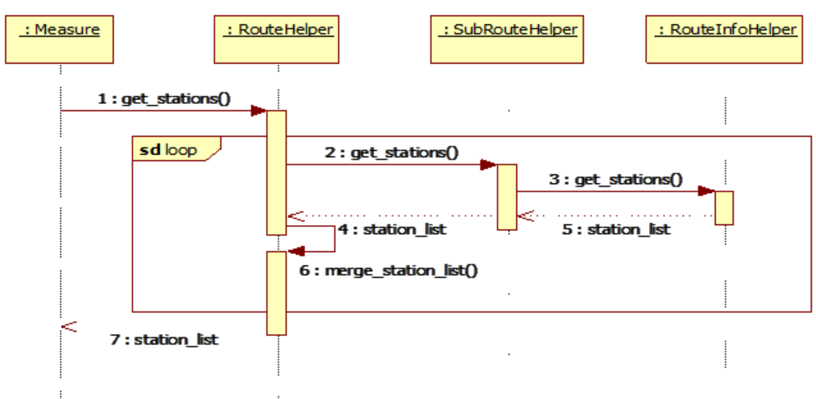


Figure 3.3.5 Sequence-diagram for developing station list

FUNCTION **get_stations**():
IF the sub-route does not have crossover OR the opposite direction crosses over to the route THEN
RETURN station list of *RNodeInfo* class object
ELSE THEN
CALL *get_opposite_stations()* RETURN *opposite_stations*
RETURN *opposite_stations*
END IF
END FUNCTION

FUNCTION **get_opposite_stations()**:
SET *stations* TO station list of *RNodeInfo* class object
SET *orn_s* TO nearby station in the opposite direction for the first station
SET *orn_e* TO nearby station in the opposite direction for the last station
RETURN station list from *orn_s* to *orn_e*
END FUNCTION

Figure 3.3.6 Pseudo code of *get_stations()* function in *RouteInfoHelper*

After the list of the stations for a given route is developed, the *RouteHelper* in Figure 3.3.7 creates the *check_detector* function, which examines the validity of a detector by returning *False* if a given detector does not belong to any of sub-routes. The '*check_detector*' function is also used to delete the detectors on the closed-lanes when the traffic-data is collected from each station in a given work-zone. Figure 3.3.7 shows the sequence-diagram of the *get_detector_checker()* function. The pseud-code of the *check_detector()* function is presented in Figure 3.3.8.

Once all the detectors in the travel-lanes for a given route are identified and assembled, the travel-time calculation function, *estimate_travel_time()* in the *TravelTime* module, is called and the travel-times from the most upstream station to each station downstream in a route are calculated.

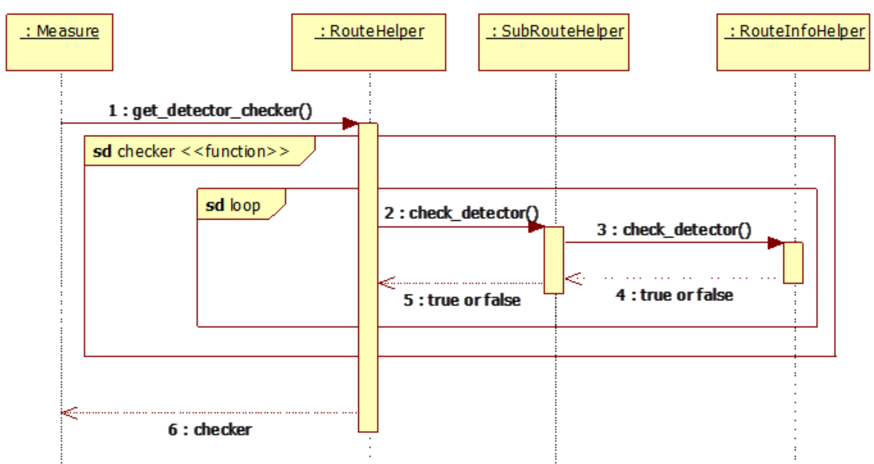


Figure 3.3.7 Sequence-diagram for checking detectors

```
FUNCTION check_detector(det):

    IF it has lane close:
        RETURN True if the given detector exists in closed_lane_list of RouteInfo
    END IF

    IF it has lane shift:
        RETURN True if the given detector exists in shifted_lane_list of RouteInfo
    END IF

    IF it has cross over to opposite lane:
        IF the given detector does not exist in the opposite direction
            RETURN False
        END IF
        RETURN (det.lane > station.lanes  – crossover lanes)
    END IF

    IF it has cross over from opposite lane:
        IF the given detector does not exist in the route
            RETURN False
        END IF
        RETURN (det.lane <= station.lanes – crossover lanes)

    END IF

    RETURN True if the given detector exists, or False

END FUNCTION
```

Figure 3.3.8 Pseudo-code of the *check_detector()* function in *RouteInfoHelper*

3.4 DEVELOPMENT AND TESTING OF THE ROUTE-CONFIGURATION AND TRAVEL-TIME CALCULATION MODULES

The route-configuration and the travel-time modules designed in the previous sections were implemented in Phython and incorporated into the TTRMS server, while the user-interface client was developed in Java. The server program consists of a main script and three Python packages: *pyticas*, *pyticas_server*, *pyticas_ttrms.* Figure 3.4.1 shows the main script, where the server instance is created and the TTRMS service is added to the server instance as an application. Figure 3.4.2 includes the screenshot of the graphical user-interface developed in this task for entering the lane-configuration data for a work-zone.

```
from pyticas_server.server import TICASServer
from pyticas_ttrms.app import TTRMSApp

# create server instance
ticasServer = TICASServer(data_path=”./data”)
# add app
ticasServer.add_app(TTRMSApp("TTRMS: Travel Time Reliability Management System"))
# start server
ticasServer.start(port=5000)
```

Figure 3.4.1 Main script for operating TTRMS server

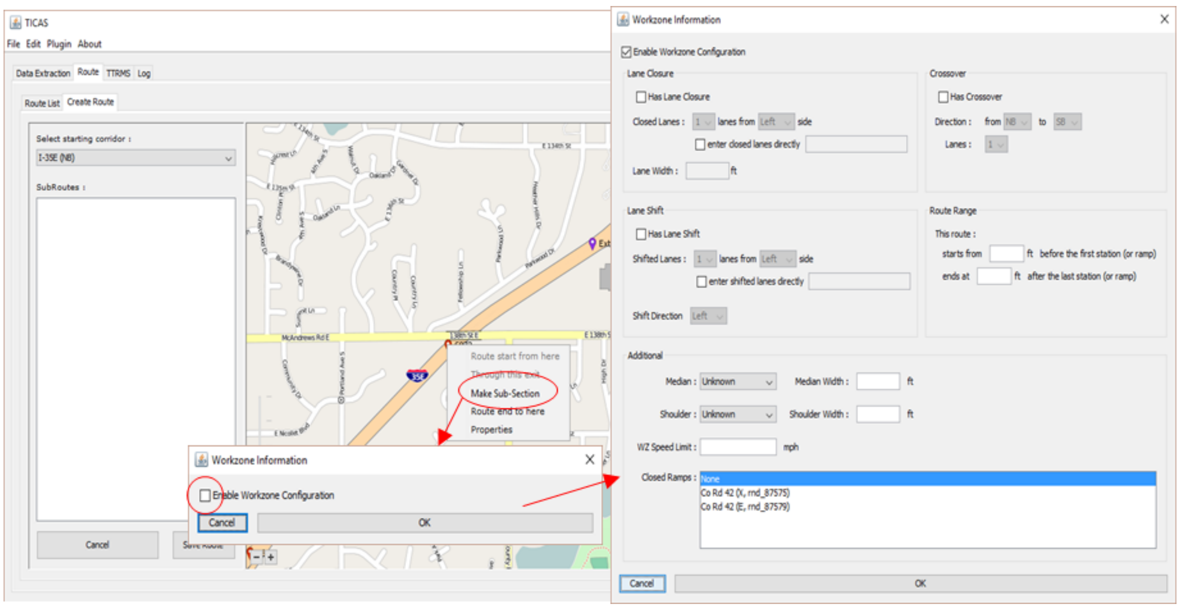


Figure 3.4.2 Graphical User-Interface for Work-Zone Route/Lane-Configuration

Testing Work-Zone Route-Configuration and Travel-Time Calculation Modules

In this section, the work-zone route-configuration and travel-time calculation modules were tested with the real work-zones in the metro freeway network. The main focus of the testing is to examine 1) if the route configuration module correctly identifies the detectors on the travel-lanes for given work-zones and 2) calculates the travel-times for the travel-routes with the speed data from selected detectors. Three work-zone cases with the different types of lane-configurations were used in this testing: a lane-closure, a lane-closure with a crossover to/from an opposing direction.

Figure 3.4.3 shows the lane-configuration of the I-694 EB work-zone that has one-lane closure in each direction without a crossover. The screenshot of the route-configuration panel applied for this site is included in Figure 3.4.4. First, the route-configuration module was applied to identify the detectors on the EB open-lane and the speed data from those detectors were collected. Table 3.4.1 shows the IDs and speed data from those detectors on the travel-lane. Finally, the travel-time module calculated the travel-times of the travel-route for each time interval with those speed data. Table 3.4.2 includes the travel-time calculation results for the I-694 EB work-zone on June 19th, 2012. In this testing, the IDs of the selected detectors by the route-configuration module were validated through the manual comparison between those in Figure 3.4.3 and Table 3.4.1. Further, the station speed data in Table 3.4.1 were also verified with the data downloaded separately using the current TICAS. Finally, the travel-time results in Table 3.4.2 were confirmed with the manual calculation using the station speed and the distance data between stations.

The above testing process was also applied to the other two sites, which include

- I-35E NB (split to TH77), June 18th, 2013: One lane-closure with a crossover to the SB left-most lane,
- I-35E NB (split to TH77), July 9th, 2013: One-lane closure with a crossover from the SB traffic, i.e., the NB left-lane was used by the SB traffic flow.

In particular, the above two cases include the crossovers to or from an opposing lane and the identification of the correct detectors on the travel-route would be of critical importance for the accurate calculation of travel-times. Figures 3.4.5-3.4.8 show the lane-configurations of those two cases and the Tables 3.4.3-3.4.6 include the speed data and the travel-time calculation results for the travel-routes in those two cases. As in the first case, the IDs of the detectors automatically selected by the route-configuration module were manually verified with the actual lane-configuration data for both cases. Further, the speed data and the travel-time results for each case were also confirmed with the manual calculation results.

I-694 EB (N Jct I-35E to 40th St): 7 to 8AM, 19th June 2012

RampName				E:T.H.61		X:T.H.61		E:White Bear Ave		X:White Bear Ave		E:Century Ave		X:Century Ave		E:T.H.36 W	X:T.H.36 V	E:T.H.36 EB		X:T.H.36 EB		
Ramp				0		0		0		0		0		0		0	0	0		0		
lane 1			X		<-		<-		<-		<-		<-		<-				<-		<-	<-
lane 2			X		<-		<-		<-		<-		X		<-				<-		<-	<-
lane 3			X																			
WB <--			S1445		S1456		S1424		S1423		S1422		S1421		S1420				S1419		S1418	S1417
Div																						
EB -->	S1454				S1455		S1393		S1394		S1395		S1396		S1397				S1398		S1399	S1400
lane 3	X				X																	
lane 2	->				->		X		X		X		X		X				X		X	X
lane 1	->				->		->		->		->		->		->				->		->	->
Ramp		0		0		0		0		0		0		0		0	0	0		0		
RampName		E:I-35E SB		X:T.H.61		E:T.H.61		X:White Bear Ave		E:White Bear Ave		X:Century Ave		E:Century Ave		X:T.H.36 V	E:T.H.36 W	X:T.H.36 EB		E:T.H.36 EB		

Figure 3.4.3 Lane configuration of I-694 EB case

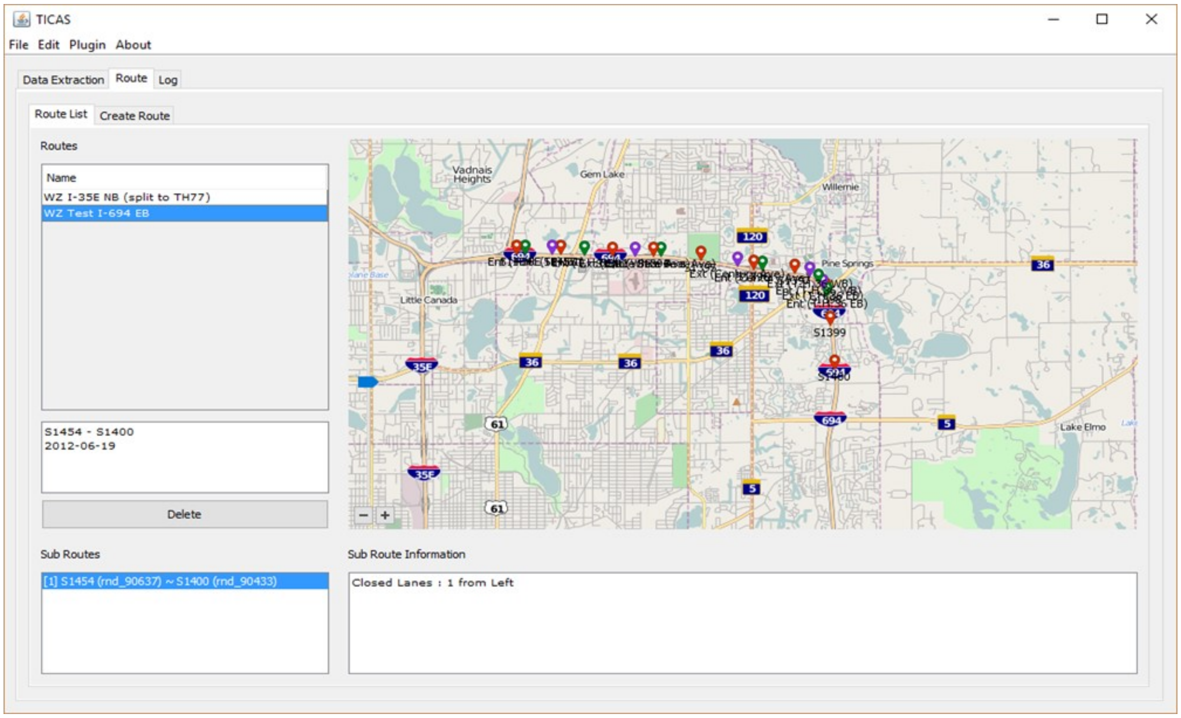


Figure 3.4.4 Screenshot of the I-694 EB route created in the client program

Table 3.4.1 Selected Detectors and Speed Data on the Travel-route in I-694 EB work-zone

	N Jct I-35E (S1454) 2/3 lanes	T.H.61 (S1455) 2/3 lanes	E of T.H.61 (S1393) 1/2 lanes	White Bear Ave (S1394) 1/2 lanes	McKnight Rd (S1395) 1/2 lanes	Century Ave (S1396) 1/2 lanes	W of T.H.36 (S1397) 1/2 lanes	T.H.36 (S1398) 1/2 lanes	50th St (S1399) 1/2 lanes	40th St (S1400) 1/2 lanes
Used Detectors (Lane:ID)	*1:5513 2:5514*	*1:5517 2:5518*	*1:6182*	*1:6185*	*1:6188*	*1:6191*	*1:6194*	*1:6200*	*1:6206*	*1:6208*
Distance	0	0.7	1.5	2.1	2.9	3.7	4.3	4.9	5.5	6.2
07:05:00	70.2450	70.4566	68.6751	54.0248	22.9619	31.3629	16.0908	22.0186	48.0953	64.5759
07:10:00	72.4638	68.4692	68.4821	58.0743	47.2945	11.8531	23.4647	24.9547	44.8192	68.3284
07:15:00	73.5799	75.2370	74.7033	68.8092	39.9132	27.0638	34.2688	17.7647	45.6441	60.2875
07:20:00	66.8569	63.1279	62.1610	63.4275	63.8496	18.7008	10.5475	16.7774	48.3223	61.3817
07:25:00	69.5400	64.2741	57.2482	59.0705	58.8898	7.5798	15.2840	24.5803	52.5885	74.3924
07:30:00	62.6898	65.7855	55.2284	49.2908	23.6904	18.1732	22.8312	22.1103	51.9969	75.2951
07:35:00	66.1267	68.1839	46.3186	26.3533	39.4983	11.3336	23.2600	16.1583	55.9254	72.5288
07:40:00	61.5135	56.5383	21.9269	58.4026	26.9005	21.7348	18.0034	15.0331	46.7167	68.4402
07:45:00	67.1471	14.5139	30.3370	43.0363	34.4891	5.6500	14.5203	14.7679	50.9065	71.0790
07:50:00	66.6978	18.5011	39.9961	48.9401	25.1016	7.6784	10.5887	15.3442	55.5585	77.9456
07:55:00	65.8537	41.9044	38.2130	28.0039	17.9319	7.8355	11.7924	15.7425	55.2021	77.3010
08:00:00	70.8903	66.0400	49.6674	11.2323	17.8833	9.3926	9.9907	16.1211	48.1276	74.9212

Table 3.4.2 Calculated travel-time of I-694 EB Work-Zone

	N Jct I-35E (S1454) 2/3 lanes	T.H.61 (S1455) 2/3 lanes	E of T.H.61 (S1393) 1/2 lanes	White Bear Ave (S1394) 1/2 lanes	McKnight Rd (S1395) 1/2 lanes	Century Ave (S1396) 1/2 lanes	W of T.H.36 (S1397) 1/2 lanes	T.H.36 (S1398) 1/2 lanes	50th St (S1399) 1/2 lanes	40th St (S1400) 1/2 lanes
Used Detectors (Lane:ID)	*1:5513 2:5514*	*1:5517 2:5518*	*1:6182*	*1:6185*	*1:6188*	*1:6191*	*1:6194*	*1:6200*	*1:6206*	*1:6208*
Distance	0	0.7	1.5	2.1	2.9	3.7	4.3	4.9	5.5	6.2
07:05:00	0	0.5971	1.2861	1.8653	3.1887	5.0474	7.3348	8.8301	10.0000	10.7771
07:10:00	0	0.5938	1.2948	1.8578	2.7631	5.0000	6.1024	7.4822	8.8569	9.6725
07:15:00	0	0.5654	1.2054	1.7040	2.5974	4.0293	5.6939	8.5067	10.0000	10.5942
07:20:00	0	0.6438	1.4093	1.9836	2.7382	4.0979	6.6340	8.5663	9.6351	10.3204
07:25:00	0	0.6246	1.4105	2.0312	2.8449	5.1148	6.9159	8.5139	9.6578	10.3460
07:30:00	0	0.6562	1.4446	2.1281	3.4930	6.1517	8.5091	10.3328	11.8425	12.6038
07:35:00	0	0.6268	1.4643	2.4464	3.9800	5.7025	7.4938	10.0000	11.1215	11.8351
07:40:00	0	0.7078	2.0322	3.1304	4.3067	7.9915	12.6223	15.5166	16.8950	17.5522
07:45:00	0	1.1799	3.6183	4.6439	6.0189	10.0000	13.2836	16.0140	17.4433	18.1628
07:50:00	0	1.0676	2.9560	3.7834	5.1616	9.0940	12.9796	15.7608	17.1202	17.8965
07:55:00	0	0.7708	1.9638	3.0357	5.1317	8.7357	11.7947	13.9320	15.2807	16.0123
08:00:00	0	0.6107	1.4360	2.8029	5.5991	8.0015	11.3882	15.0516	16.2922	17.0398

I-35E NB (split to TH77): 7 to 8AM 18th June 2013

RampName		E:Co Rd 42		X:Co Rd 42		E:Co Rd 11			X:Co Rd 11		
Ramp		O		O		O			O		
lane 1	<-		<-		<-		<-	<-		<-	
lane 2	<-		<-		->		->	->		->	
lane 3											
SB <--	S905		S904		S903		S902	S901		S900	
Div											
NB -->	S870		S871		S872		S873	S874		S875	
lane 3											
lane 2	->		X		X		X	X		X	
lane 1	X		->		X		X	X		X	
Ramp		O		O		X			X		O
RampName		X:Co Rd 42		E:Co Rd 42		X:Co Rd 11			E:Co Rd 11		X:I-35E CD

Figure 3.4.5 Lane configuration of I-35E NB case (June 18th, 2013)

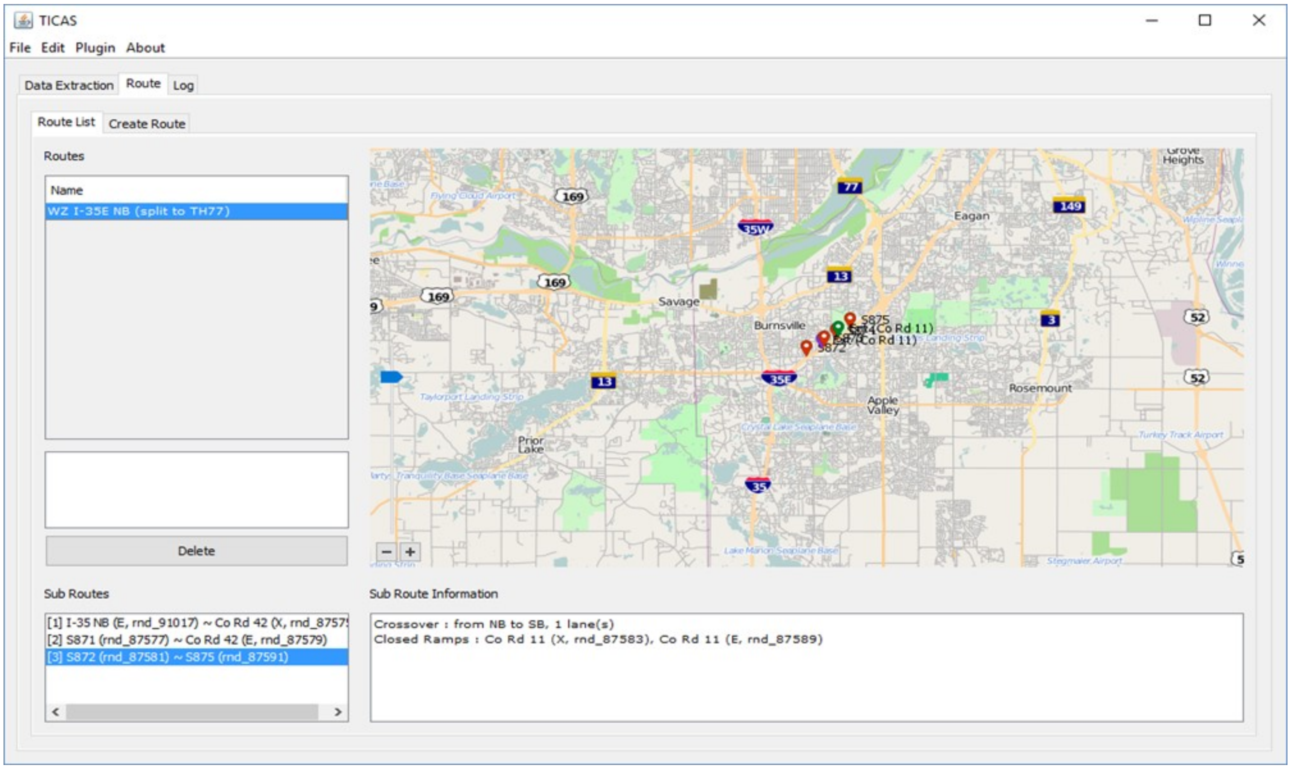


Figure 3.4.6 I-35E NB case (June 18th, 2013) created in client program

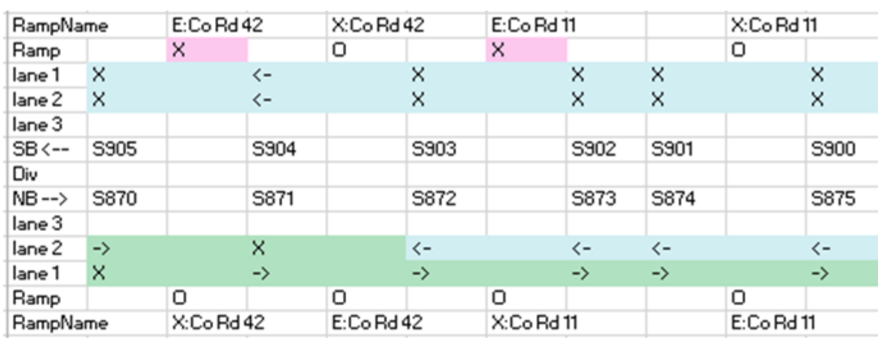
Table 3.4.3 Selected Detectors and Speed Data on the travel-lanes of I-35E NB case (June 18th, 2013)

	Southcross Dr (S870) 1/2 lanes	Co Rd 42 (S871) 1/2 lanes	McAndrews Rd (S903) 1/2 lanes	S of Co Rd 11 (S902) 1/2 lanes	Co Rd 11 (S901) 1/2 lanes	N of Co Rd 11 (S900) 1/2 lanes
Used Detectors (Lane:ID)	*2:3701*	*1:3703*	*2:3833*	*2:3830*	*2:3828*	*2:3825*
Distance	0	0.6	1.4	2	2.4	3
07:00:00	73.38738	80.4959	72.40175	74.85773	80.47823	73.5794
07:05:00	70.88932	82.81104	71.10476	73.14209	75.76596	77.08986
07:10:00	70.87189	77.86545	72.37777	65.41239	69.92276	70.09153
07:15:00	71.11661	84.2419	71.68507	71.44966	75.75714	74.56452
07:20:00	71.89279	79.75559	71.81819	72.94343	75.10264	71.24823
07:25:00	73.36661	85.28239	71.93161	69.80221	73.34179	76.87305
07:30:00	71.71628	82.42224	72.90091	72.95775	77.28061	77.23371
07:35:00	73.39112	78.68604	71.27584	75.51968	76.38476	72.22324
07:40:00	72.71953	85.71766	71.54477	72.85648	78.45469	72.86346
07:45:00	68.87064	70.32054	72.40661	71.99642	74.90305	69.29931
07:50:00	72.09765	88.01359	72.27133	72.57216	73.76587	75.49609
07:55:00	67.60372	81.84997	72.14625	74.45962	76.91878	79.29077
08:00:00	75.12775	72.54606	71.66609	75.58397	76.04802	73.00578

Table 3.4.4 Calculated travel time of I-35E NB case (June 18th, 2013)

	Southcross Dr (S870) 1/2 lanes	Co Rd 42 (S871) 1/2 lanes	McAndrews Rd (S903) 1/2 lanes	S of Co Rd 11 (S902) 1/2 lanes	Co Rd 11 (S901) 1/2 lanes	N of Co Rd 11 (S900) 1/2 lanes
Used Detectors (Lane:ID)	*2:3701*	*1:3703*	*2:3833*	*2:3830*	*2:3828*	*2:3825*
Distance	0	0.6	1.4	2	2.4	3
07:00:00	0	0.471997338	1.096804667	1.587164972	1.899072748	2.363404758
07:05:00	0	0.475954054	1.095975553	1.596344911	1.920136957	2.391866719
07:10:00	0	0.488413011	1.124989219	1.643778553	2.001510582	2.515847498
07:15:00	0	0.471682514	1.083629093	1.58651446	1.915043576	2.393398157
07:20:00	0	0.479536315	1.109822501	1.60785132	1.933274406	2.423252545
07:25:00	0	0.46083268	1.067708556	1.574488097	1.911940258	2.393265257
07:30:00	0	0.473678077	1.088447592	1.582108252	1.903966867	2.369919547
07:35:00	0	0.476481851	1.113660072	1.606708635	1.923149204	2.405572411
07:40:00	0	0.46223421	1.068869817	1.568255976	1.888529973	2.361739626
07:45:00	0	0.518200352	1.192131299	1.690501735	2.018904766	2.51534199
07:50:00	0	0.459455884	1.054610742	1.551872163	1.880554279	2.363892168
07:55:00	0	0.49168717	1.111702538	1.604168293	1.922563531	2.384702746
08:00:00	0	0.486214552	1.15140824	1.642715447	1.959514282	2.441018135

I-35E NB (split to TH77), 9th July 2013



RampName		E:Co Rd 42		X:Co Rd 42		E:Co Rd 11			X:Co Rd 11	
Ramp		X		O		X			O	
lane 1	X		<-		X		X	X		X
lane 2	X		<-		X		X	X		X
lane 3										
SB <--	S905		S904		S903		S902	S901		S900
Div										
NB -->	S870		S871		S872		S873	S874		S875
lane 3										
lane 2	->		X		<-		<-	<-		<-
lane 1	X		->		->		->	->		->
Ramp		O		O		O			O	
RampName		X:Co Rd 42		E:Co Rd 42		X:Co Rd 11			E:Co Rd 11	

Figure 3.4.7 Lane configuration of I-35E NB work-zone (9th July 2013)

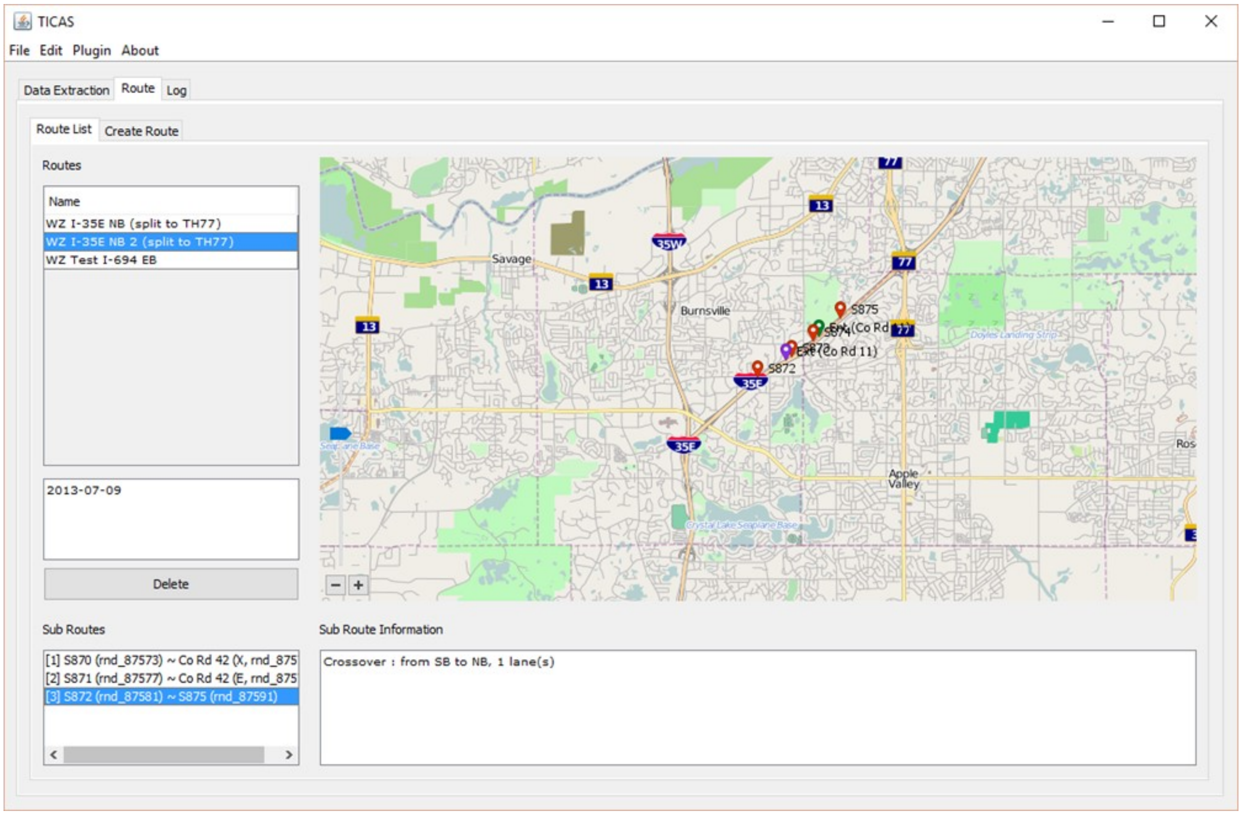


Figure 3.4.8 I-35E NB work-zone (9th July 2013) created in client program

Table 3.4.5 Selected Detectors and Speed Data on the travel-lanes of I-35E NB work-zone (July 9th, 2013)

	Southcross Dr (S870) 0/2 lanes	Co Rd 42 (S871) 1/2 lanes	McAndrews Rd (S872) 1/2 lanes	S of Co Rd 11 (S873) 1/2 lanes	Co Rd 11 (S874) 1/2 lanes	N of Co Rd 11 (S875) 1/2 lanes
Used Detectors (Lane:ID)	*2:3701*	*1:3703*	*1:3707*	*1:3709*	*1:3712*	*1:3716*
Distance	0	0.6	1.4	2	2.4	3
07:05:00	56.71865	68.23322	62.64995	65.34987	68.25299	63.84069
07:10:00	58.92379	68.55757	64.30286	69.55743	68.64977	65.48608
07:15:00	55.68993	58.72029	59.09189	61.63366	62.04305	57.97938
07:20:00	59.88649	63.75212	59.81076	41.36438	28.49986	18.39183
07:25:00	61.31178	69.04297	29.56312	24.97482	23.18537	42.11981
07:30:00	56.6512	25.41004	24.4194	34.13962	24.87503	47.85533
07:35:00	50.93543	9.329138	23.49624	28.37316	38.93668	48.32745
07:40:00	11.34843	9.145341	24.91671	23.96184	38.30469	44.61817
07:45:00	10.82176	8.108953	18.22778	27.88203	37.05413	50.87485
07:50:00	9.032855	7.209054	36.14239	19.8912	24.91597	48.94181
07:55:00	6.653098	9.987665	17.51786	22.77286	49.21652	48.84597
08:00:00	10.86724	4.549702	38.1972	43.03404	54.25733	50.92407

Table 3.4.6 Calculated travel time of I-35E NB work-zone (July 9th, 2013)

	Southcross Dr (S870) 0/2 lanes	Co Rd 42 (S871) 1/2 lanes	McAndrews Rd (S872) 1/2 lanes	S of Co Rd 11 (S873) 1/2 lanes	Co Rd 11 (S874) 1/2 lanes	N of Co Rd 11 (S875) 1/2 lanes
Used Detectors (Lane:ID)	*2:3701*	*1:3703*	*1:3707*	*1:3709*	*1:3712*	*1:3716*
Distance	0	0.6	1.4	2	2.4	3
07:05:00	0	0.527602	1.257998	1.822603	2.183871	2.726206
07:10:00	0	0.525106	1.245237	1.787051	2.133789	2.668596
07:15:00	0	0.613076	1.428259	2.026879	2.41531	3.012153
07:20:00	0	0.564687	1.339013	2.040517	2.700854	4.215868
07:25:00	0	0.521414	1.553122	2.859366	3.847114	5.051392
07:30:00	0	1.416763	3.339022	4.620947	5.356591	6.201478
07:35:00	0	3.858877	7.342658	8.81118	9.639738	10.48305
07:40:00	0	3.93643	8.045514	9.699787	10.68023	11.77186
07:45:00	0	5	8.028878	9.305011	10.25525	10.98902
07:50:00	0	5	8.791534	10.32297	10.83242	11.51368
07:55:00	0	3.604446	7.04906	7.945823	8.45527	9.136526
08:00:00	0	7.09289	10.97833	11.83111	12.3658	13.38951

CHAPTER 4: DEVELOPMENT OF A DATA-CONVERSION AND ROUTE-CONFIGURATION MODULE

4.1 INTRODUCTION

This chapter develops the Data Conversion and Route Configuration Module, whose main functionalites include,

- Importing and converting non-traffic data from external sources and convert them into suitable formats for estimating travel-time reliability measures,
- Managing and storing converted data into the database developed in this chapter,
- Configuring travel-time reliability routes whose reliability measures will be estimated with traffic and external non-traffic data.

Figure 4.2.1 shows the overall architecture of the Travel-Time Reliability Measurement System (TTRMS) and the specific modules developed in this chapter. They include:

1) ***Database***, which stores all the data necessary to calculate travel-time reliability measures.

2) ***Admin/WZ Clients,*** which manage input and edit process of external data.

3) ***Reliability Services,*** which handles the requests from the Client to manage external data.

4) ***DB Access Layer,*** which manages the functions to access the database, such as INSERT, DELETE, GET, UPDATE and LIST.

5) ***DB Connection and Model,*** which defines the internal database structure and provides the database with connection to *DB Access Layer*.

6) ***External Data Reader,*** which imports weather and incident data.

7) ***Route and Route Config,*** which defines data structure and configuration method for travel-time routes.

8) ***Server and API server,*** which are accessed via HTTP.

The rest of this report describes the details of the above modules developed in this chapter.

4.2 DEVELOPMENT OF AN INTERNAL DATABASE FOR MANAGING AND STORING RELIABILITY-RELATED DATA

As noted in Figure 4.2.1, there are two types of external non-traffic data needed for estimating travel-time reliability measures. They are:

- Type 1: work zone, special event, winter-snow management data,
- Type 2: incident and weather data.

In this chapter, an internal database is first developed to store all the required data to estimate reliability measures. Next, the data-management modules necessary to import, convert and store external data are developed. The Type 2 data will be imported automatically by *External-Data Reader module*, while Type 1 data will be entered through the Client module by the system administrator, who also defines the travel-time routes for reliability estimation. Figure 4.2.2 shows the relational diagram of the database developed in this study and Table 4.2.1 includes the corresponding database tables. For example, '*tt* table' stores calculated travel-times for all the pre-defined routes, which are stored in '*ttr_route'* table. Each data row of '*tt* table' has the foreign keys pointing to external data, so that calculated travel-times and its associated external-data can be retrieved efficiently. The specific database schema for the database table is included in Table 4.2.2.

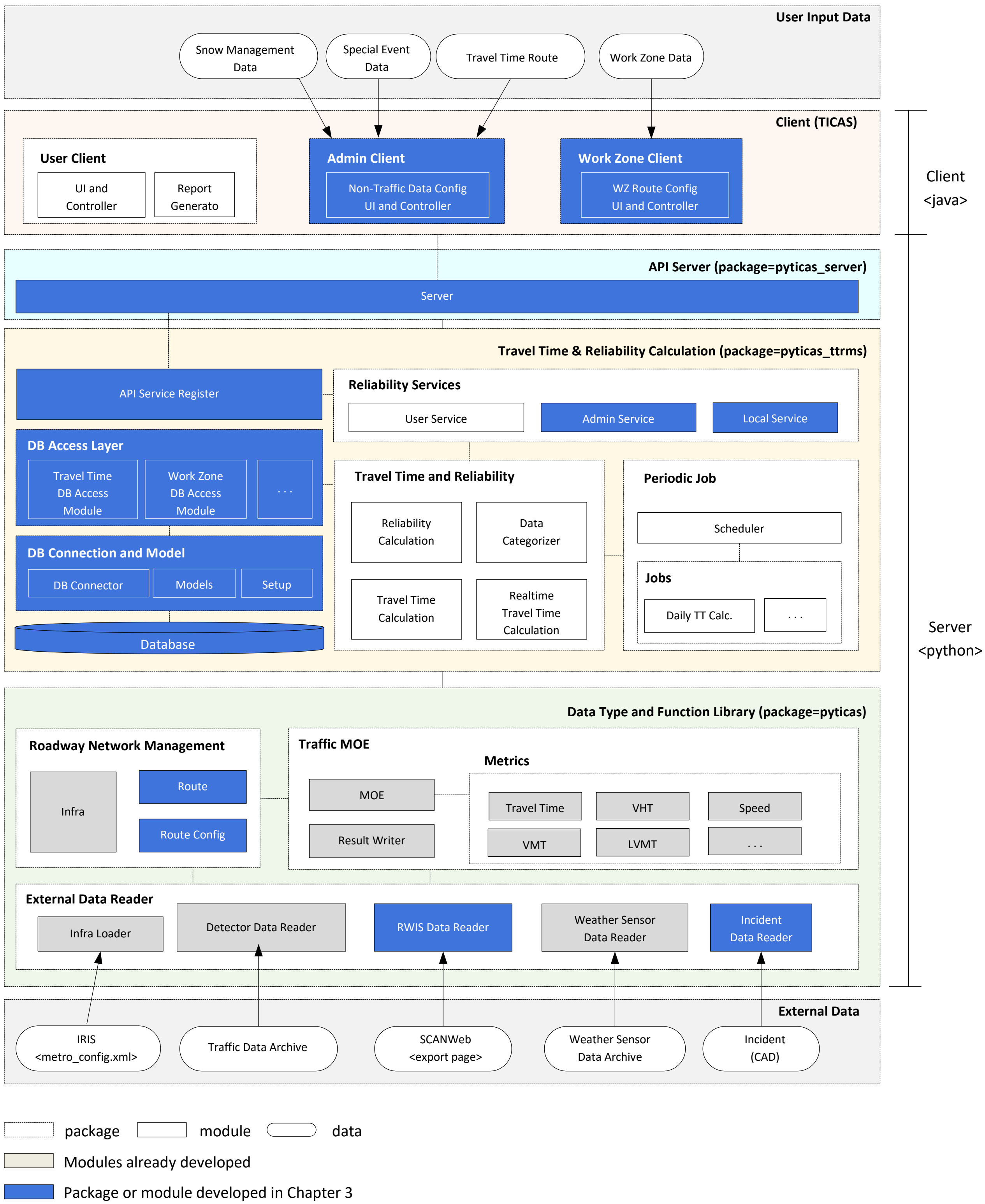


Figure 4.2.1 The New Modules developed in Chapter4 in TTRMS Architecture

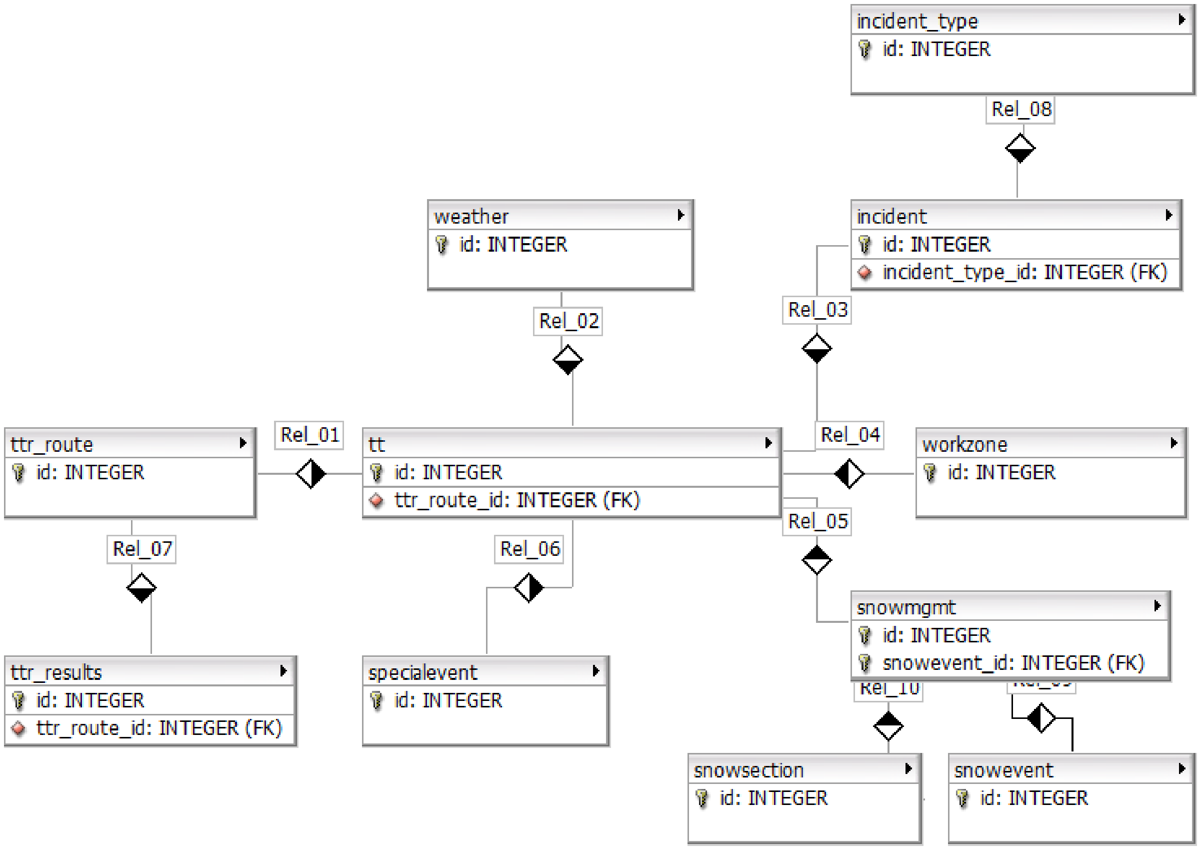


Figure 4.2.2 Relationship Diagram of Database

Table 4.2.1 Database Tables

Table Name	Data Fields
ttr_route	- name : route name - description : description - corridor : corridor name that the route is located, if the route goes through multiple corridors, it is comma separated lists of the corridor names - route : serialized route data (serialized *Route* object)
tt	- dtime : timestamp - speed : average speed - vmt : VMT - precip_type : precipitation type - precip_intensity : precipitation intensity - inc_type: incident type - inc_loc_type: incident location type (UP, DOWN and IN) - inc_loc: distance from the TTR route - wz_loc_type: location type of work zone - wz_loc: distance from the TTR route to the upstream of work zone - wz_lncfg: lane configuration (e.g. 2to1, 3to2) - wz_features: features (e.g., crossover, lane shifted and ramp construction) - wz_length: length of work zone
weather	- site_id: RWIS site id - sen_id: sensor id - time: timestamp - temp: surface temperature - air_temp: air temperature - visibility: visibility distance - wind_dir: wind direction - wind_speed: wind speed - precip_type: precipitation type (e.g., RAIN, SNOW, MIXED, SLEET, HAIL) - precip_intens: precipitation intensity (e.g., LIGHT, MODERATE, HEAVY, SLIGHT) - precip_amount: precipitation amount - surf_condition: surface condition (e.g., DRY, WET, FROST, SNOW_ICE_WATCH)
incident	- incident_type: incident type (e.g., CRASH, HAZARD) - cdts: created timestamp - udts: updated timestamp - xdts: closed timestamp - lat: latitude - lon: longitude
workzone	- name: work zone name - description: description - start_time: start time of duration - end_time: end time of duration - section1: lane configuration information for a direction (serialized *Route* object) - section2: lane configuration information for the other direction (serialized *Route* object)
snowmgmt	- lane_lost_time : lane lost time according to snow management report - lane_regain_time : lane regain time according to snow management report - duration: lane lost duration in hours
snowevent	- start_time: snow start time - end_time: snow end time

snowsection	- name: name of snow management section - prj_id: project id of snow management section - description: description - section: station and ramp information of the section (serialized *Route* object)
specialevent	- name: event name - description: description - start_time: start time of the event - end_time: end time of the event - lat: latitude of the location - lon: longitude of the location - attendance: number of attendance
ttr_result	- ttr_type: reliability index type (e.g., buffer index, planning index) - reliability: calculated reliability index - start_time: start time of the target duration - end_time: end time of the target duration

Table 4.2.2 Database Schema

Table Name : ttr_route

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(255)		NN				
description	TEXT						
corridor	VARCHAR(20)						
ttr_route	LONGTEXT						
reg_date	DATETIME						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Table Name : tt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
ttr_route_id	INTEGER		NN	UNSIGNED			
specialevent_id	INTEGER		NN	UNSIGNED			

weather_id	INTEGER	NN	UNSIGNE D
workzone_id	INTEGER	NN	UNSIGNE D
snowmgmt_snowevent_i d	INTEGER	NN	UNSIGNE D
snowmgmt_id	INTEGER	NN	UNSIGNE D
incident_id	INTEGER	NN	UNSIGNE D
dtime	DATETIME	NN	
tt	FLOAT	NN	
speed	FLOAT	NN	
vmt	FLOAT	NN	
precip_type	INTEGER	NN	UNSIGNE D
precip_intensity	INTEGER		UNSIGNE D
inc_impact	INTEGER		UNSIGNE D
inc_type	INTEGER	NN	UNSIGNE D
inc_loc	INTEGER		UNSIGNE D
wz_loc_type	INTEGER		UNSIGNE D
wz_loc	FLOAT		
wz_lncfg	VARCHAR(5)		
wz_features	VARCHAR(100)		
wz_length	FLOAT		

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id
tt_FKIndex2	Index	incident_id
tt_FKIndex4	Index	snowmgmt_id snowmgmt_snowevent_id
tt_FKIndex5	Index	workzone_id
tt_FKIndex6	Index	weather_id

tt_FKIndex6	Index	specialevent_id
tt_FKIndex6	Index	ttr_route_id

Table Name : incident

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
incident_type_id	INTEGER		NN	UNSIGNED			
cdts	DATETIME		NN				
udts	DATETIME						
xdts	DATETIME						
lat	FLOAT		NN				
lon	FLOAT		NN				
xstreet1	VARCHAR(50)						
xstreet2	VARCHAR(50)						
efeatyp	VARCHAR(10)						
openevent	BOOL						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id
incident_FKIndex1	Index	incident_type_id

Table Name : snowevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
start_time	DATETIME		NN				
end_time	DATETIME		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Table Name : snowmgmt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
snowevent_id	**INTEGER**	PK	NN	UNSIGNED			
snowsection_id	INTEGER		NN	UNSIGNED			
lane_lost_time	DATETIME						
lane_regain_time	DATETIME						
duration	FLOAT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id snowevent_id
snow_mgmt_FKIndex1	Index	snowevent_id
snowmgmt_FKIndex2	Index	snowsection_id

Table Name : snowsection

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(20)						
prj_id	VARCHAR(20)						
description	TEXT						
section	TEXT						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id
snowroute_unique	Index	name

Table Name : specialevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(100)		NN				
description	TEXT						
start_time	DATETIME		NN				
end_time	DATETIME		NN				
lat	FLOAT						

lon	FLOAT						
attendance	INTEGER			UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Table Name : ttr_results

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
ttr_route_id	INTEGER		NN	UNSIGNED			
ttr_type	INTEGER			UNSIGNED			
reliability	INTEGER		NN	UNSIGNED			
start_time	DATETIME						
end_time	DATETIME						

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id
ttr_results_FKIndex1	Index	ttr_route_id

Table Name : weather

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
site_id	INTEGER			UNSIGNED			
sen_id	INTEGER			UNSIGNED			
time	DATETIME		NN				
temp	FLOAT						
air_temp	FLOAT						
visibility	FLOAT						
wind_dir	INTEGER			UNSIGNED			
wind_speed	FLOAT						
precip_type	INTEGER			UNSIGNED			
precip_intens	INTEGER			UNSIGNED			
precip_amount	FLOAT						
surf_condition	INTEGER			UNSIGNED			

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id

Table Name : workzone

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
name	VARCHAR(255)		NN				
description	TEXT		NN				
years	VARCHAR(255)		NN				
start_time	DATETIME		NN				
end_time	DATETIME		NN				
corridors	VARCHAR(255)		NN				
section1	TEXT		NN				
section2	TEXT		NN				
reg_date	DATETIME		NN				

IndexName	IndexType	Columns
PRIMARY	PRIMARY	id
workzone_unique	Unique Index	name

Definition of Classes for Data Types

Figure 4.2.3 shows the data types representing 'reliability routes' and non-traffic, external data used in TTRMS. To implement the database schema shown in Table 4.2.2, a set of classes are defined using Object Relational Mapper (ORM) for all the database tables in Table 4.2.1. I.e., a database table is represented by a class, which is used by *DB Access Layer* to access the database. Figure 4.2.4 shows an example database-table class defined for special-event data using ORM, which provides abstractions of data access and portability across different database systems.

Figures 4.2.5 and 4.2.6 show the example definitions of Python and Java data-classes for special-event data. It can be noted that the names of the member variables in both classes are same as with those in the data-table classes. In TTRMS, Python data-classes are used by other modules in the server, while data-table classes are only used to access the database. Further, Java data-classes are specifically used to manage external, non-traffic data in the client. For the Python Server to communicate with the Java Client, each instance of Python data-classes is serialized to JSON string by using the ***json*** module of Python library, while any instance in Java data-Classes is serialized to JSON string with ***gson***, an open-source serialization library, and converted to a Java object.

In the system setup phase, the *SetUp* module in the *DB Connection and Data model* package creates all database tables by calling *create* functions in the ORM library.

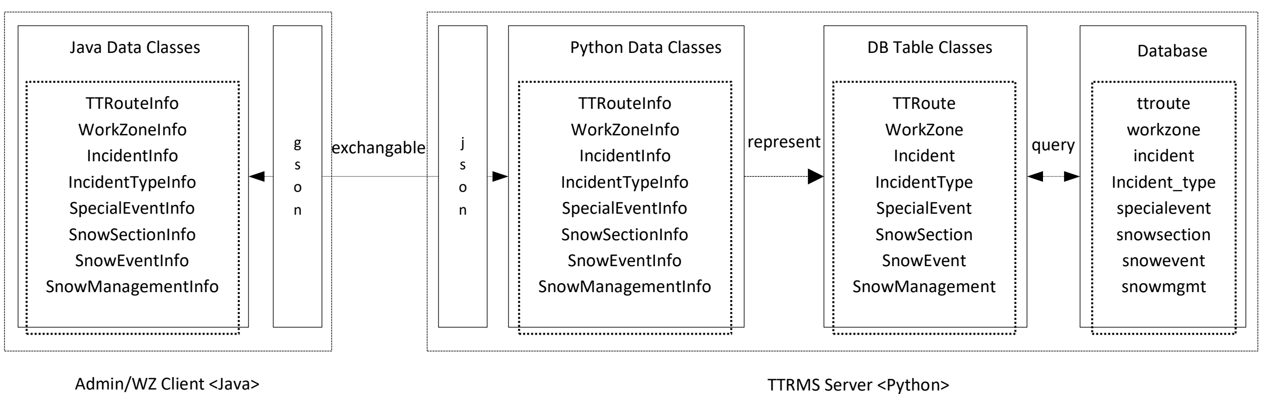


Figure 4.2.3 Types of External Data in Client and Server

```
# Database Table Class
class Specialevent(Base):
   __tablename__ = 'specialevent'
   id = Column(Integer, primary_key=True)
   name = Column(String(100), nullable=False)
   description = Column(Text, nullable=True)
   start_time = Column(DateTime, nullable=False)
   end_time = Column(DateTime, nullable=False)
   lat = Column(Float, nullable=False)
   lon = Column(Float, nullable=False)
   attendance = Column(Integer, nullable=True)
   reg_date = Column(DateTime, nullable=False, default=datetime.datetime.now)
```

Figure 4.2.4 Sample Database-Table Class for Special Event Data

```
# Python Data Class
class SpecialEventInfo(InfoBase):

   _info_type_ = 'special event'
   _dt_attrs_ = ['start_time', 'end_time']

   def __init__(self):
      self.id = None
      self.name = None
      self.description = None
      self.years = None
      self.start_time = None
      self.end_time = None
      self.lat = None
      self.lon = None
      self.attendance = None

   def set_years(self):
      start_time = datetime.datetime.strptime(self.start_time, '%Y-%m-%d %H:%M:%S')
      end_time = datetime.datetime.strptime(self.end_time, '%Y-%m-%d %H:%M:%S')
      self.years = self.years_string(start_time.year, end_time.year)
```

Figure 4.2.5 Sample Python Data-Class for Special Event Data

```
// Java Data Class
package edu.umn.natsrl.ttrms.types;

public class SpecialEventInfo extends InfoBase {
   public String name;
   public String description;
   public String start_time;
   public String end_time;
   public Double lat;
   public Double lon;
   public Integer attendance;
   public String years;

   public SpecialEventInfo() {
      this.setTypeInfo(TTRMSConfig.INFO_TYPE_SPECIAL_EVNET);
   }
   // implementations of the below functions are omitted
   private Calendar toCalendar(String dts) { … }
   public String getDuration() { … }
   public void setDuration(Date sdt, Date edt) { … }
   private String DateToString(Date dt) {… }
   public Date getEndDate() {… }
   public Date getStartDate() {… }

   @Override
   public String toString() {… }

   @Override
   public SpecialEventInfo clone() {… }
```

Figure 4.2.6 Sample Java Data Class for Special Event

4.3 DEVELOPMENT OF CONVERSION MODULES FOR EXTERNAL NON-TRAFFIC DATA

4.3.1 Design of Data-Conversion Process

Figure 4.3.1 shows the overall process and the modules developed in this section to import non-traffic external data and convert/store them to the data types suitable for estimating travel-time reliability measures. The major features of the data conversion and storing process are as follows:

- Snow-management, special-event and work-zone data are entered and edited through the user-interfaces (UI) in *Admin/WZ Client,* which uses the data management services of the API server, such as *Data Insert, Update, Fetch, Delete and List,* developed for each data-type.
- *DB Access Layer* provides *API Server* and *External Data Reader* with the functions to access the external data stored in the database.
- The weather and incident data imported by *External Data Reader package* are stored in the database through *DB Access Layer*

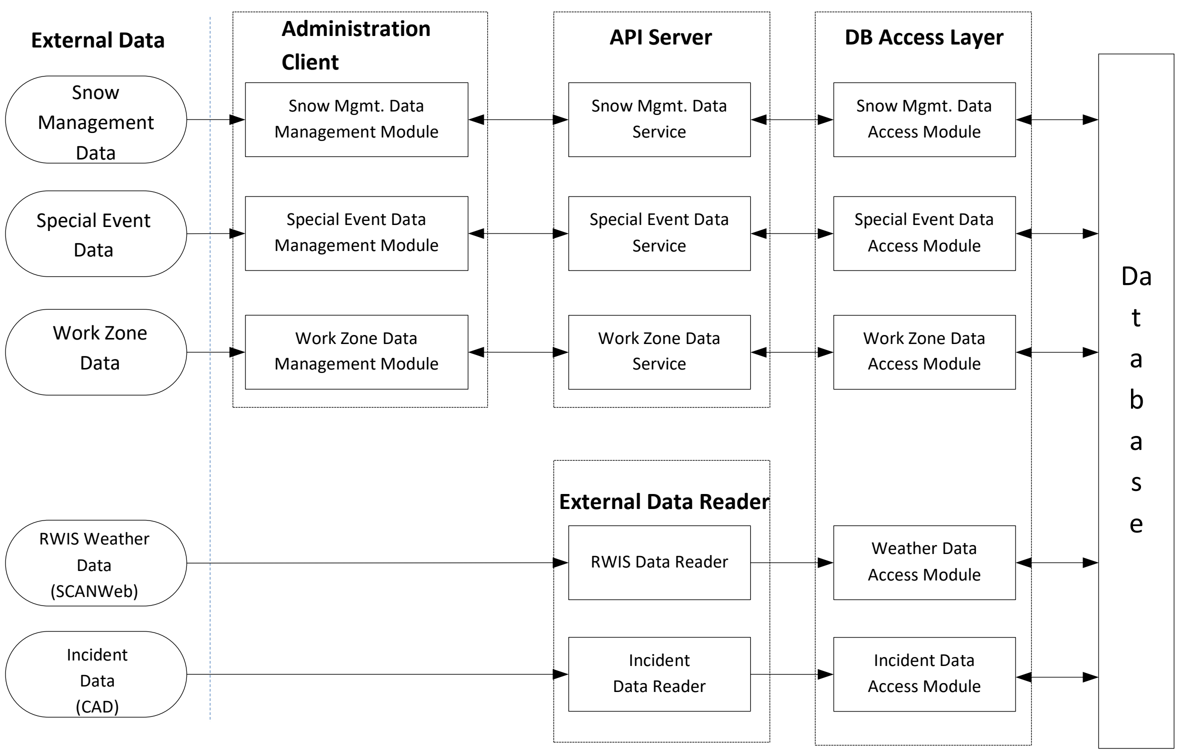


Figure 4.3.1 Overview of Data-Conversion Process for External Data

Figure 4.3.2 shows the modules and interfaces developed to convert the Type 1 data, which are imported through the *Administration Client* module. The main features of this process are as follows:

1) Data service modules in *API Server* provide the *Client* with the interfaces, such as LIST, INSERT, GET, UPDATE and DELETE. Each interface has the entry point specified as HTTP URL.

- *For example, Work Zone Data Service* module has the entry point of *"/ttrms/admin/workzone/list"* and the *Client* obtains work-zone information by accessing the URL in HTTP GET method.
- Similarly, *"/ttrms/admin/workzone/insert"* is the entry point for INSERT, which is accessible through HTTP POST for work-zone information.

2) *DB Access* Modules in *Data Access Layer* have also the interfaces of LIST, INSERT, GET, UPDATE and DELETE, which are accessed by calling specific functions. For example*, Work Zone DB Access Module* has the functions named as *insert(), list(), get(), update() and delete(),* which are called by *Work Zone Data Service* module as needed.

3) *DB Access* Modules request query to the database via 3rd party SQL library.

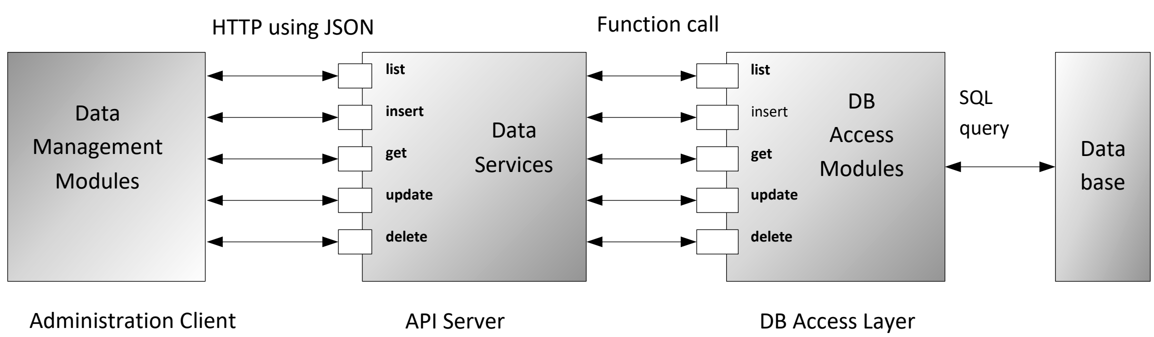


Figure 4.3.2 Data-Conversion Modules for Type 1 External Data

The general sequence of the conversion process for Type 1 data is as follows:

1) The external data and parameters entered by an administrator are converted to JSON (JavaScript Object Notation) string and packed to the body of HTTP POST request.

- HTTP GET method is used if additional data are not required, e.g., *listing*.
- An example JSON string that represents a snow-event is shown in Figure 4.3.3.

```
{
   "id" : 1,
   "start_time" : "2016-02-12 05:00:00",
   "end_time" : "2016-02-12 12:00:00",
}
```

Figure 4.3.3 An Example JSON String

2) The packed, requested data is sent to the entry point of the *API server.*

3) *Data Service* Module converts received JSON data to corresponding data-object, which is used in the server, and calls corresponding function of *DB Access* Module*.*

4) The Client receives the data from the *Data Service* Module*.*

- When the *Client* requests Delete or Update service and no data is returned, the service-process status code is returned.

5) The converted data is updated and shown in the User Interface by the *Client*.

Figure 4.3.4 illustrates an example data-conversion sequence to add a new data:

1) User enters the data through the User Interface of the Client program.

2) The entered data are converted to JSON object and serialized to string.

3) The serialized data are sent to the entry point of insertion in the API Server.

4) Data Service Module converts the serialized string to the corresponding data object.

5) Converted data object are passed to INSERT function of DB Access Module.

6) DB Access Module inserts data to the database.

7-8) Data are returned to Data Service Module.

9-10) Returned data and status code are converted to JSON string and sent to the Client.

11) The Client receives the data and converts to the data object used in the Client.

12) User Interface of the Client is updated.

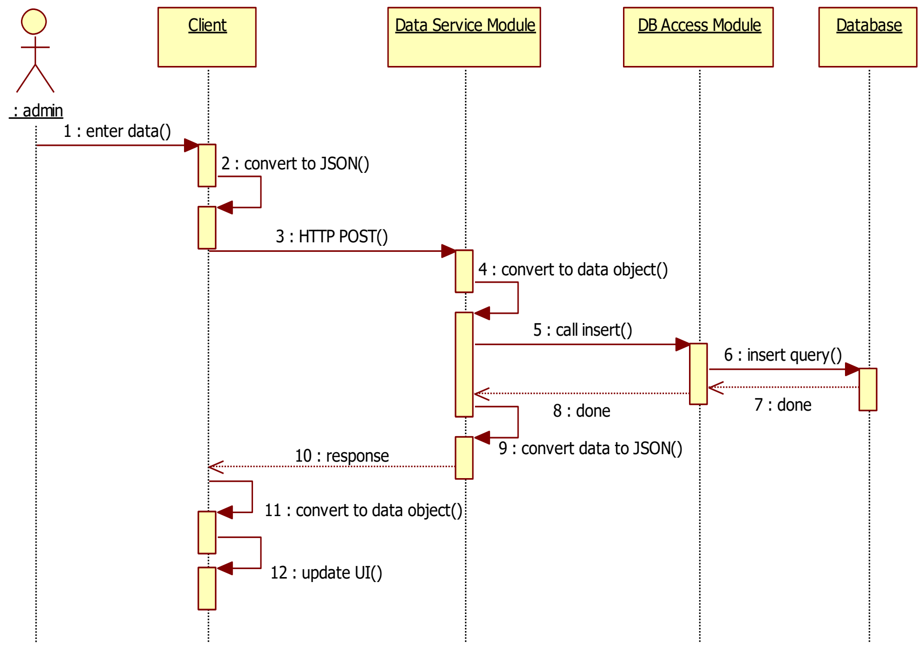


Figure 4.3.4 An Example Sequence Diagram to Add a New Data

Figure 4.3.5 shows the specific modules and interfaces developed to convert the Type 2 data, i.e., weather and incident data. As noted earlier, the Type 2 data are directly imported through the *External-Data Reader* module. The conversion process for the Type 2 data can be summarized as follows:

1) *Data Pre-fetch* module calls the data-reading function in *Data Reader.*

2) *Data Reader* checks the cached data in *Local Disk.*

- The cached data are returned if they exist.

3) *Data Reader* module sends a request to the external data server if cached data are not found.

- The received data are stored to *Local Disk* for future access.

4) The requested data are saved in the *Database* by *Data Pre-fetch* module through *DB Access Modules.*

**The Incident Data Access module in the DB Access Layer has been developed in this chapter, while the Incident Data Reader will be developed when the API server for CAD data is available.*

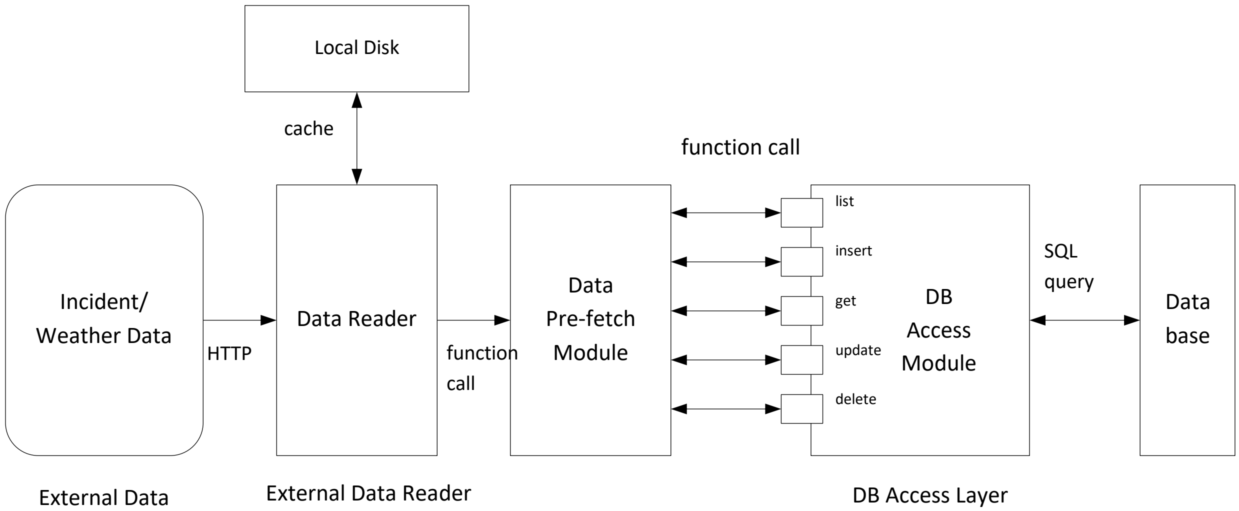


Figure 4.3.5 Data-Conversion Process and Modules for Type 2 Data

4.3.2 Development of Data-Access Layer

Figure 4.3.6 shows the internal structure and individual modules of the *Data-Access Layer, which* contains *Data-Access Modules* for different types of external data. The *Data-Access Modules* are designed to delegate the database-access functions to the *Data-Access-Base* module. If additional functions are needed for specific types of data, those functions can be implemented in corresponding *Data Access Modules*. Figures 4.3.7 and 4.3.8 include the source codes for the *Special Event Data Access* and the *Data-Access-Base* modules.

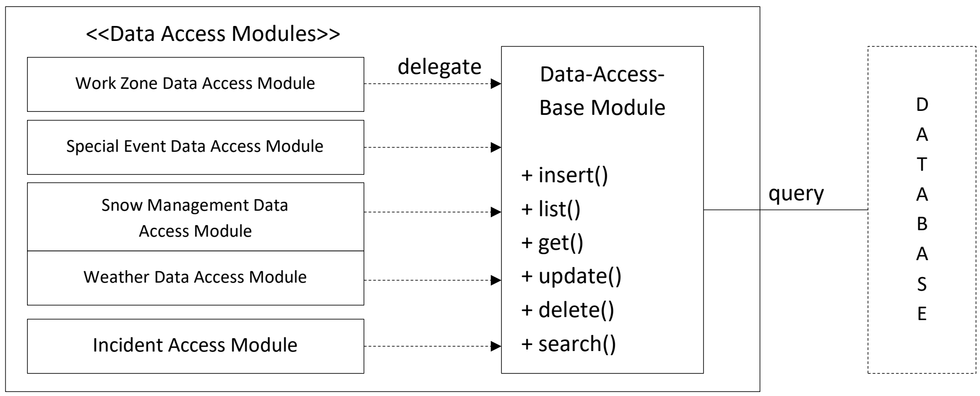


Figure 4.3.6 Structure of Data-Access Layer

Testing Data-Access Modules

The database-access functions in the above Database-Access modules, i.e., *Insert*, *Update*, *List* and *Delete*, were tested using a test program with a set of hypothetical incident data. The following procedure was used for this testing:

1) Create and Insert incident types

2) Get 'List' of incident types

3) Update incident types

4) Create and Insert incident data (incident has dependency to incident type)

5) Get 'List' incident data

6) Update incident data

7) Delete incident data and Get 'List' to check if data are deleted

8) Delete incident types and Get 'List' to check if incident types are deleted

Figure 4.3.9 shows the results of the test program with the above procedure. As shown in this figure, each function performed correctly as expected.

```
class DataAccess(object):

  def __init__(self, **kwargs):
      self.da_base = DataAccessBase(model.Specialevent, SpecialEventInfo, **kwargs)
   def exist(self, name, start_time, end_time):
      if isinstance(start_time, str):
         start_time = datetime.datetime.strptime(start_time, '%Y-%m-%d %H:%M:%S')
      if isinstance(end_time, str):
         end_time = datetime.datetime.strptime(end_time, '%Y-%m-%d %H:%M:%S')
      exs = self.da_base.search([('name', name), ('start_time', start_time), ('end_time', end_time)], op='and',
                        cond='match')
      return exs
   def list(self):
      return self.da_base.list()
   def list_by_year(self, years):
      wheres = [('years', y) for y in years]
      return self.da_base.search(wheres, op='or', cond='like')
   def years(self):
      ys = []
      for sei in self.da_base.list():
         for y in sei.years.split(','):
```

```
            iy = int(y)
            if iy not in ys:
                ys.append(iy)
        return sorted(ys)
    def get_by_id(self, se_id):
        return self.da_base.get_data_by_id(se_id)
    def delete(self, se_id, autocommit=False):
        return self.da_base.delete(se_id, autocommit=autocommit)
    def insert(self, sei, autocommit=False):
        return self.da_base.insert(sei, autocommit=autocommit)
    def update(self, se_id, field_data, autocommit=False):
        return self.da_base.update(se_id, field_data, autocommit=autocommit)
    def rollback(self):
        self.da_base.session.rollback()
    def commit(self):
        self.da_base.commit()
    def close(self):
        self.da_base.close()
```

Figure 4.3.7 Data-Access Module for Special Event Data

```
class DataAccessBase(object):
    def __init__(self, dbModel, dataInfoType, **kwargs):
        """
        :param dbModel: DB model defined in `pyticas_ttrms.db.model`
        :param dataInfoType:  corresponding class to DB model defined in `pyticas_ttrms.ttrms_types`
        """
        self.dbModel = dbModel
        self.dataInfoType = dataInfoType
        self.dt_attrs = dataInfoType._dt_attrs_ if hasattr(dataInfoType, '_dt_attrs_') else []
        self.route_attrs = dataInfoType._route_attrs_ if hasattr(dataInfoType, '_route_attrs_') else []
        self.rel_attrs = dataInfoType._rel_attrs_ if hasattr(dataInfoType, '_rel_attrs_') else {}
        self.session = kwargs.get('session', conn.get_session())
        """:type: sqlalchemy.orm.Session """
    # implementations of the below function are omitted
    def get_model_by_name(self, name):
    def get_model_by_id(self, id):
    def get_data_by_name(self, name):
    def get_data_by_id(self, id):
    def insert(self, data, autocommit=False):
    def list(self, **kwargs):
    def delete(self, id, autocommit=False):
    def update(self, id, field_data, autocommit=False):
```

```
def search(self, searches, op='and', cond='match', **kwargs):
def search_date_range(self, sinfo, einfo):
def to_info(self, model_data, data_info_type = None):
def to_model(self, info_data, data_info_type = None):
def commit(self):
def close(self):
```

Figure 4.3.8 Data-Access-Base module

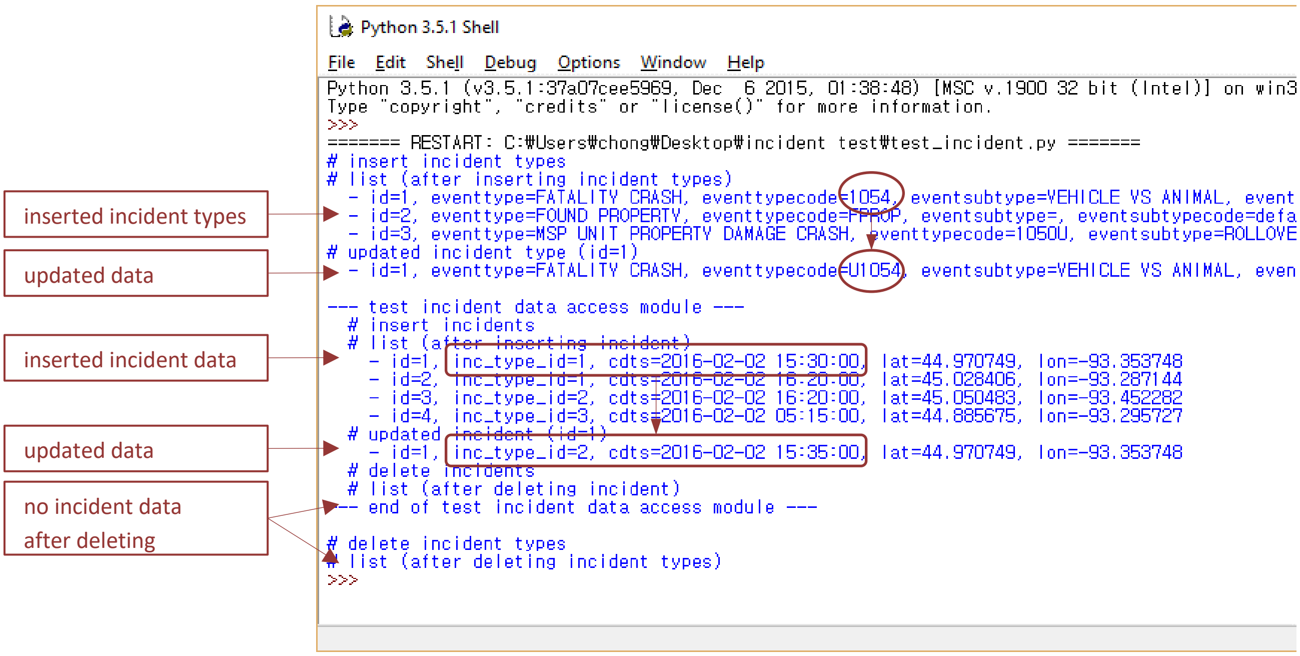


Figure 4.3.9 DB Access-Module Test Results with an Incident Data

4.3.3 Development of API Server

Figure 4.3.10 shows the internal structure of the API server, which is consisted with *Request Dispatcher, Data Services* and *TeTRES-API* modules. The functions handling data-service requests are built into the TeTRES-API module. In the beginning of the system operation, the *Data Services* are registered to the *Request Dispatcher*. When the *Client* requests data services, *Request Dispatcher* runs the corresponding services. Each *Data Service* delegates the data requests to the *TeTRES-Api* module, while each *Data Service* module can have additional functions to handle different types of data requests. Figure 4.3.11 shows the source code for *Special Event Data Service* module. The source code for *TeTRES-API* module is included in Figure 4.3.12.

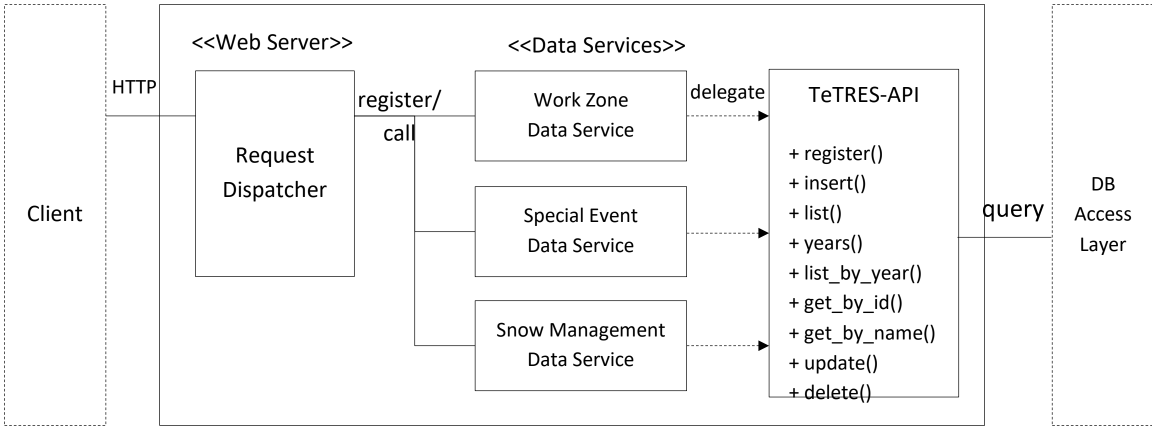


Figure 4.3.10 Structure of API Server and Data Service Modules

```
def register_api(app):
    TTRMSApi(app, 'specialevent', json2sei, SpecialEventDataAccess, {
        'insert': (api_urls.SE_INSERT, ['POST']),
        'list': (api_urls.SE_LIST, ['GET']),
        'list_by_year': (api_urls.SE_LIST_BY_YEAR, ['POST']),
        'get_by_id': (api_urls.SE_GET, ['POST']),
        'update': (api_urls.SE_UPDATE, ['POST']),
        'delete': (api_urls.SE_DELETE, ['POST']),
        'years': (api_urls.SE_YEARS, ['GET']),
    }).register()
```

Figure 4.3.11 Special-Event Data-Service module

```
class TTRMSApi(object):

    def __init__(self, app, name, json2obj, da_class, uris):
        self.app = app
        self.name = name
        self.json2obj = json2obj
        self.data_access_class = da_class
        self.uris = uris

    def register(self):
        autodoc = get_autodoc()
        for fname, (uri, methods) in self.uris.items():
            if not hasattr(self, fname):
                continue
            self.app.add_url_rule(uri,
                                  'ttrms_%s_%s' % (self.name, fname),
                                  getattr(self, fname),
                                  methods=methods)

    # implementations of the below functions are omitted
    def insert(self):
    def insert_all(self):
    def list(self):
    def list_by_year(self):
    def get_by_id(self):
    def get_by_name(self):
    def update(self):
    def delete(self):
    def years(self):
```

Figure 4.3.12 TeTRES-API Module

4.3.4 Development of External-Data Reader

The External-Data Reader package includes the data-importing modules for both RWIS and Incident data. Figure 4.3.13 shows the structure of the External-Data Reader developed to import the RWIS Data. The *RWIS-Data Reader* module provides RWIS site information and the functions to get weather information. The *Web Page Parser* module reads the export page of the SCANWeb site with user-specified parameters, such as time duration and RWIS site, and extracts weather information from HTML, while the *RWIS Data Cache* module is used to save weather information to local disk for a quick access. Figure 4.3.14 includes the some of the source code for the RWIS Data Reader module.

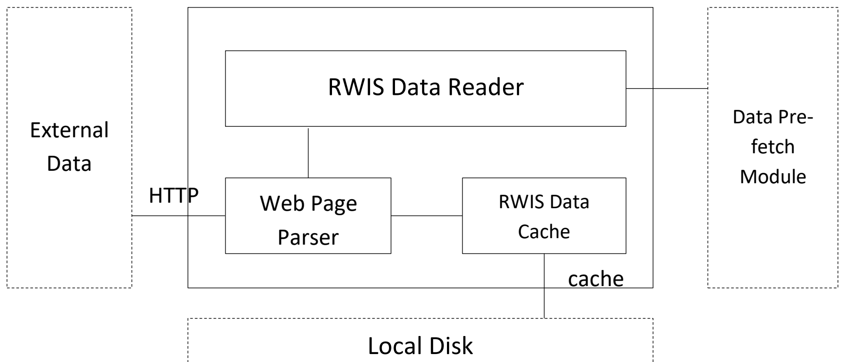


Figure 4.3.13 Structure of RWIS Data Reader

```
import datetime, enum

from pyticas import cfg, logger
from pyticas.dr.rwis_reader import scanweb_export, scanweb_html
from pyticas.tool import distance
from pyticas.tool.cache import lru_cache
from pyticas.ttypes import RWISData, RWISSiteInfo, Period

logging = logger.getDefaultLogger(__name__)

class RWIS_READER(enum.Enum):
   SCANWEB_EXPORT = scanweb_export
   SCANWEB_HTML = scanweb_html

# implementations of the below functions are omitted
def find_nearby_sites(s_lat, s_lon):
def get_site_by_id(site_id):
def get_all_rwis_sites():
def get_weather_by_site(group_id, site_id, prd, **kwargs):
def get_weather(site, prd, **kwargs):
```

Figure 4.3.14 RWIS Data Reader module

Testing RWIS Data-Reader module

The RWIS-Data Rader module was tested by importing the RWIS data for a detector station and comparing the imported data with those from the SCANWeb site. In this testing, the RWIS-Data Reader module was applied to import the RWIS data for the detector station S73 on I-35W. Specifically, the following functions of the RWIS-Data Rader module were tested:

- *find_nearby_sites(lat, lon)* : find the RWIS site information close to a given coordinate

- *get_weather(site, time_period)* : read weather data from SCANWeb site for a given RWIS site and time period

Figure 4.3.15 shows the location of the RWIS site, I-35@ Minnesota River, found by the RWIS-Data Reader module for the test station S73. The Figure 4.3.16 includes the RWIS data imported from this site, while Figures 4.3.17 and 4.3.18 show the data from the SCANWeb site for the same RWIS site. As can be noted from these figures, the imported data from the RWIS-Data Reader module exactly match those from the SCANWeb site for the same RWIS site.

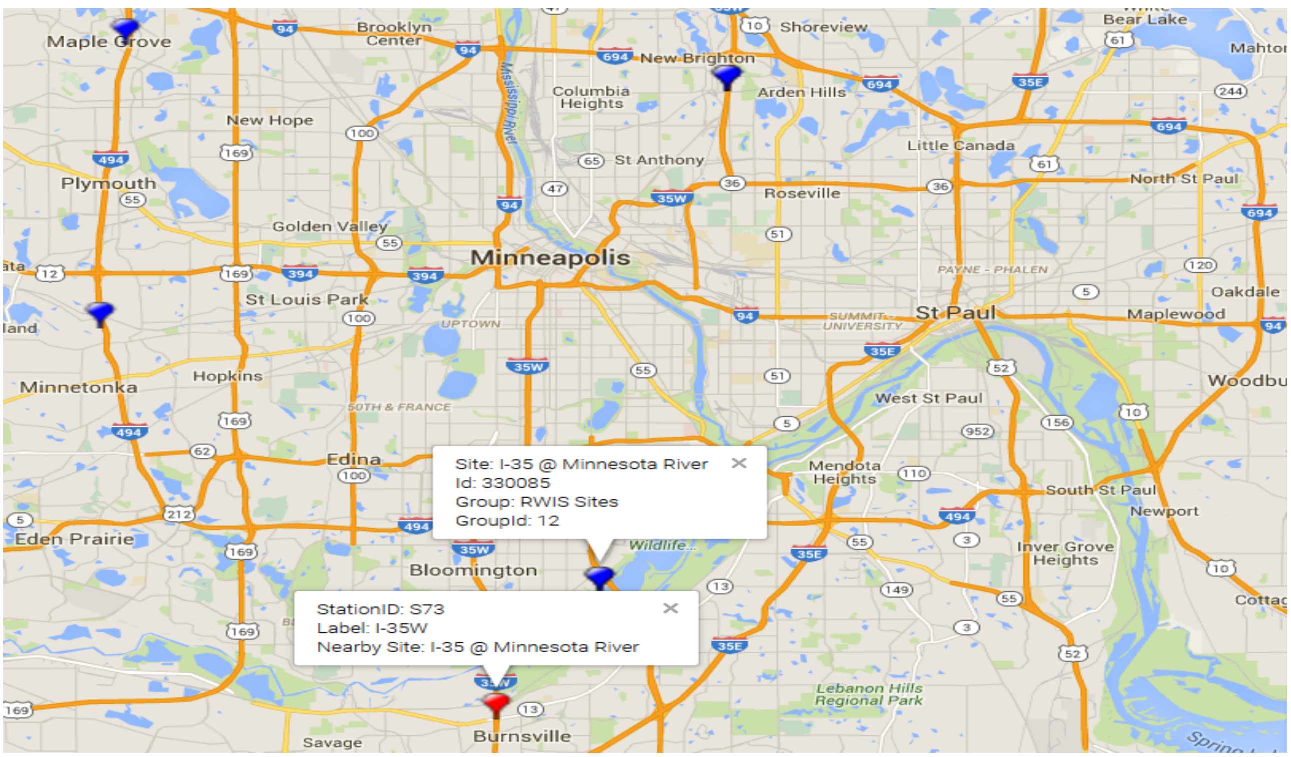


Figure 4.3.15 Detector station S73 and nearby RWIS site locations

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec  6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>

=========== RESTART: test_rwis.py =====================
```

station=**S73**, label=I-35W, lat=44.774990, lon=-93.288440

site name=**I-35 @ Minnesota River**, site id=**330085**, distance to station=**4.414634** mile

Weather Data

- **DateTime** ['03/15/2016 07:05', '03/15/2016 07:10', '03/15/2016 07:15', '03/15/2016 07:20', '03/15/2016 07:25', '03/15/2016 07:30', '03/15/2016 07:35', '03/15/2016 07:40', '03/15/2016 07:45', '03/15/2016 07:50', '03/15/2016 07:55', '03/15/2016 08:00', '03/15/2016 08:05', '03/15/2016 08:10', **'03/15/2016 08:15', '03/15/2016 08:20', '03/15/2016 08:25', '03/15/2016 08:30', '03/15/2016 08:35', '03/15/2016 08:40', '03/15/2016 08:45', '03/15/2016 08:50', '03/15/2016 08:55'**]

- **SfStatus** ['Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', **'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture', 'Trace Moisture'**]

- **SfTemp** [55.6, 55.9, 55.8, 55.8, 55.9, 55.9, 55.8, 55.6, 55.6, 55.4, 55, 55.2, 54.9, 54.9, 54.7, 54.9, 54.9, **54.7, 54.5, 54.3, 54.5, 54.3, 54.1**]

- **PrecipRate** [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, **0, 0, 0, 0, 0.3, 0.5, 0.3, 0.4, 0.2**]

- **PrecipType** ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', None, None, None, None, None, None, None, 'Yes', **'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes'**]

Figure 4.3.16 Output Screen of Test Program

Metro Sites
I-35 @ Minnesota River(330085)
Precipitation History
03/14/2016 09:00 to 3/15/2016 09:00

Surface/Precip. History | Surface/Atmos. History | Surface History Graph | Atmospheric History | Export

Change End Date: 3/15/2016 09:00 Go <-- -->

Date/Time (CDT)	Type	Intens	Rate	Start Time	End Time	10 min. Accum	1 hr. Accum	3 hr. Accum	6 hr. Accum	12 hr. Accum	24 hr. Accum
03/15/2016 08:55	Yes	-	0.2 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:50	Yes	-	0.4 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:45	Yes	-	0.3 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:40	Yes	-	0.5 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:35	Yes	-	0.3 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:30	Yes	-	0.0 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:25	Yes	-	0.0 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:20	Yes	-	0.0 iph	03/15/2016 08:19	-	-	-	-	-	-	-
03/15/2016 08:15	Yes	-	0.0 iph	03/15/2016 08:12	-	-	-	-	-	-	-
03/15/2016 08:10	None	-	0.0 iph	03/15/2016 07:34	03/15/2016 07:36	-	-	-	-	-	-
03/15/2016 08:05	None	-	0.0 iph	03/15/2016 07:34	03/15/2016 07:36	-	-	-	-	-	-
03/15/2016				03/15/2016	03/15/2016						

Figure 4.3.17 Precipitation History Page in SCANWeb for the Same Duration with Test Code

Metro Sites
I-35 @ Minnesota River - 35W Minnesota River (330085.0)
Surface/Atmospheric History
03/14/2016 09:00 to 3/15/2016 09:00

Surface/Precip. History | Surface History Graph | Atmospheric History | Precip. History | Export

Change End Date: 3/15/2016 09:00 Go <-- -->

Date/Time (CDT)	Surface				Air			Wind			Precip	
	Status	Sfc	Pvt	Sub	Air	RH	Dew	Avg	Gust	Dir	Type	Intens
03/15/2016 08:55	Trace Moisture	54.1F	-	50F	50F	97%	52F	13 mph	17 mph	E	Yes	-
03/15/2016 08:50	Trace Moisture	54.3F	-	50F	50F	96%	52F	9 mph	16 mph	E	Yes	-
03/15/2016 08:45	Trace Moisture	54.5F	-	50F	50F	96%	52F	9 mph	20 mph	SE	Yes	-
03/15/2016 08:40	Trace Moisture	54.3F	-	50F	50F	96%	52F	13 mph	20 mph	E	Yes	-
03/15/2016 08:35	Trace Moisture	54.5F	-	50F	50F	96%	52F	9 mph	16 mph	SE	Yes	-
03/15/2016 08:30	Trace Moisture	54.7F	-	50F	50F	96%	52F	6 mph	14 mph	SE	Yes	-
03/15/2016 08:25	Trace Moisture	54.9F	-	50F	50F	96%	52F	7 mph	12 mph	SE	Yes	-
03/15/2016 08:20	Trace Moisture	54.9F	-	50F	50F	96%	52F	Calm	8 mph	E	Yes	-
03/15/2016 08:15	Trace Moisture	54.7F	-	50F	50F	96%	52F	4 mph	6 mph	E	Yes	-
03/15/2016 08:10	Trace Moisture	54.9F	-	50F	50F	95%	52F	Calm	6 mph	NE	None	-
03/15/2016 08:05	Trace Moisture	54.9F	-	50F	50F	95%	52F	Calm	6 mph	SE	None	-
03/15/2016 08:00	Trace Moisture	55.2F	-	50F	50F	95%	52F	4 mph	6 mph	SE	None	-
03/15/2016 07:55	Trace Moisture	55.0F	-	50F	50F	95%	52F	Calm	4 mph	E	None	-
03/15/2016 07:50	Trace Moisture	55.4F	-	50F	50F	95%	52F	Calm	4 mph	E	None	-
03/15/2016	Trace	55.6F	-	50F	50F	95%	52F	Calm	4 mph	SE	None	-

Figure 4.3.18 Surface history page for the same duration with test code in SCANWeb site

4.4 DEVELOPMENT OF ADMINISTRATION-CLIENT MODULE

Figure 4.4.1 shows the structure of the Administration-Client module developed for each of Type 1 data, i.e., work zone, special event and snow management data. The main features of the internal process shown in this figure can be summarized as follows:

- The DataAPI module extends the APIClient module, whose functions are inherited by the DataAPI module. The interfaces to access the external data in the server are built into the APIClient module, while the *HttpClient* module has the functions to send the request of HTTP GET and POST to server.

- *The DataAPI* module sends the request from *User Interface* to server through the *HttpClient* module and triggers the corresponding events after receiving a response from the server.

- The Event handler updating status in *User Interface* module is executed by the events from the *DataAPI* module.

Figures 4.4.2 and 4.4.3 include the source codes of the *DataAPI* module for special-event data and the *APIClient* module. The rest of this section describes the user-interfaces developed in this task for different types of Type 1 data.

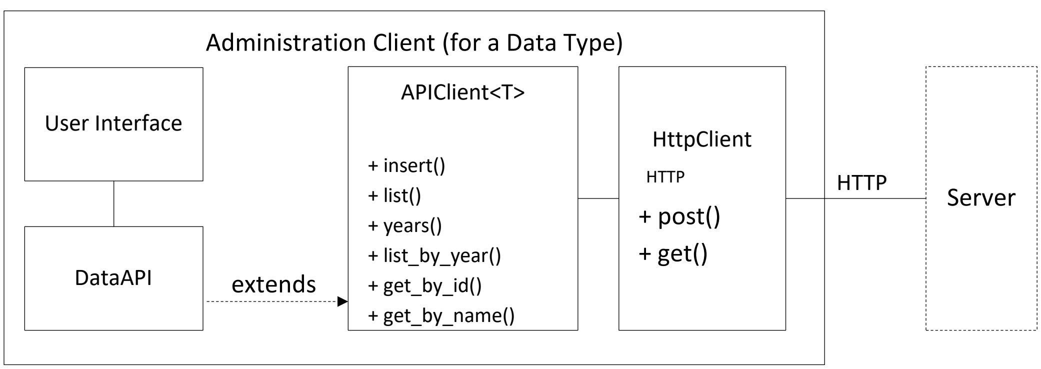


Figure 4.4.1 Structure of Administration-Client Module for Each Data Type

```
public class SpecialEvent extends APIClient<SpecialEventInfo> {

    public SpecialEvent() {
        this.RESPONSE_LIST_TYPE = new TypeToken<ListResponse<SpecialEventInfo>>(){}.getType();
        this.RESPONSE_TYPE = new TypeToken<ObjectResponse<SpecialEventInfo>>(){}.getType();
        this.DATA_TYPE = SpecialEventInfo.class;

        this.URL_DELETE = Config.getAPIUrl(ApiURIs.URI.SE_DELETE);
        this.URL_YEARS = Config.getAPIUrl(ApiURIs.URI.SE_YEARS);
        this.URL_LIST = Config.getAPIUrl(ApiURIs.URI.SE_LIST);
        this.URL_LIST_BY_YEAR = Config.getAPIUrl(ApiURIs.URI.SE_LIST_BY_YEAR);
        this.URL_INSERT = Config.getAPIUrl(ApiURIs.URI.SE_INSERT);
        this.URL_UPDATE = Config.getAPIUrl(ApiURIs.URI.SE_UPDATE);
        this.URL_GET = Config.getAPIUrl(ApiURIs.URI.SE_GET);
    }

    @Override
    protected Comparator<SpecialEventInfo> getComparator() { …  }
}
```

Figure 4.4.2 DataAPI module for Special-Event data

```
public abstract class APIClient<T extends InfoBase> {

    protected String URL_DELETE;
    protected String URL_LIST;
    protected String URL_LIST_BY_YEAR;
    protected String URL_INSERT;
    protected String URL_INSERT_ALL;
    protected String URL_UPDATE;
    protected String URL_GET;
    protected String URL_YEARS;
    protected Type RESPONSE_LIST_TYPE;
    protected Type RESPONSE_TYPE;
    protected Type DATA_TYPE;
    protected Gson gsonBuilder = new GsonBuilder().create();
    public final ArrayList<T> dataList;
    protected List<AbstractDataChangeListener<T>> changeListeners = new ArrayList<>();
    protected boolean isLoadingList = false;
    protected Logger logger;

    // implementations of the below functions are omitted
    public APIClient() { …  }
    public void years() { … }
```

```
public void list() { …  }
public void listByYear(Integer year) { … }
private T getDataById(int id) { … }
public void delete(final List<Integer> ids) { …  }
public void insert(final T obj) { …   }
public void insertAll(final List<T> dataList) { …  }
public void get(final String id) { …  }
public T getSynced(String id) { …  }
public T getByNameSynced(final String name) { …  }
public void update(T exData, final T newData) { …  }
private IHttpResultCallback getListCallback() { …  }
private void notSupportedAPI(String name) { …  }
public String toJson(Object obj) { …  }
protected abstract Comparator<T> getComparator();
public void addChangeListener(AbstractDataChangeListener<T> listener) { …  }
public void removeChangeListener(AbstractDataChangeListener<T> listener) { … }
protected void fireListSuccess() { …  }
protected void fireListFailed(HttpResult httpResult) { …  }
protected void fireGetSuccess(T obj) { …  }
protected void fireGetFailed(HttpResult httpResult, int id) { …  }
protected void fireGetFailed(HttpResult httpResult, String id) { …  }
protected void fireUpdateSuccess(int id) { …  }
protected void fireUpdateFailed(HttpResult httpResult, T obj) { …  }
protected void fireDeleteSuccess(List<Integer> ids) { …  }
protected void fireDeleteFailed(HttpResult httpResult, List<Integer> ids) { …  }
protected void fireInsertSuccess(Integer insertedId) {  …  }
protected void fireInsertFailed(HttpResult httpResult, T obj) { …  }
protected void fireYearsSuccess(List<Integer> obj) { …  }
protected void fireYearsFailed(HttpResult httpResult) { …  }
protected void fireInsertAllSuccess() { …  }
protected void fireInsertAllFailed(HttpResult result, List<T> dataList) { …  }
```

Figure 4.4.3 APIClient module

User Interface for Work-Zone Data

Figures 4.4.4-4.4.7 show the screenshots of the user-interface developed for entering work-zone data. The major features of the User Interface for editing work-zone data are as follows:

- Figure 4.4.4 shows the map-based Work-Zone List Panel, which displays the list and location of the work-zones for selected corridors and construction time-periods. In this panel, user can edit the information on specific work-zones or add/delete work-zones. The example screen shown in Figure 4.4.4 includes two work-zones selected on the I-35W NB and SB.

- Figure 4.4.5 is a dialog to add a new work-zone. This dialog pops up when the 'add' button in the List Panel is clicked. In this dialog, user can enter general information of a new work-zone, such as name, description and construction duration.

-User also can define a 'Work-zone route', i.e., the direction and boundaries of a work zone in terms of detector station IDs. When user specifies a work-zone route for one direction, the interface automatically creates a second route in an opposing direction with the same upstream/downstream boundaries as with the first route.

- The detailed lane-configuration within a work-zone can be specified by clicking '*Edit Lane-Configuration*' button, which displays a dialog with two choice buttons as shown in Figure 4.4.6.

- To create a new lane-configuration file, click '*Create New Lane-Configuration File*' button, which will open a spread-sheet file with a set of default information for current work zone, such as mile points, station/ramp ids and open/closure status of ramps, etc. The detailed lane-closure conditions of a given work zone can be entered by user in this spreadsheet using the symbols shown in Table 4.4.1.

Table 4.4.1 Symbols in Lane-Configuration Spreadsheet File

Symbol	**Remarks**
↓,↑	traffic flow direction on mainline
↓~~, ~~↓, ↑~~, ~~↑	lane shifted
↓X, ↑X,	lane closed
O	ramp opened
X	ramp closed
↓ (A), ↑ (A)	auxiliary lane
↓ (H), ↑ (H)	HOV(T) lane
S<Number>	Station ID
E:<Label>	Entrance ID
X:<Label>	Exit ID

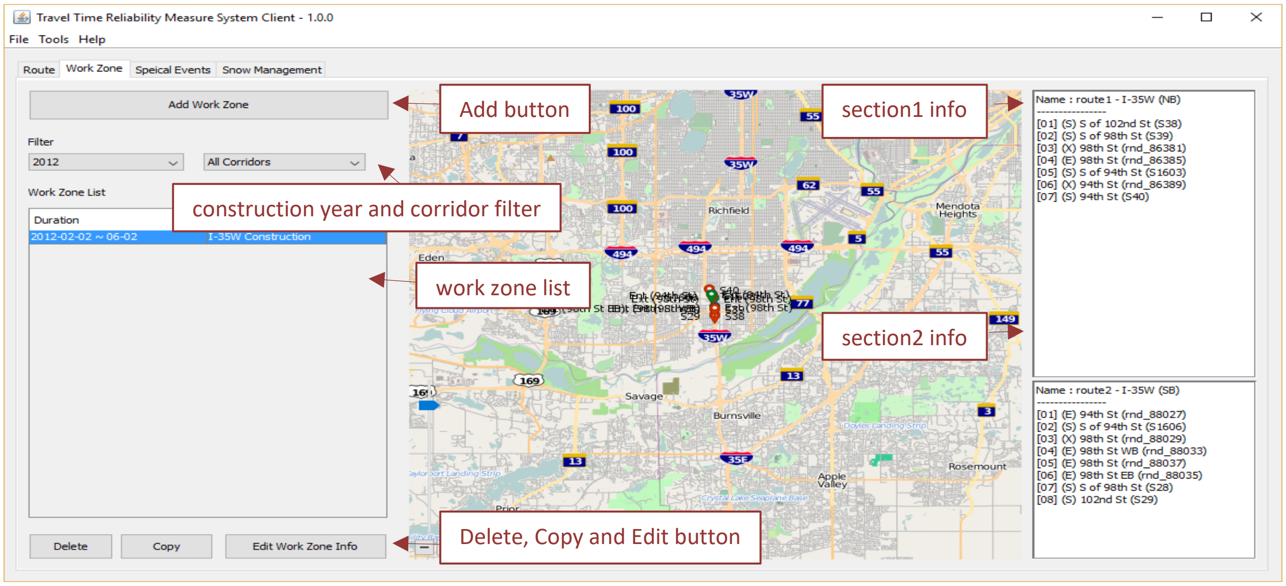


Figure 4.4.4 Work-Zone List Display Panel

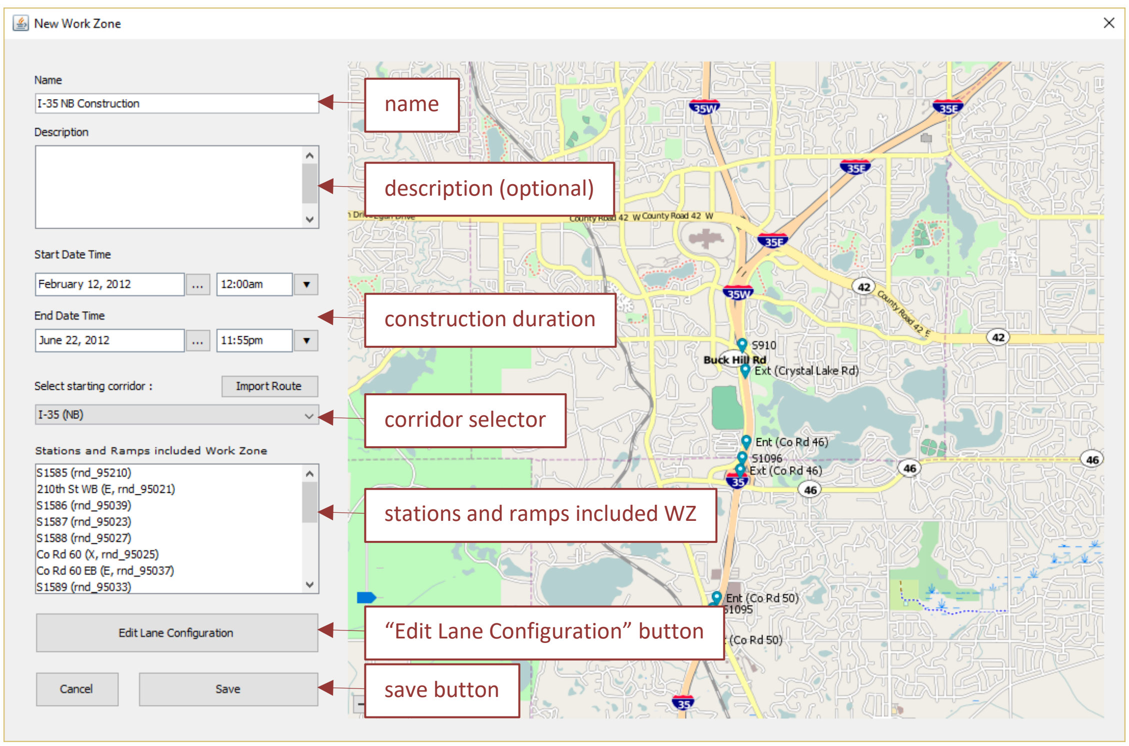


Figure 4.4.5 Dialog to Add a New Work-Zone

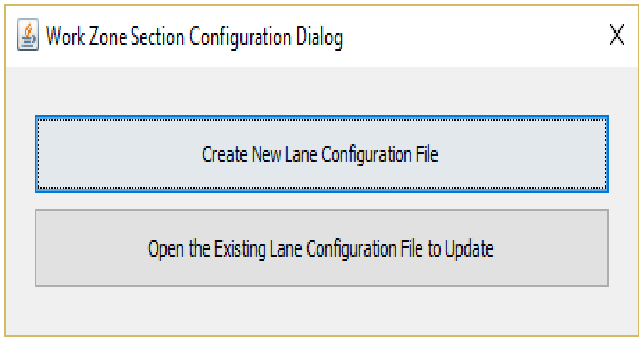


Figure 4.4.6 Lane Configuration Edit Dialog

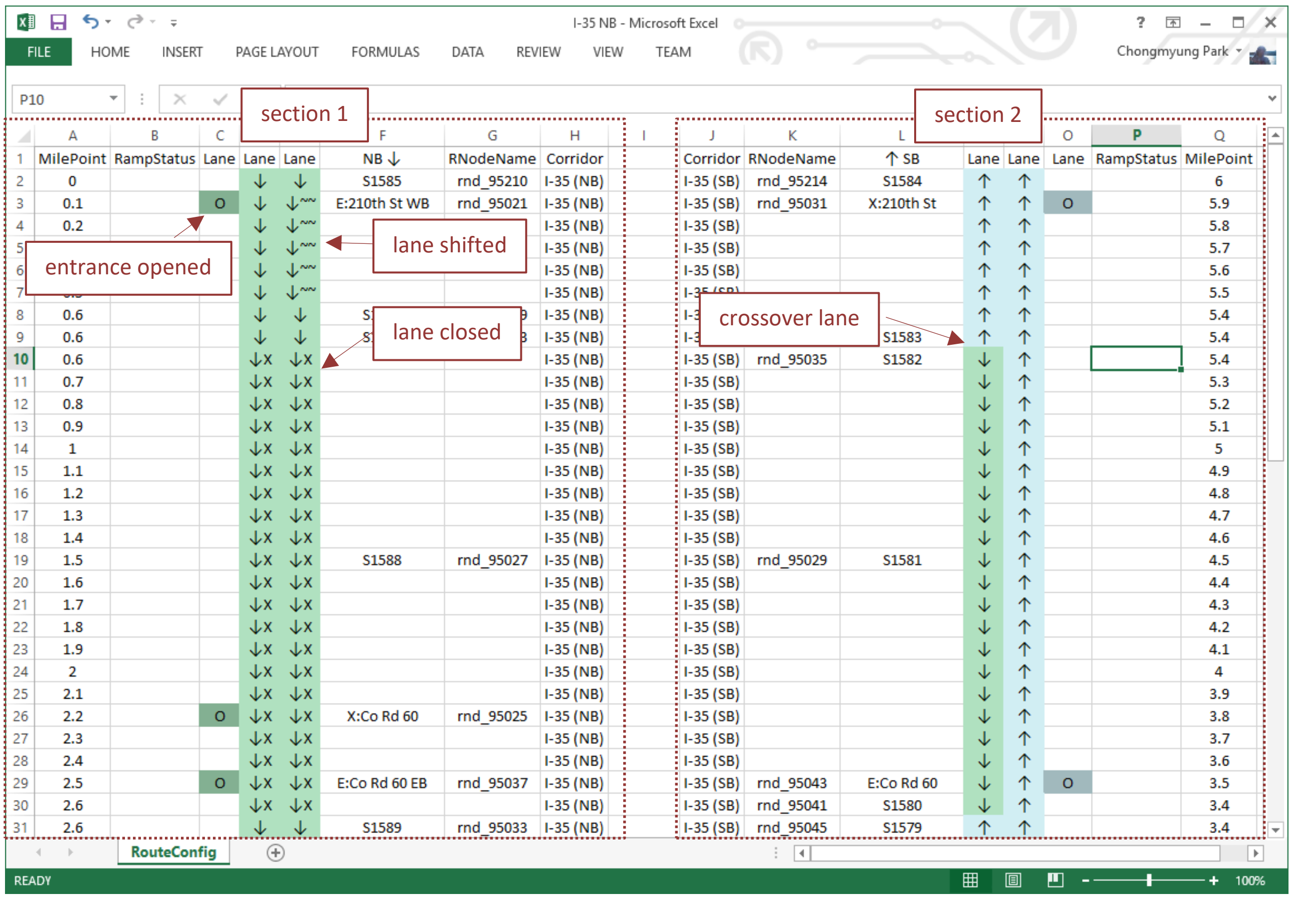


Figure 4.4.7 Spreadsheet for Lane Configuration

User Interface for Special-Event Data

Figure 4.4.8 shows the Special-Event data user-interface, where user can add, edit and delete special-event data. Further, a list of special events selected by user for a specific year can be shown in this panel in a table format. Figure 4.4.9 is a dialog to add a new special-event data, which include name, location and event duration. In particular, the location of a new event can be entered from the background map of the user interface.

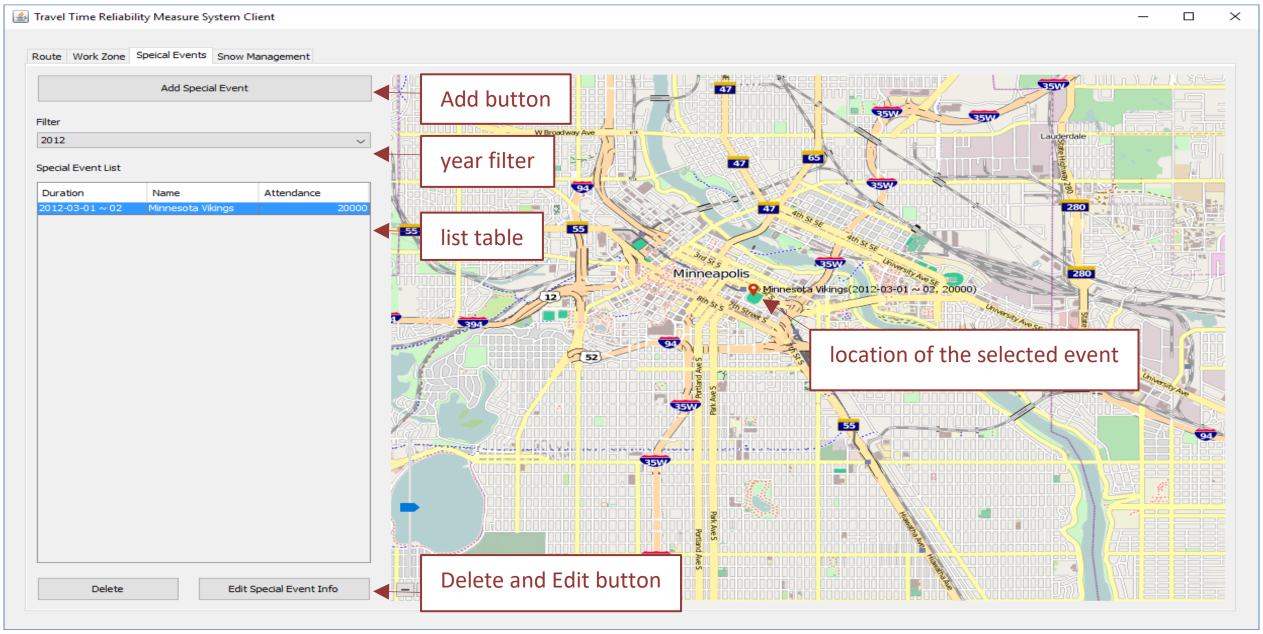


Figure 4.4.8 Main Window of Special-Event User Interface

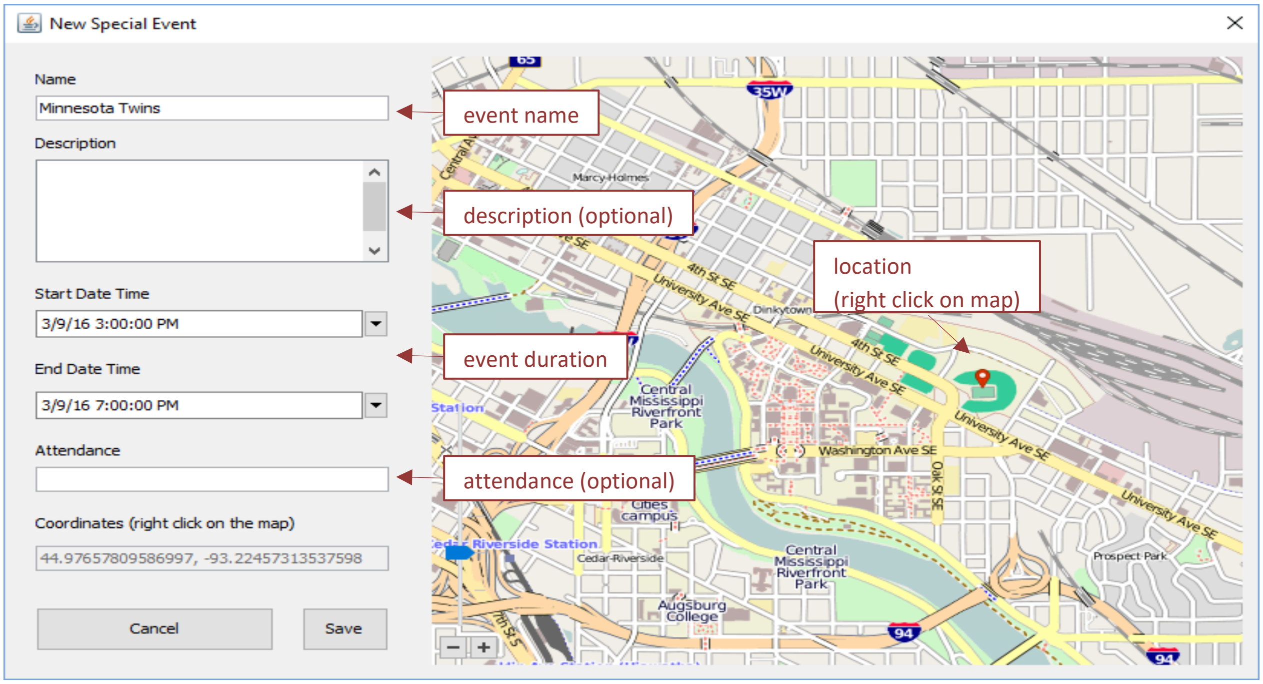


Figure 4.4.9 Dialog to add a New Special Event

User Interface for Snow Management Data

The user interface for managing snow-management data consists of two tabs, “Snow Management Information” and “Snow Management Section” tabs as shown in Figure 4.4.10. The past snow-event data can be entered and edited in the ‘Snow Management Information’ tab, where user can retrieve a set of past snow events in a table format. The specific information on a selected snow event, such as snow start/end time and lane regain time, can also be seen in this panel. Figure 4.4.11 shows the dialogs where user can enter new snow-event data, which include event duration, snow management section boundaries, lane lost and regain times.

Figure 4.4.12 shows “Snow Management Section” tab, where detailed information regarding snow-management sections can be entered, edited and deleted by user. Further, a list of snow-management sections on each freeway corridor can be generated in this panel in a table format that also includes specific information on each snow-management section. Figure 4.4.13 shows a dialog to add a new snow management section, whose data includes section boundaries in terms of detector station IDs, name, description and snow-route ID managed by MnDOT. Each Snow-Management section consists of two routes, i.e., one for each direction. When a route for one direction is created, the other route for an opposing direction is created automatically by the user interface.

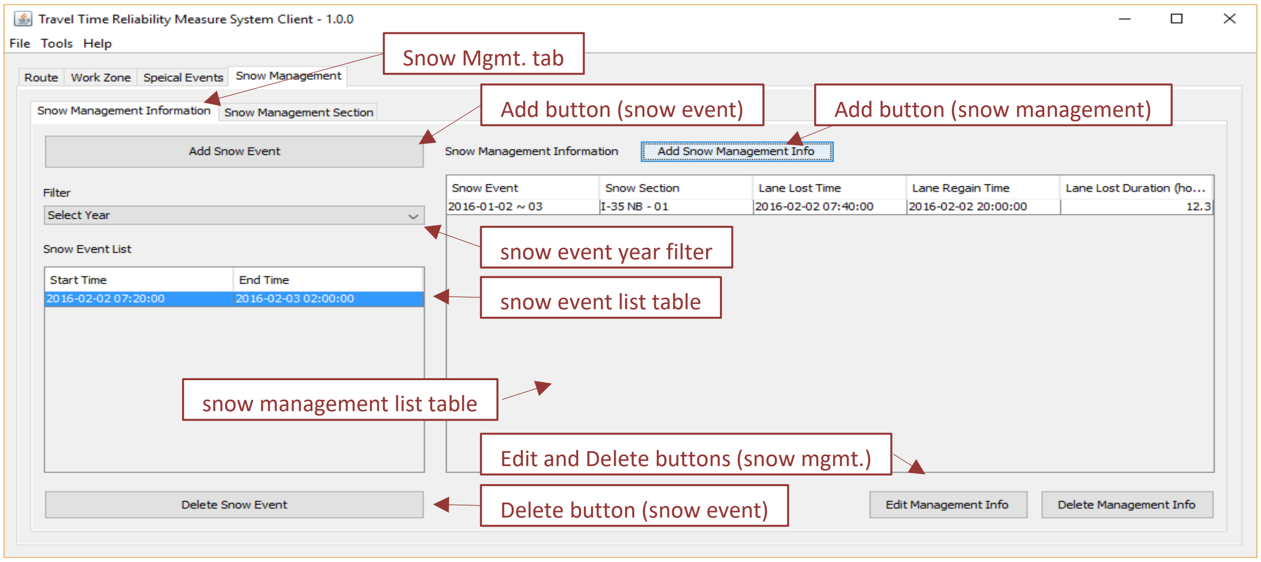


Figure 4.4.10 Snow Management Information panel

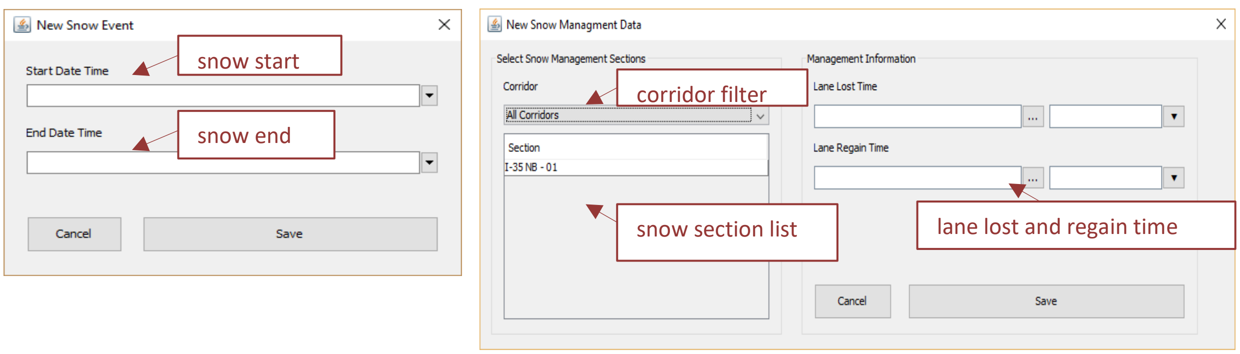


Figure 4.4.11 Dialog for New Snow-Event and Management Data

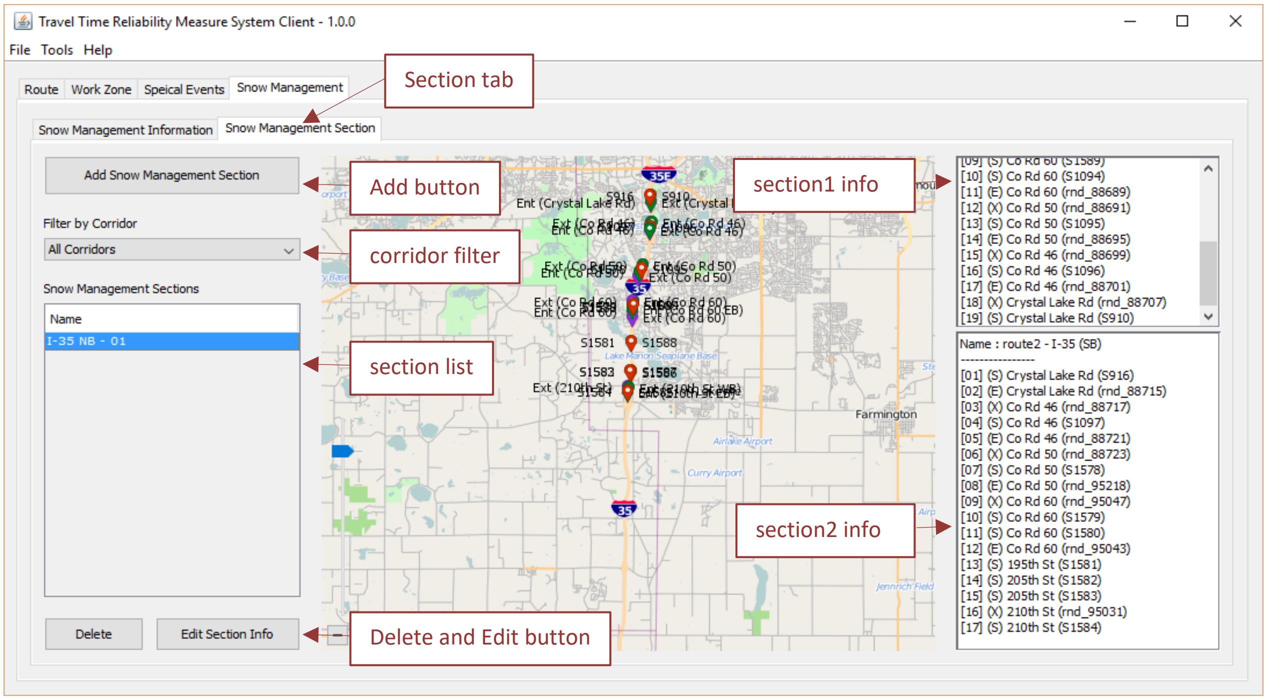


Figure 4.4.12 Snow-Management Section List panel

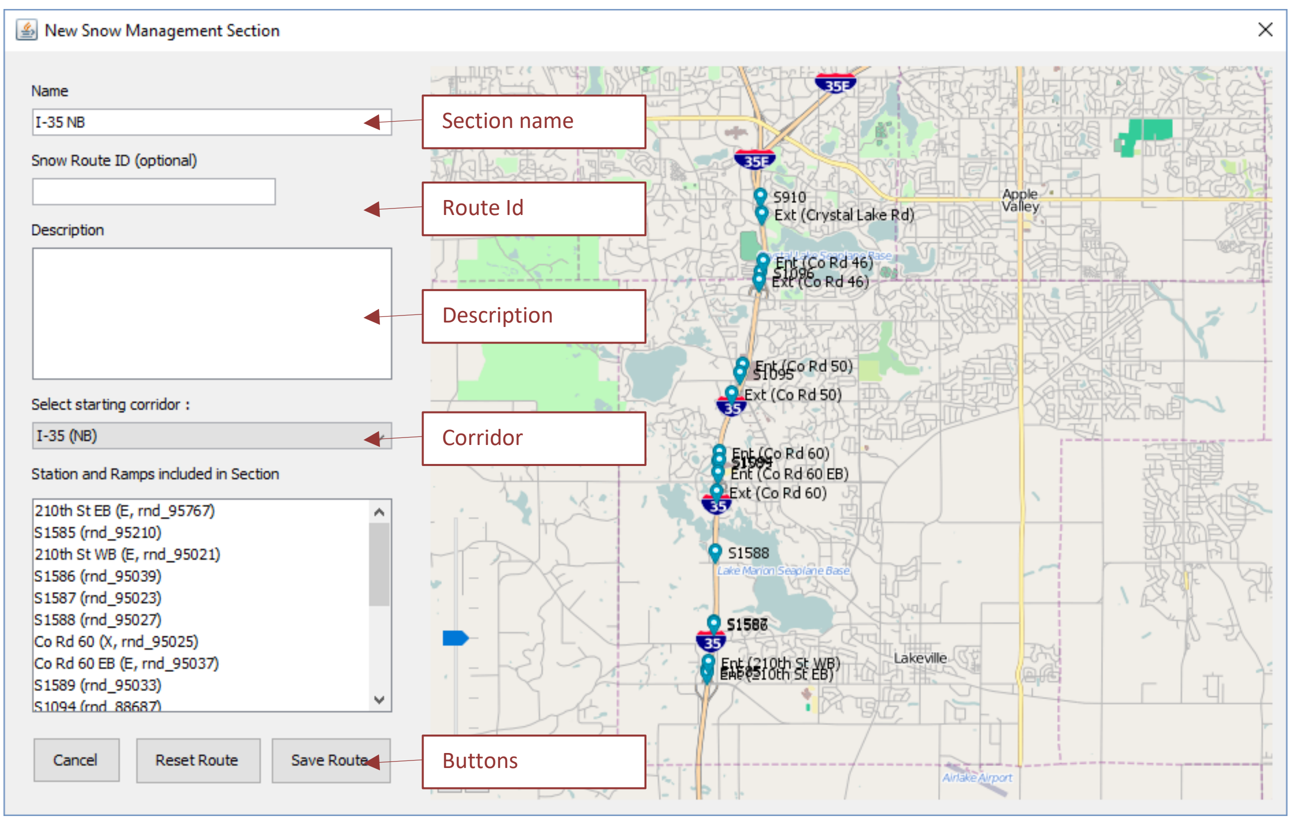


Figure 4.4.13 Dialog for New Snow-Management Section

4.5 DEVELOPMENT OF TRAVEL-TIME RELIABILITY-ROUTE CONFIGURATION MODULES

In this section, a set of the modules are developed to be used for configuring travel-time routes, whose reliability measures will be calculated by TTRMS. Figure 4.5.1 shows the travel-time route configuration process, which includes two sub-processes, i.e., route-configuration through the map-based user interface and the conversion and storing of user-specified route data into the database. The route-configuration process with the user interface is managed by the Administration Client, while the data-conversion process uses those modules developed in the previous chapter.

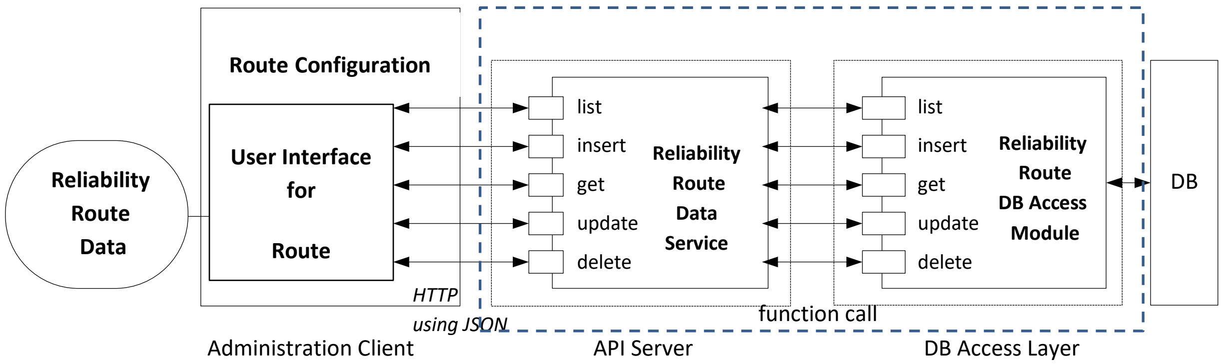


Figure 4.5.1 Travel-Time Route Configuration Process

User Interface for Route Configuration

Figure 4.5.2 shows the main-panel of the user interface, where user can add, edit and delete reliability routes. Further, a list of existing routes can be displayed in a table format including detailed information on a selected route. Figures 4.5.3 and 4.5.4 show a dialog for adding a new reliability route, e.g., a route between the southern boundary of I-35 and I-35E NB/TH 77. As shown in this example, if an exit ramp is included in a multi-corridor route, an additional dialog is used to specify a corresponding corridor. Also, a second route in an opposing direction can be automatically created depending on user choice.

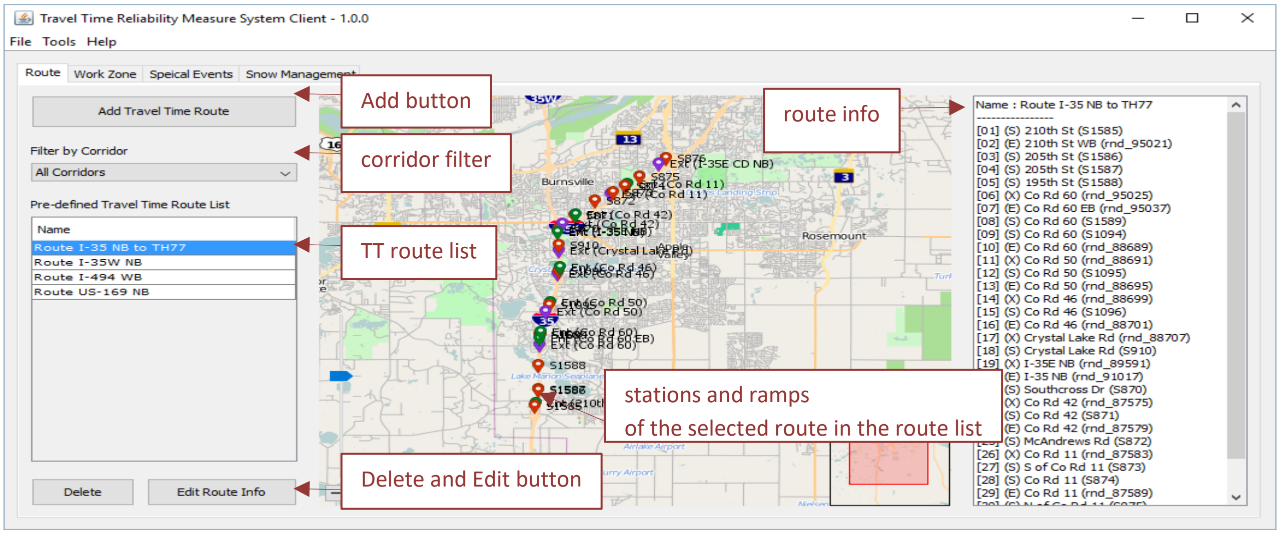


Figure 4.5.2 Reliability-Route Configuration Main User Interface

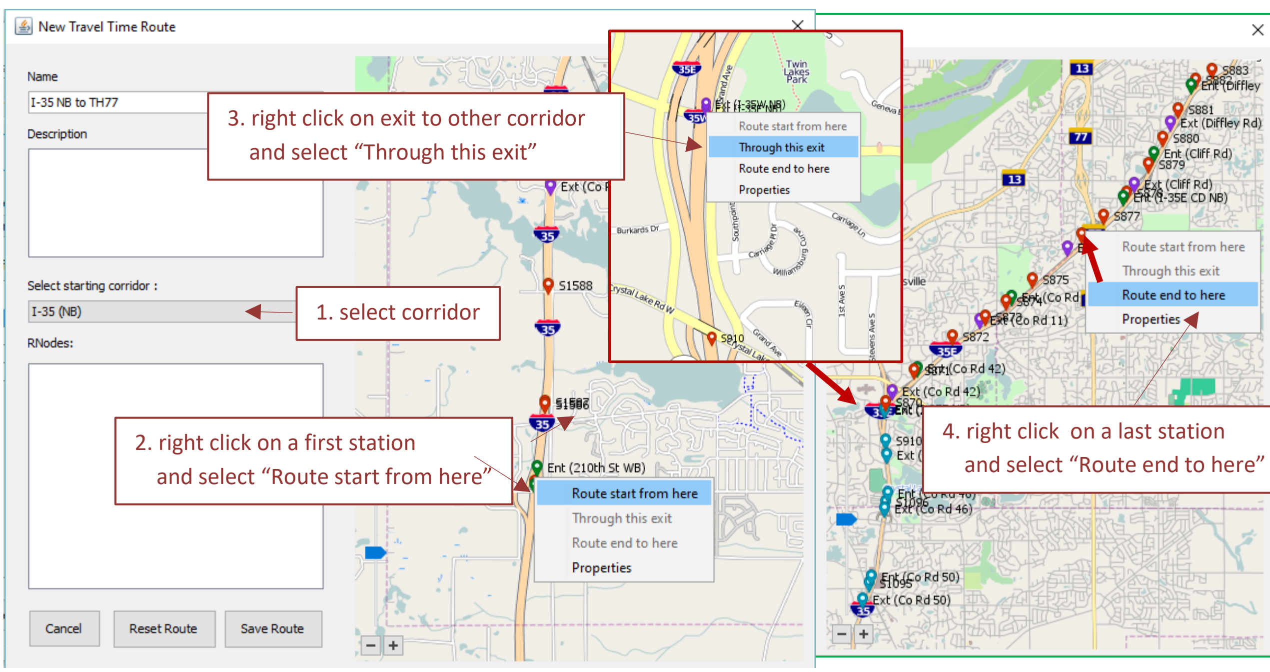


Figure 4.5.3 Dialog for New Reliability-Route

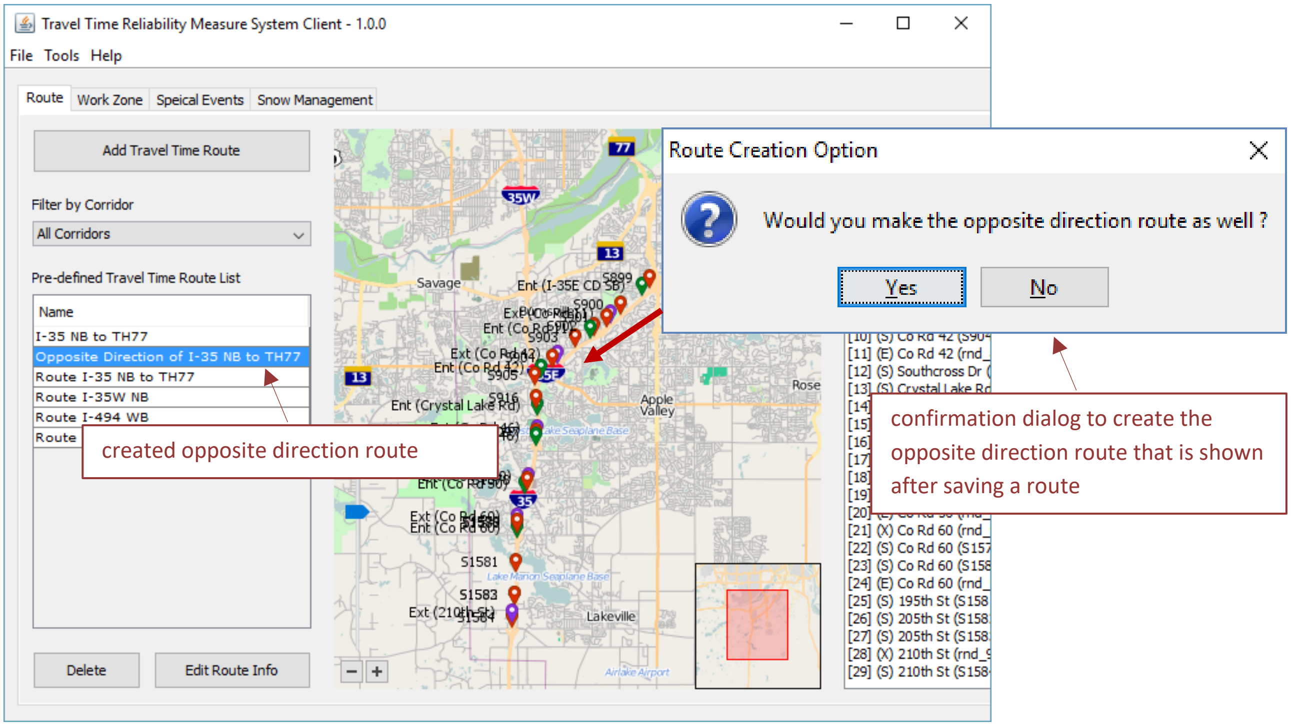


Figure 4.5.4 Confirmation Dialog to Create Opposite Direction Route

Route-Data Conversion Process

The route-specific data entered by user, i.e., system administrator, are converted and stored in the database with the same process developed in the previous chapter for external non-traffic data. The route-specific data, including a list of stations, ramps and corridor IDs, are first converted into a JSON string in the Administration Client, which sends them to the entry point of the Reliability Route Service of the API Server using the HTTP POST method. The entry point to insert a reliability route is specified as the URL of *"/ttrms/admin/ reliability_route/insert"*. The JSON-string data are then converted to Python objects in the *Reliability Route Service* for the *Reliability Route DB Access* modules to interface with the database. Figure 30 includes the class definition for 'Reliability Route' in the Administration Client and Figures 4.5.5-8 show the source codes of the modules used for the route-data conversion process, i.e., Data API, Reliability-Route Service and Data Access modules. Further, Figure 4.5.9 shows a reliability-route database table entered through the user interface for the example route in Figure 4.5.2. The freeway route created in Figure 4.5.2 is saved as JSON string to 'route' field of the database table, whose freeway section data including the list of stations and ramps are shown in Figure 4.5.10.

```
public class ReliabilityRouteInfo  extends InfoBase {
   public String name;
   public String description;
   public String corridor;
   public Route route;
   public ReliabilityRouteInfo() {
      this.setTypeInfo(TTRMSConfig.INFO_TYPE_TTROUTE);
   }

   // implementations of the below functions are omitted
   public ReliabilityRouteInfo(Route r) { … }
}
```

Figure 4.5.5 Class definition for Reliability-Route in Client

```
public class ReliabilityRoute extends APIClient<ReliabilityRouteInfo> {

   public ReliabilityRoute() {
      this.RESPONSE_LIST_TYPE = new TypeToken<ListResponse<ReliabilityRouteInfo>>() {}.getType();
      this.RESPONSE_TYPE = new TypeToken<ObjectResponse<ReliabilityRouteInfo>>() {}.getType();
      this.DATA_TYPE = ReliabilityRouteInfo.class;
      this.URL_DELETE = Config.getAPIUrl(ApiURIs.URI.ROUTE_DELETE);
      this.URL_LIST = Config.getAPIUrl(ApiURIs.URI.ROUTE_LIST);
      this.URL_INSERT = Config.getAPIUrl(ApiURIs.URI.ROUTE_INSERT);
      this.URL_UPDATE = Config.getAPIUrl(ApiURIs.URI.ROUTE_UPDATE);
      this.URL_GET = Config.getAPIUrl(ApiURIs.URI.ROUTE_GET);
   }

   // implementations of the below functions are omitted
   public Route opposingRoute(int id) { …  }

   @Override
   protected Comparator<ReliabilityRouteInfo> getComparator() { …   }
}
```

Figure 4.5.6 Data API module for Reliability-Route in Client

```
def register_api(app):
    autodoc = get_autodoc()

    TTRMSApi(app, 'ttroute', json2ttri, RouteDataAccess, {
        'insert': (api_urls.ROUTE_INSERT, ['POST']),
        'list': (api_urls.ROUTE_LIST, ['GET']),
        'get_by_id': (api_urls.ROUTE_GET, ['POST']),
        'get_by_name': (api_urls.ROUTE_GET, ['POST']),
        'update': (api_urls.ROUTE_UPDATE, ['POST']),
        'delete': (api_urls.ROUTE_DELETE, ['POST']),
    }).register()

    @app.route(api_urls.ROUTE_OPPOSITE_ROUTE, methods=['POST'])
    @autodoc.doc()
    def ttrms_route_opposite_route():
        route_id = request.form.get('id')

        da = RouteDataAccess()
        ttri = da.get_by_id(route_id)
        da.close()

        route_setup(ttri.route)
        opposite_route = route.opposite_route(ttri.route)
        if not isinstance(opposite_route, Route):
            return prot.response_fail('fail to load route configuration file')
        return prot.response_success(opposite_route)
```

Figure 4.5.7 Reliability-Route Service module in Server

```
class DataAccess(object):

    def __init__(self, **kwargs):
        self.da_base = DataAccessBase(model.TTRoute, TTRouteInfo, **kwargs)
    def list(self):
        return self.da_base.list()
    def get_by_id(self, route_id):
        return self.da_base.get_data_by_id(route_id)
    def get_by_name(self, route_name):
        return self.da_base.get_data_by_name(route_name)
    def delete(self, id, autocommit=False):
        return self.da_base.delete(id, autocommit=autocommit)
    def insert(self, r, autocommit=False):
        return self.da_base.insert(r, autocommit=autocommit)
```

```
def update(self, id, field_data, autocommit=False):
    return self.da_base.update(id, field_data, autocommit=autocommit)
def rollback(self):
    self.da_base.session.rollback()
def commit(self):
    self.da_base.commit()

def close(self):
    self.da_base.close()
```

Figure 4.5.8 Data-Access module for Reliability-Route Data in Server

id	name	description	corridor	route	
	Filter	Filter	Filter	Filter	Filter
1	Route I-35W NB	Route created at 2016-0...	I-35W (NB)	{"rnodes": ["rnd_86379", "rnd_86383", "rnd_86381", "r...	2016-
2	Route I-494 WB	Route created at 2016-0...	I-494 (WB)	{"rnodes": ["rnd_87009", "rnd_89027", "rnd_87047", "r...	2016-
3	Route US-169 NB	Route created at 2016-0...	U.S.169 (NB)	{"rnodes": ["rnd_84663", "rnd_84665", "rnd_84671", "r...	2016-
4	Route I-35 NB to TH77		I-35 (NB)	{"rnodes": ["rnd_95210", "rnd_95021", "rnd_95039", "r...	2016-
5	I-35 NB to TH77		I-35 (NB)	{"rnodes": ["rnd_95767", "rnd_95210", "rnd_95021", "r...	2016-
6	Opposite Direction of I-35 ...		I-35E (SB)	{"rnodes": ["rnd_86891", "rnd_86877", "rnd_85977", "r...	2016-

Figure 4.5.9 Sample Reliability-Route Data stored in Database

```
{
  "rnodes": [
    "rnd_95767",    "rnd_95210",    "rnd_95021",    "rnd_95039",    "rnd_95023",
    "rnd_95027",    "rnd_95025",    "rnd_95037",    "rnd_95033",    "rnd_88687",
    "rnd_88689",    "rnd_88691",    "rnd_88693",    "rnd_88695",    "rnd_88699",
    "rnd_88705",    "rnd_88701",    "rnd_88707",    "rnd_88709",    "rnd_89591",
    "rnd_91017",    "rnd_87573",    "rnd_87575",    "rnd_87577",    "rnd_87579",
    "rnd_87581",    "rnd_87583",    "rnd_87585",    "rnd_87587",    "rnd_87589",
    "rnd_87591",    "rnd_90763",    "rnd_88521"    ],
  "desc": "",
  "name": "I-35 NB to TH77",
  "__class__": "Route",
  "cfg": null,
  "__module__": "pyticas.ttypes"
}
```

Figure 4.5.10 “route” Field Data of Sample Reliability-Route Database

CHAPTER 5: DEVELOPMENT OF THE TRAVEL TIME RELIABILITY COMPUTATION MODULE

5.1 INTRODUCTION

In this chapter, the *Travel-Time Reliability Computation module (TTRCM)* is developed as the main computational engine of the TTRMS, Travel-Time Reliability Measurement System, being developed in this research. The main functionalities of the TTRCM developed in this chapter include,

- Calculation of the travel times for selected routes during given time periods,
- Association of calculated travel times to corresponding operating conditions specified by non-traffic data,
- Calculation of travel-time reliability indices for given operating conditions.

Further, to address the needs for long-term, stable accessibility of weather data, the *Weather-Data Reader,* developed in the previous chapter, has been enhanced to be able to download the data from the National Oceanic and Atmospheric Administration (NOAA). A new Incident-Data Reader module that can read the incident data stored in the CAD and IRIS databases is also developed in this chapter.

Figure 5.1.1 shows the TTRCM and its sub-modules developed in this chapter in the overall architecture of the TTRMS. They include:

1) ***Weather Data Reader,*** which reads weather data and stores those into the TTRMS database.

2) ***Incident Data Reader,*** which reads the incident data from the CAD/IRIS databases and combines/stores them into the TTRMS database.

3) ***Travel Time Calculation,*** which calculates the travel times for given corridors and time periods explicitly reflecting the lane configurations of work zones if applicable.

4) ***Data Categorization***, which categorizes travel times by linking them to different operating conditions specified with non-traffic data.

5) ***Reliability Calculation,*** which calculates the values of the travel-time reliability indices, whose types are pre-defined.

The rest of this chapter describes the details of the above modules.

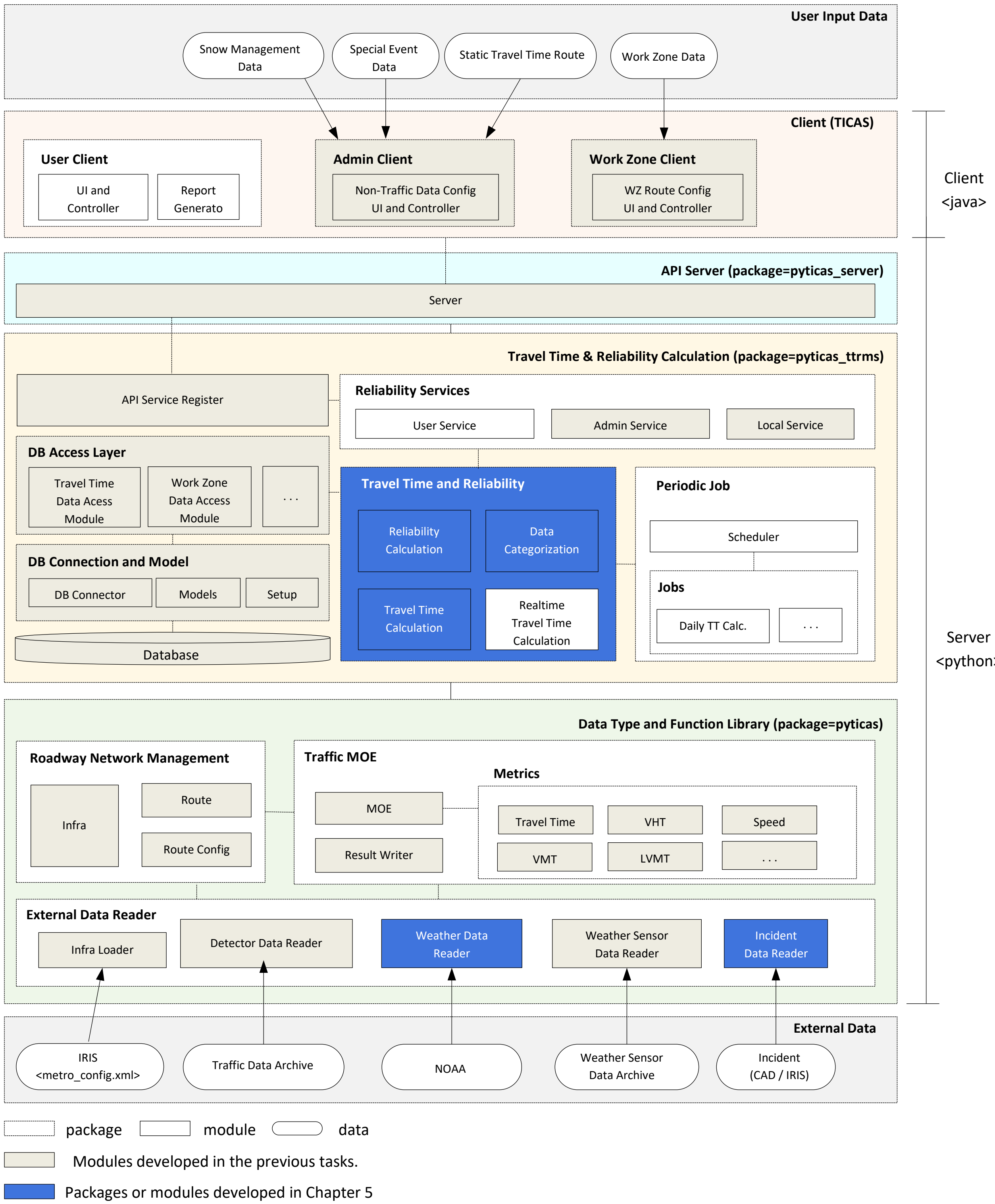


Figure 5.1.1 The Modules developed in Chapter 5 in TTRMS Architecture

5.2 DEVELOPMENT OF WEATHER AND INCIDENT DATA READER MODULES

5.2.1 Development of Additional Data Reader module for NOAA Weather Data

In this section, the Weather-Data-Reader module, developed in the previous chapter for the MnDOT RWIS database, was enhanced by adding an additional module to access the weather data from NOAA, where diverse types of weather data are expected to be available on a long-term basis. Figure 5.2.1 shows the data flow among the submodules developed in this chapter to import and process the weather data from the NOAA's Integrated Surface Data (ISD) archive. The weather data imported by the Weather Data Reader module are further processed by the Daily Data Processing (DDP) module, which stores them into the TTRMS database on a daily basis. Table 5.2.1 shows the database table designed in this study to store the NOAA weather data. The DDP module also processes/stores the incident and travel time data into the database, which can be accessed by other modules, such as the Reliability estimation module. The DDP module is scheduled to be developed in the subsequent task in this study.

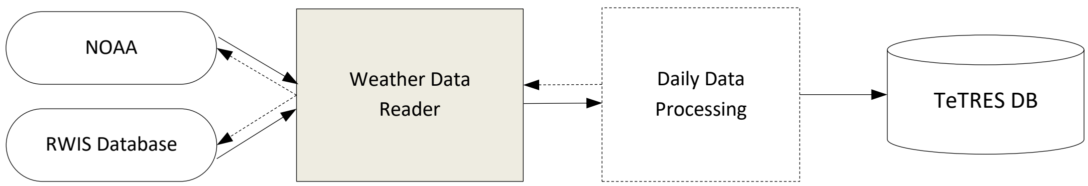
The main functions of the submodules in the Weather Data Reader module shown in Figure 5.2.1 are as follows:

- *ISD Station Information Reader* module downloads the list of the available weather stations from the NOAA FTP site and parses them.
- *ISD Data Reader* module downloads the archived weather data from NOAA FTP site for the selected stations and parses them.
- *NOAA Weather Data Read Interface* module contains a set of functions needed by other modules, e.g., *ISD Station Information Reader* and *ISD Data Reader,* for processing the weather data, such as for finding near-by weather stations for given coordinates and also for identifying weather data for given time periods. It can be noted that the current weather stations located in the Twin Cities' metro area are pre-configured for TTRMS with the *ISD Station Information Reader.* Figure 5.2.2 shows the locations of the 7 weather stations currently being used by NOAA for the Twin Cities metro area.

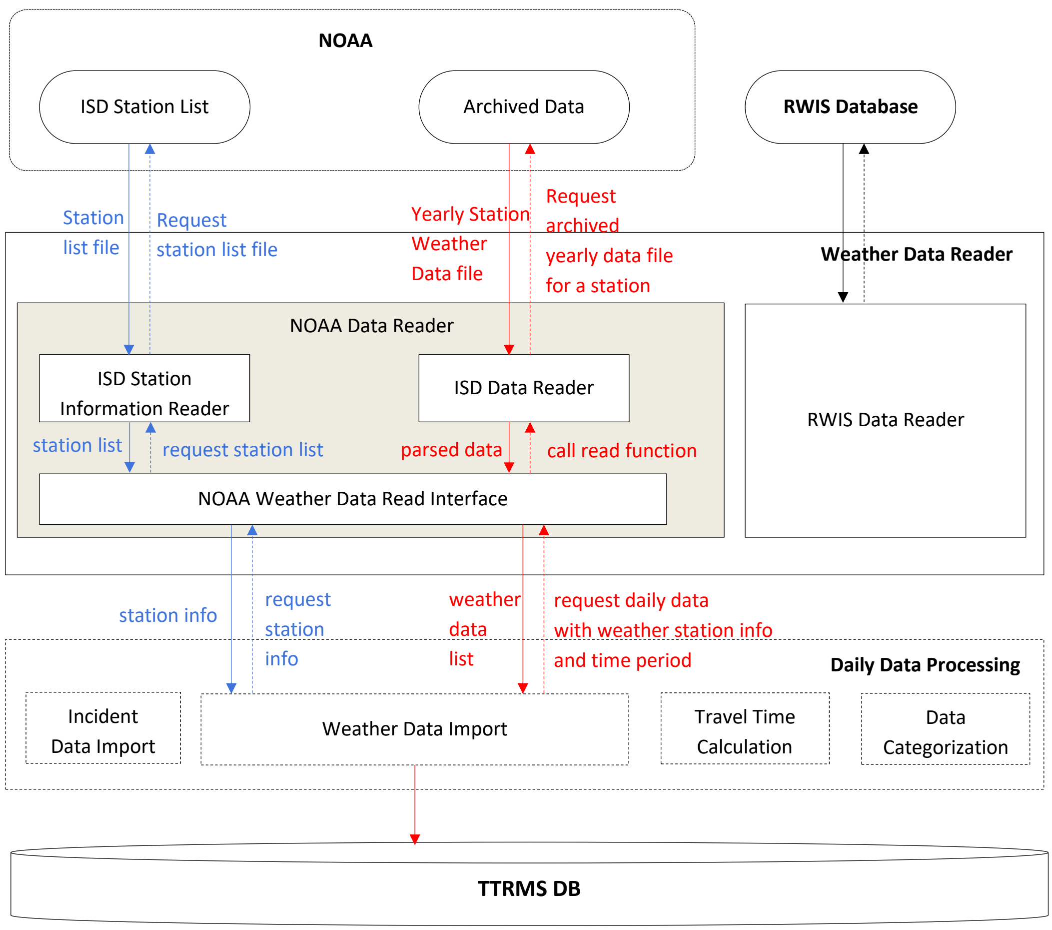
Table 5.2.1 Weather Data Table

Table Name	Data Fields
noaa_weather	- id : (int) sequential number - usaf : (string) identification of U.S. Air Force for weather station - wban: (string) identification of Weather-Bureau-Army-Navy for weather station - dtime: (datetime) timestamp - precip: (float) precipitation for one hour (inch) - precip_type: (string) precipitation type e.g. No Precip, Drizzle, Raion, Snow - precip_intensity: (string) precipitation intensity e.g. Light, Moderate, Heavy - relative_humidity: (float) relative humidity (%)

	- visibility: (float) visibility (mile) - air_temp: (float) air temperature (F) - dew_point: (float) dew point (mile) - wind_dir: (int) wind direction (0 – 360) - wind_speed: (float) wind speed (mph) - wind_gust: (float) wind gust (mph)



(a) Data Flow for Weather Data Processing



(b) Detailed Structure of Weather Data Reader module

Figure 5.2.1 Data Flow and Structure of Weather Data Reader Module

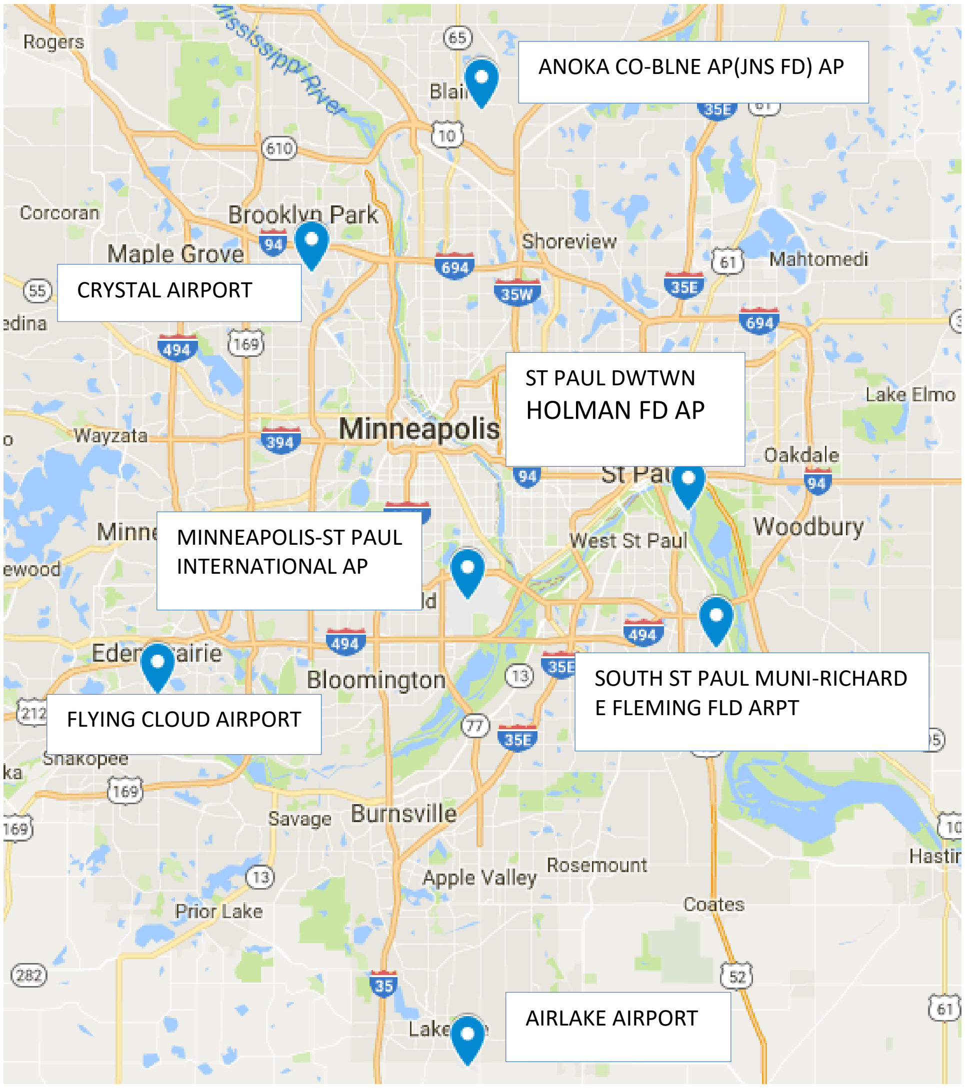


Figure 5.2.2 Weather Stations in the Twin Cities Metro Area

5.2.2 Development of Incident Data Reader module

Figure 5.2.3 shows the structure of the *Incident Data Reader* module developed in this chapter to import the incident-related data for the Twin Cities' metro freeway network from two external data sources, i.e., CAD (Computer-Aided Dispatch) system from the Department of Public Safety and IRIS of RTMC, MnDOT.

As shown in Figure 5.2.3, the imported data from each data source are integrated into a combined data format by the Incident-Data-Integration submodule. In this study, the Incident-Impact data, i.e., lane-closure status, and Lane-type information from the IRIS-Incident database are extracted and merged onto the data from the CAD system, which is considered as the main data source for incidents. The integrated- incident data is then accessed and stored in the main database of TTRMS by the Incident-Data-Import submodule in the ‘Daily Data Processing’ module. The incident-data tables of TTRMS are shown in Table 5.2.2.

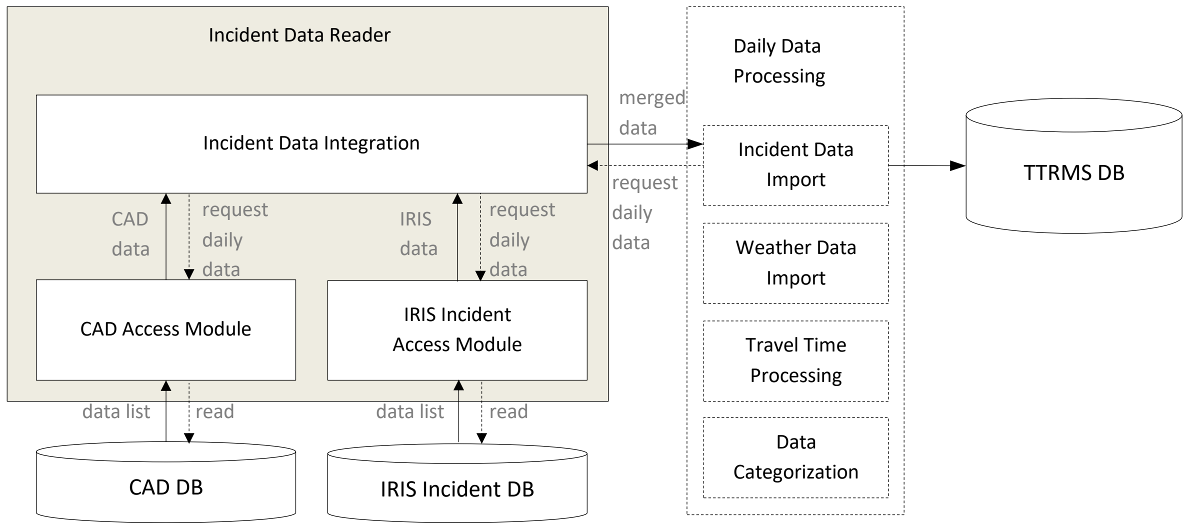


Figure 5.2.3 Incident Data Reader module

Table 5.2.2 Incident Data Tables

Table Name	Data Fields
incident	- id: (int) sequential number - incident_type_id: (int) foreign key referring incident type - cad_pkey: (int) pky field of CAD table - iris_event_id: (int) event_id field of IRIS incident table - iris_event_name: (string) event_name field of IRIS incident table - cdts: (datetime) created timestamp - udts: (datetime) updated timestamp - xdts: (datetime) closed timestamp - lat: (float) latitude - lon: (float) longitude - road: (string) corridor name - direction: (string) direction e.g. NB, SB, WB, EB - impact: (string) closed lane information - lane_type: (string) lane type e.g. MAINLINE, EXIT, ENTRANCE
incident_type	- id: (int) sequential number - eventtype: (string) event type name - eventsubtype: (string) event sub-type name - eventtypecode: (string) event type code - eventsubtypecode: (string) event sub-type code - classification: (string) incident classification - iris_class: (string) incident classification used in IRIS

	- iris_detail: (string) incident details used in IRIS - blocking: (boolean) true if there is lane blocking - occupied: (boolean) true if road is occupied - rollover: (boolean) true if there is rollover - injury: (boolean) true if there is personal injury - fatal: (boolean) true if there is fatal injury - cars_eventtype: (string) event type for cars e.g. FATALITY CRASH, VEHICLE SPINOUT - cars_eventcode: (string) event type code for cars

5.3 DEVELOPMENT OF TRAVEL-TIME PROCESSING MODULE

The *Travel-Time Processing Module* calculates the travel times for given routes and time periods specified by user. The calculated travel times are then stored in the main database through the Database Access Layer. In particular, the specific lane-configuration of each work zone is explicitly reflected in calculating the travel times for work-zone routes. Further, the current version of TTRMS is designed to process the travel-times for pre-defined routes on a daily basis. Figure 5.3.1 shows the structure and data flow of the Travel-Time Processing Module, which includes the following submodules:

- ***Work Zone Lane Configuration*** module collects the lane-closure information stored in the main database for the work zones in a given route, and configures the travel-time route with appropriate detector stations.

- ***Travel-Time Calculation (TTC)*** module calculates the travel times for a given route and time period by calling the travel-time routine in the *Traffic MOE* module. The TTC module also manages the calculation of Vehicle-Miles Traveled (VMT) and average speed for a given route using the VMT and Speed routines of the *Traffic MOE* module. The calculated travel-time, VMT and average speed data for a given route are then sent to the main database through the DB Access Layer. Table 5.3.1 shows the database table schema with a foreign key linking to corresponding travel-time routes.

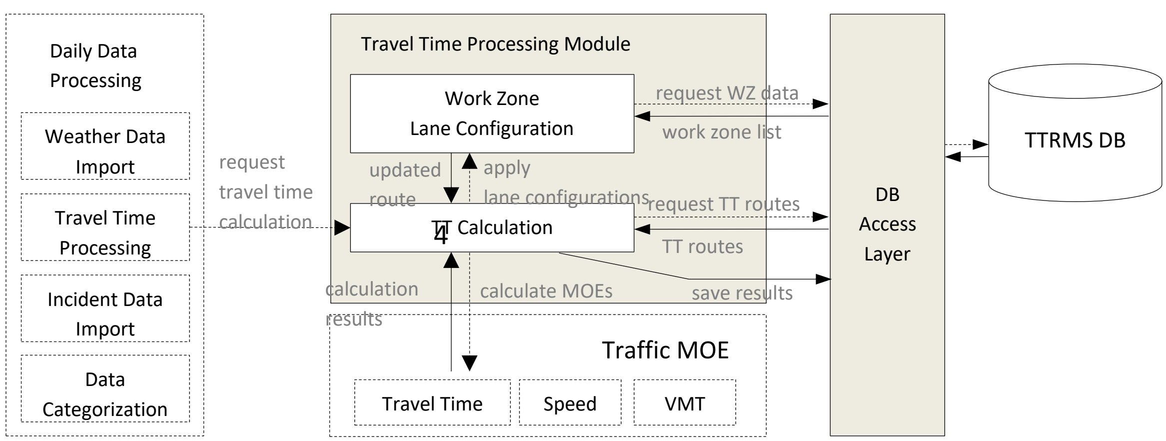


Figure 5.3.1 Structure of Travel-Time Processing module

Table 5.3.1Travel-time table schema

Table Name : tt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
id	**INTEGER**	PK	NN	UNSIGNED			AI
route_id	**INTEGER**	PK	NN	UNSIGNED			
time	DATETIME		NN				
tt	FLOAT		NN				
vmt	FLOAT		NN				
speed	FLOAT		NN				

5.4 DEVELOPMENT OF DATA CATEGORIZATION MODULE

5.4.1 Design of Data Categorization module

The Data-Categorization module defines the relationships between travel-time data and the non-traffic data, such as weather and incident, for every time interval for each route. Figure 5.5.1 shows the simplified structure and the data flow with the Data-Categorization module, which consists of:

- *Categorization-Management* sub-module provides the access point for other modules and runs the categorization functions of the sub-modules.
- Individual Categorization sub-modules for each type of non-traffic data, i.e., *Weather, Incident, Work Zone, Special Event and Snow,* categorize each type of the non-traffic data.

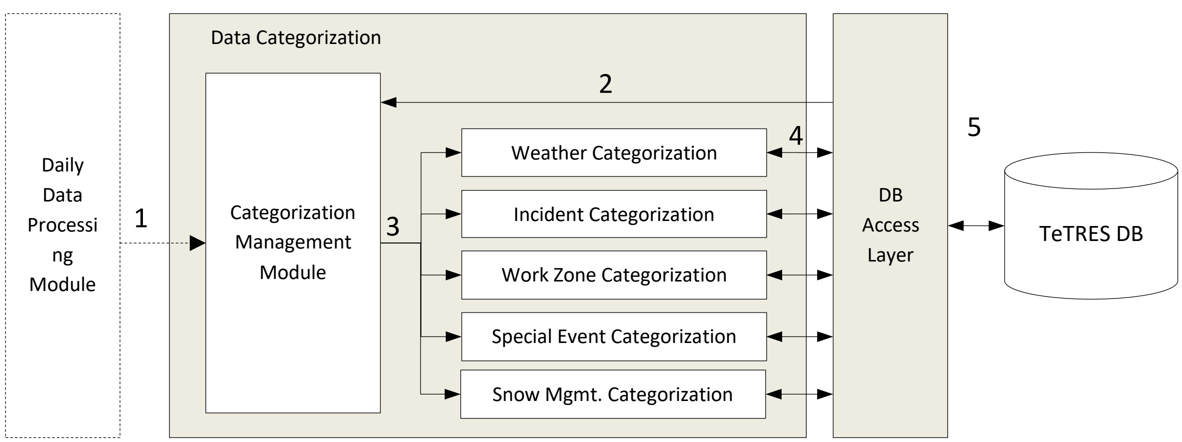


Figure 5.4.1 Data flows of Data Categorization module

Data-Categorization Process

As shown in Figure 5.4.1, the *Data Categorization* process follows the following steps.

Step 1: *Categorization* module is called with the route and time duration information by *Daily Data Processing* module.

Step 2: *Categorization* module reads travel time data from database.

Step 3: Sub categorization modules are called with the route, time duration and loaded travel time data.

Step 4: Each sub-categorization module reads the corresponding non-traffic data from the database and makes relationships between travel time and non-traffic data.

Step5: Categorized data are saved in the database by each sub-categorization module.

Database Tables

Figure 5.4.2 shows the relationship diagram of the database used by the Data-Categorization module. The main features of the database can be summarized as follows:

- “tt” table stores travel time data for each time interval, with foreign keys linking to travel time routes.
- non-traffic data are stored in “noaa_weather”, “incident”, “specialevent”, “workzone” and “snowmgmt” tables.
- there are junction tables to connect trave -time table and non-traffic data tables, which are named with prefix “tt”, such as “tt_weather”, as shown in Figure 5.5.3.
- Travel-time data can be connected to multiple non-traffic data of same type by junction table, for example, one travel time data can be related to multiple incidents, special events or work zones.

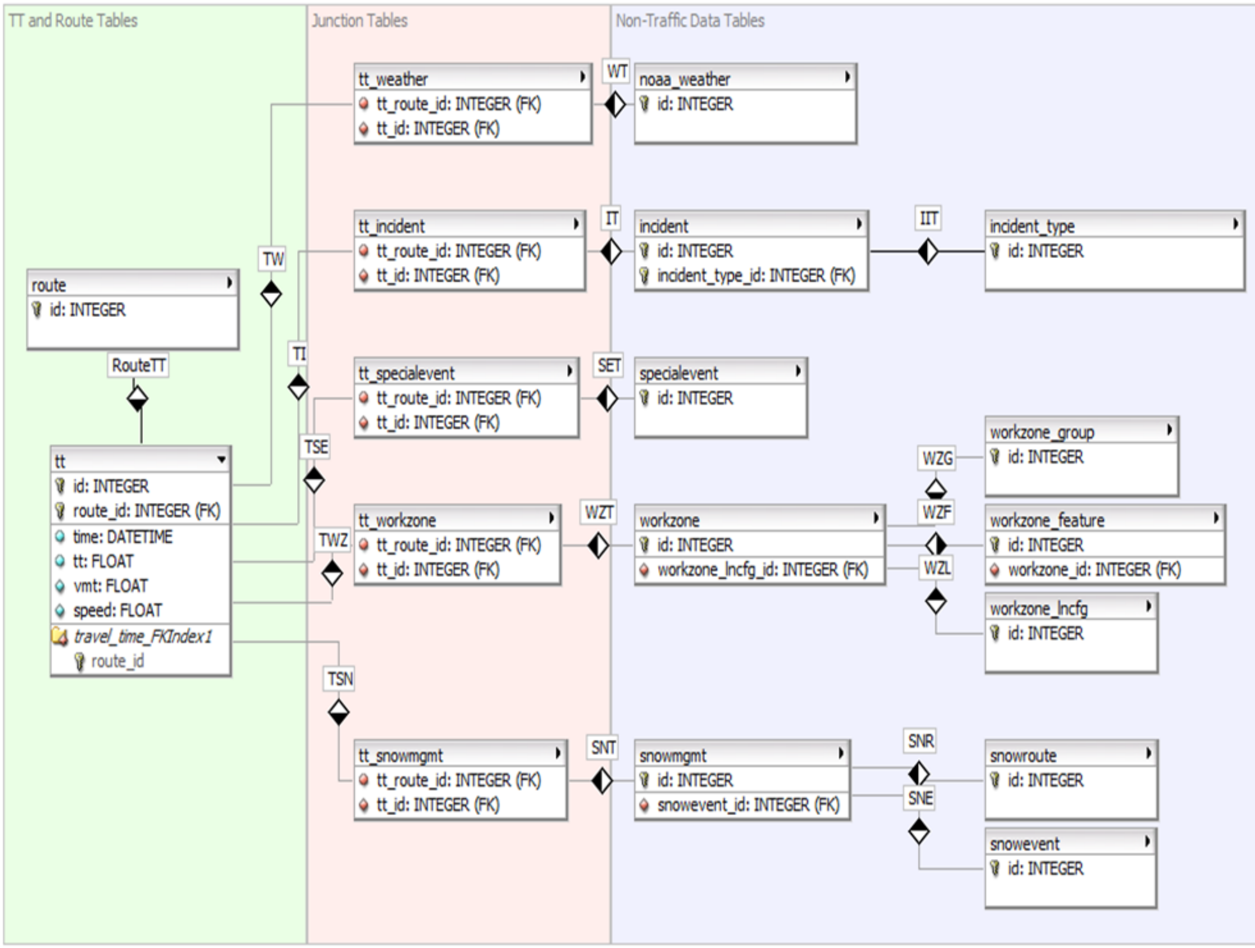


Figure 5.4.2 Relationship diagram of the database

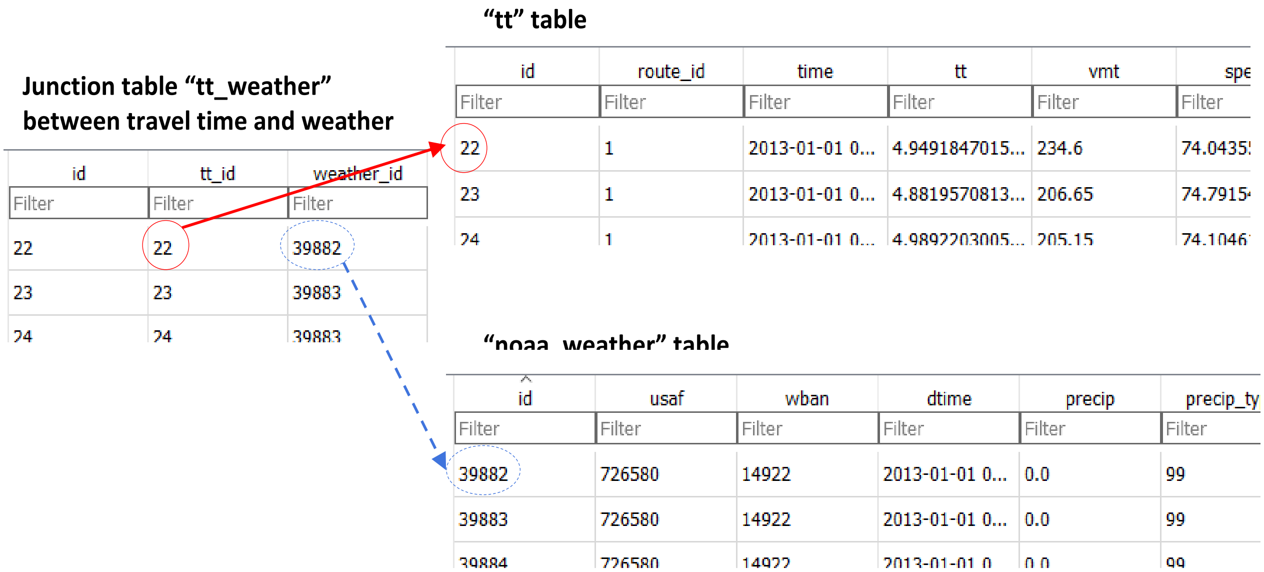


Figure 5.4.3 Example junction table between travel-time and weather tables

5.4.2 Development of Sub Modules for Categorizing Each Data Type

Each Categorization sub-module has a common 'Categorize' function, which implements the categorization of travel-time data of each route by connecting them to non-traffic data.

Weather Categorization module

Table 5.4.1 shows the weather-data categorization scheme, whose data fields include Precipitation Type and Intensity. These data fields are also shared by te imported weather data from NOAA, therefore, the *Weather Categorization* module only stores the foreign keys for travel time and weather data as shown in Table 5.4.2.

Table 5.4.1 Weather data categorization scheme

Data Field	**Values**
Precipitation Type	- SNOW, RAIN, OTHER, NONE
Precipitation Intensity	- HEAVY : Precipitation Rate > 7.6 mm(0.3 in)/hour - MODERATE : 2.5 mm(0.098 in)/hour <= Precipitation Rate < 7.6 mm(0.3 in)/hour - LIGHT : Precipitation Rate < 2.5mm(0.098 in)/hour * Glossary of Meteorology (June 2000), Rain, American Meteorological Society

Table 5.4.2 Junction-table schema between travel-time and weather tables

Table Name : tt_weather

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tt_route_id	INTEGER		NN	UNSIGNED			
tt_id	INTEGER		NN	UNSIGNED			
noaa_weather_id	INTEGER		NN	UNSIGNED			

Figure 5.4.4 shows the weather data processing steps, which can be summarized as follows:

1) Calculate center coordinates of a given route to calculate the distance to weather stations.

2) Find a nearby weather station, whose weather data exists during a given time period.

3) Repeat the following steps for all time intervals:

i) Find a weather data for a specific time interval

ii) Insert link information between travel time and weather data into the junction table.

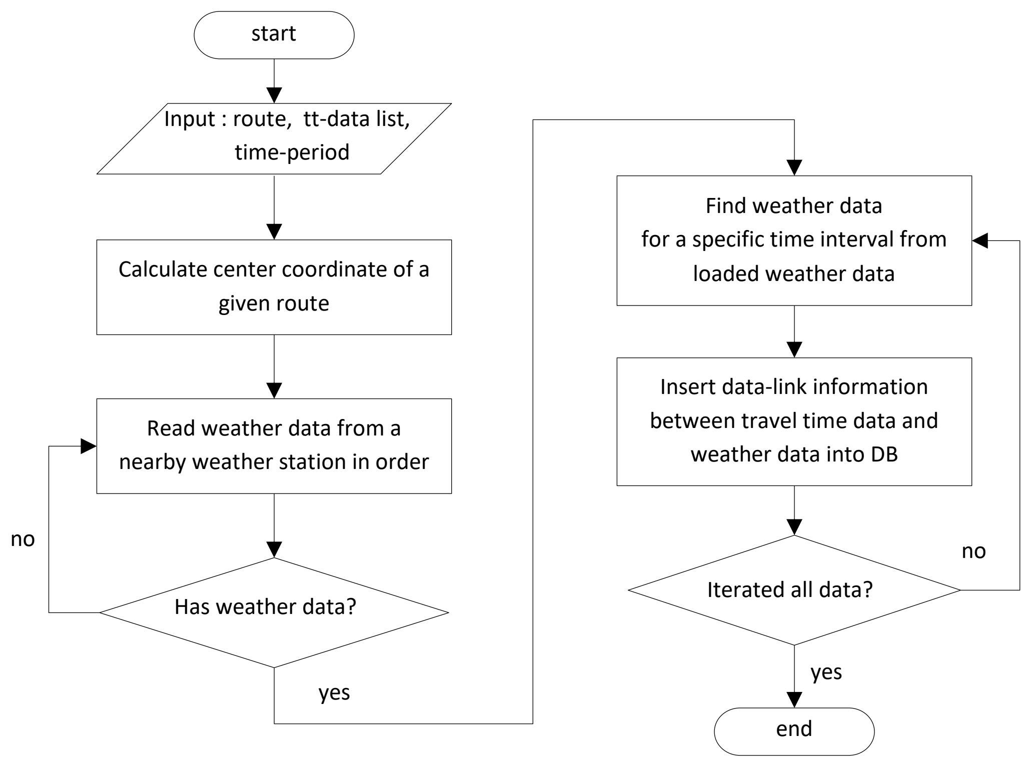


Figure 5.4.4 Flow charts of Weather Categorization module

Incident Categorization module

Table 5.4.3 shows the Incident-data categorization scheme, whose main parameters include Type, Severity, Impact and Distance. Incident type, Severity and Impact information are obtained from the incident database, which also contains additional information, such as event type and bool values named as “blocking”, “fatal”, “injury” and “rollover”, as shown in Table 5.5.3. The location of an incident in a travel –time route is represented by ‘Distance’ or ‘Offset-distance’ as shown in Figure 5.5.5. Those distance values are determined by the Weather Categorization module, which saves those calculated distance values with foreign keys into the data-table shown in Table 5.4.5.

Table 5.4.3 Incident data categorization scheme

Data Field	Values
Type	Crash, Hazard, Stall, Roadwork
Severity	Property Damage, Fatal, Injury, Other
Impact	2+ Lanes Closed, Lane Closed, Blocking, Not Blocking, Wrong Way, Run-Off Road
Distance	Distance from upstream boundary of a route
Offset-Distance	Distance from upstream or downstream boundary of route

Table 5.4.4 Junction table schema between travel time and incident tables

Table Name : tt_incident

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tt_route_id	INTEGER		NN	UNSIGNED			
tt_id	INTEGER		NN	UNSIGNED			
incident_type_id	INTEGER		NN	UNSIGNED			
incident_id	INTEGER		NN	UNSIGNED			
distance	FLOAT		NN				
offset_distance	FLOAT		NN				

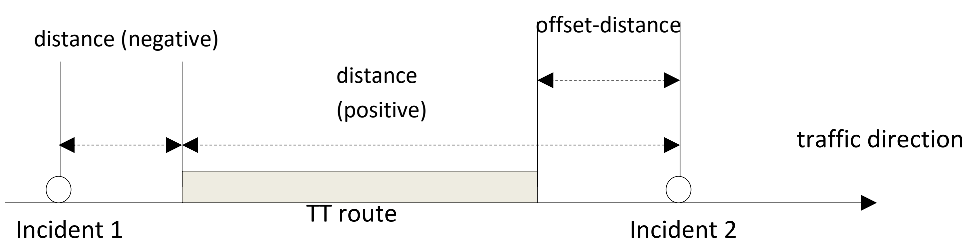


Figure 5.4.5 Distance and offset-distance in incident data categorization

Figure 5.4.6 shows the flow chart of the *Incident-Categorization* module, which follows the following steps:

1) Read all incident data from the database in a given route for a given time period in a same corridor.

2) Calculate the 'distance' and 'offset-distance' for each incident in a given route.

3) Repeat the following steps for all the time intervals:

i) Find an incident data for a specific time interval using "cdts (created)", "udts (updated)" and "xdts (closed)" field of the incident data.

ii) Insert the foreign keys between travel time and incident data into the junction table, named as tt_incident, with the calculated distances.

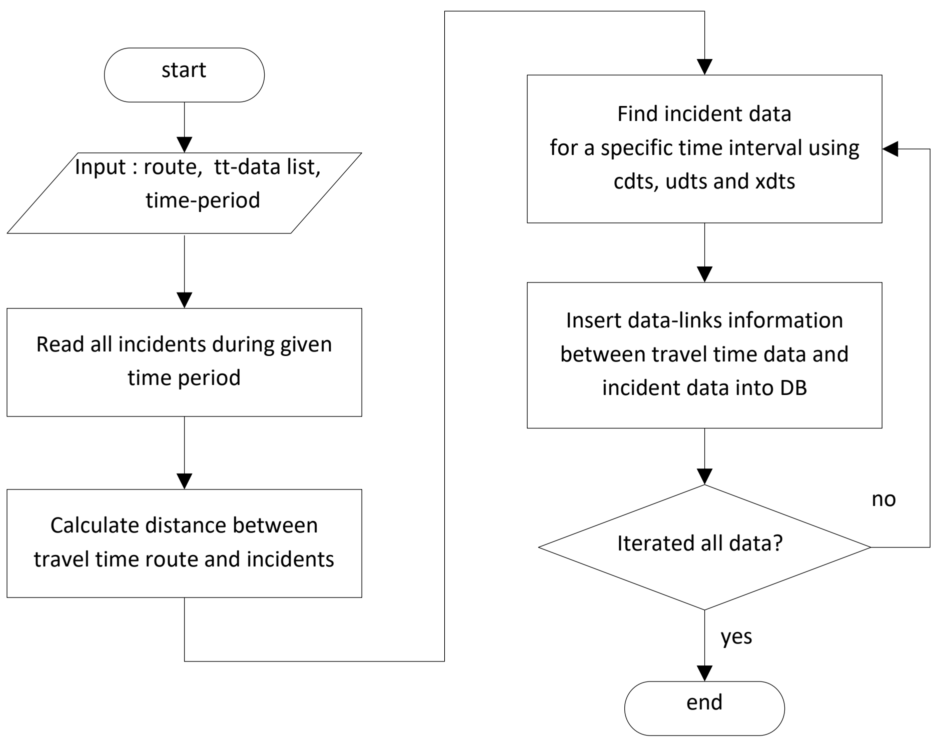


Figure 5.4.6 Flow charts of Incident Categorization module

Work-Zone Categorization module

Table 5.4.5 shows the work-zone data categorization scheme, whose data fields include Location Type, Lane configuration, Work zone characteristics, Closed-lane length, Distance and Offset Distance. The Work-zone categorization module determines Location type, Distance and Offset-Distance, while the data for other fields are collected from the database. The junction table schema to store the data connection information is included in Table 5.4.6, while Figure 5.4.7 illustrates the definitions of distances.

Table 5.4.5 Work zone data categorization scheme

Data Field	**Values**
Location Type	UP_OVERLAPPED, IN, DN_OVERLAPPED, DN, WRAP
Lane Configuration	2To0, 2To1, 2To2, 3To0, 3To1, 3To2, 3To3, 4To0, 4To1, …
Characteristics	USE_OPPOSING_LANE: when using opposing lanes USED_BY_OPPOSING_LANE: when lanes are used by the opposing traffic SHIFTED : when lanes are shifted but not closed
Length	work zone length in mile
Distance	Distance from upstream boundary of a route to upstream boundary of work zone
Offset-Distance	Distance from upstream or downstream boundary of route to upstream or downstream boundary of work zone

Table 5.4.6 Junction table schema between travel time and work zone tables

Table Name : tt_workzone

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tt_route_id	INTEGER		NN	UNSIGNED			
tt_id	INTEGER		NN	UNSIGNED			
workzone_id	INTEGER		NN	UNSIGNED			
loc_type	INTEGER		NN	UNSIGNED			
distance	FLOAT		NN				
offset_distance	FLOAT		NN				

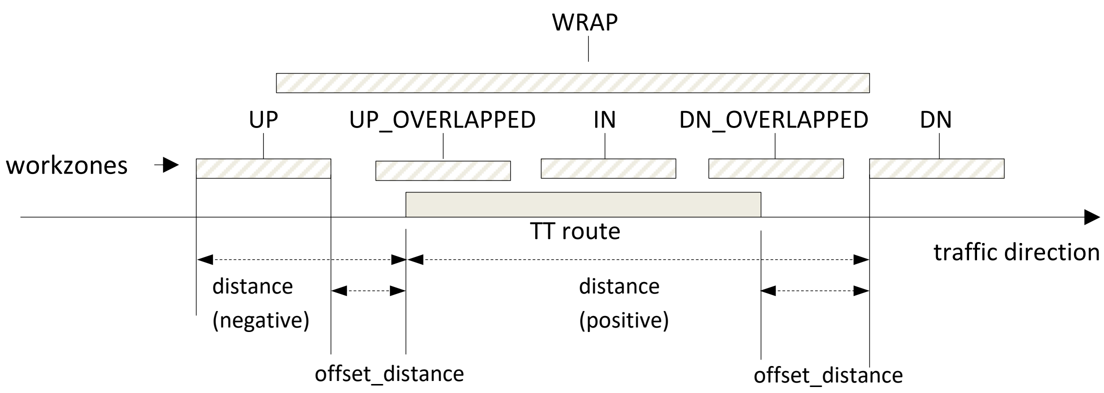


Figure 5.4.7 Location type, distance and offset-distance in work zone data categorization

The flow chart of the *Work Zone Categorization* module is shown in Figure 5.4.8, which shows the following steps:

1) Read all the work-zone data from the database for a given route and a time period in a same corridor.

2) Determine Location Type, Distance and Offset-distance for each work zone in a given route.

3) Repeat the following steps for all time intervals:

- Find a work zone data for a specific time interval.

- Insert the foreign keys into the junction table, "tt_workzone", between travel time and work zone data with the location type, distance and offset-distance determined by the Work-zone Categorization module.

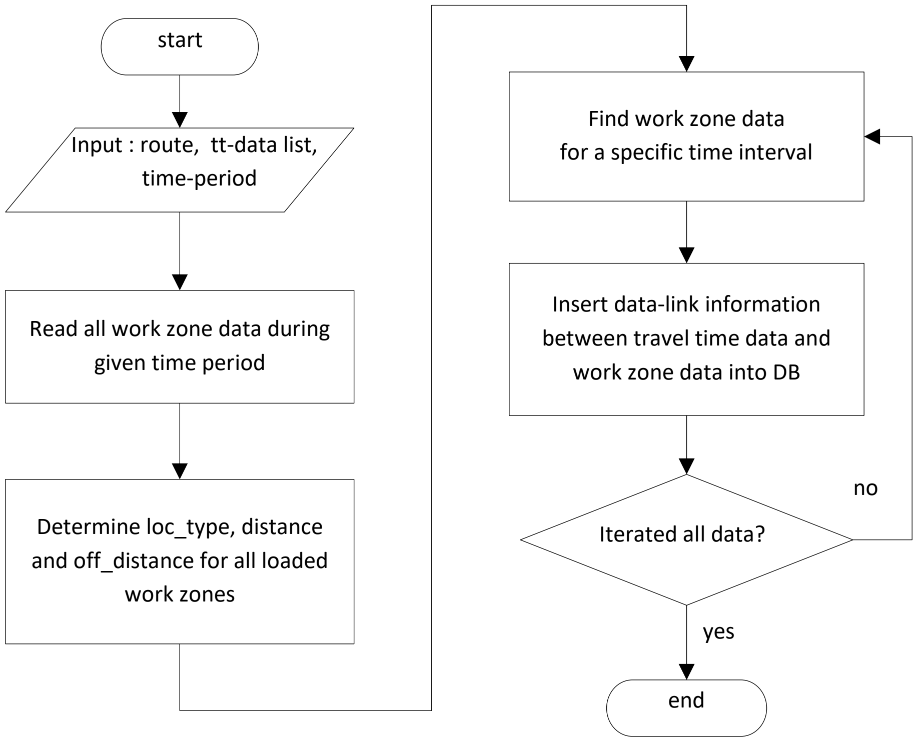


Figure 5.4.8 Flow chart of Work Zone Categorization module

Special Event Categorization module

Table 5.4.7 shows special event data categorization scheme, which uses Attendance, Distance and Event type for categorization. The Attendance data is obtained from the database, while the Distance and Event Type are determined by the Special-Event Categorization module with the following definitions:

- Distance: is the minimum distance from a given travel time route to a special event location.
- Event type: can be either "A" or "D", which indicates "Arrival" and "Departure". The Event Type is determined with the following parameters whose definitions are illustrated in Figure 5.4.9.

  - ARRIVAL_TIME_WINDOW: hours before starting event

  - DEPARTURE_TIME_WINDOW1: hours after an event starts

  - DEPARTURE_TIME_WINDOW2: lasting hours after departure starts (event start time +

DEPARTURE_TIME_WINDOW1)

The Distance and Event Type data are stored into the database table shown in Table 5.4.8, which has the foreign keys to travel time and special event data.

Table 5.4.7 Special event data categorization scheme

Data Field	Values
Attendance	Number of attendance
Distance	Line distance between special event location and TTR route
Event Type	"A" : Arrival "D" : Departure * Parameter - Arrival time window : n_a hours before starting event - Departure time window : last n_{d1} hours beginning n_{d2} hours after an event starts

Table 5.4.8 Junction table schema between travel time and work zone tables

Table Name : tt_specialevent

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tt_route_id	INTEGER		NN	UNSIGNED			
tt_id	INTEGER		NN	UNSIGNED			
specialevent_id	INTEGER		NN	UNSIGNED			
distance	FLOAT						
event_type	CHAR						

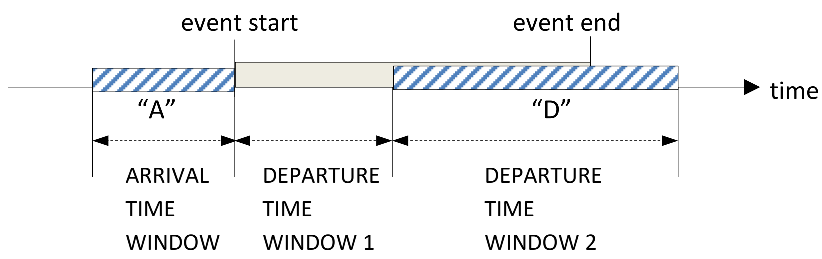


Figure 5.4.9 Arrival and Departure Time Window in special event data categorization

The flow chart of the *Special Event Categorization* module is shown in Figure 5.4.10, which has the following steps:

1) Read all special event data during a given time period from the database.

2) Calculate the distance between a given route and a special event location.

3) Repeat the following steps for all time intervals:

- Find a special event data and determine event type for a specific time interval.

- Between travel time and special event data, insert the connection information into the junction table, “tt_specialevent”, with the event type and distance.

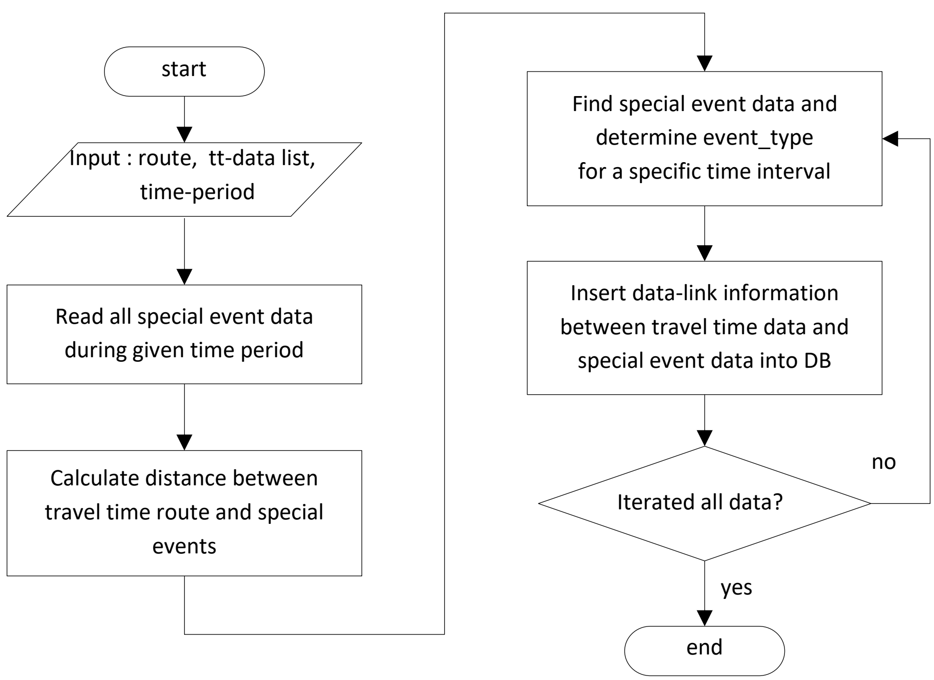


Figure 5.4.10 Flow charts of Special Event Categorization module

Snow-Management Categorization module

Table 5.4.9 shows the data-categorization scheme of the snow-management module, which uses Location type, Distance, Offset-distance and Road status for categorization. Table 5.4.10 includes the junction table with the foreign keys, which connect travel time and snow management data with the following information determined by the *Snow Management Categorization* module:

- Location Type indicates the location of a snow-plow truck route relative to a given travel-time route.
- Distance is from the upstream boundary of a travel-time route to the upstream boundary of a truck route.
- Offset-distance is from the upstream or downstream boundary of a travel time route to the upstream or downstream boundary of a snow-plow truck route.
- Road status is a binary value (0 or 1) depending on whether lane is lost or regained during snow event.

Table 5.4.9 Snow management data categorization scheme

Data Field	**Values**
Location Type	UP_OVERLAPPED, IN, DN_OVERLAPPED, DN, WRAP
Distance	Distance from upstream boundary of travel time route to upstream boundary of truck route
Offset-Distance	Distance from upstream or downstream boundary of travel time route to upstream or downstream boundary of truck route
Road Status	LOST, REGAINED

Table 5.4.10 Junction table schema between travel time and snow management tables

Table Name : tt_snowmgmt

ColumnName	DataType	PrimaryKey	NotNull	Flags	Default Value	Comment	AutoInc
tt_route_id	INTEGER		NN	UNSIGNED			
tt_id	INTEGER		NN	UNSIGNED			
snowmgmt_id	INTEGER		NN	UNSIGNED			
loc_type	INTEGER			UNSIGNED			
distance	FLOAT						
offset_distance	FLOAT						
road_status	INT						

Figure 5.4.11 shows the flow chart of the *Snow-Management Categorization* module, which has the following steps:

- Read all the snow management data during a given time period from the database.
- Determine Location type, Distance and Offset-distance for all loaded snow-management data.
- Repeat the following steps for all time intervals:

  - Find a snow-management data and determine the road status for specific time duration.

  - Insert the connection information between travel time and snow-management data in the junction table, "tt_snowmgmt", with the Location type, Distance, Off-distance and Road status.

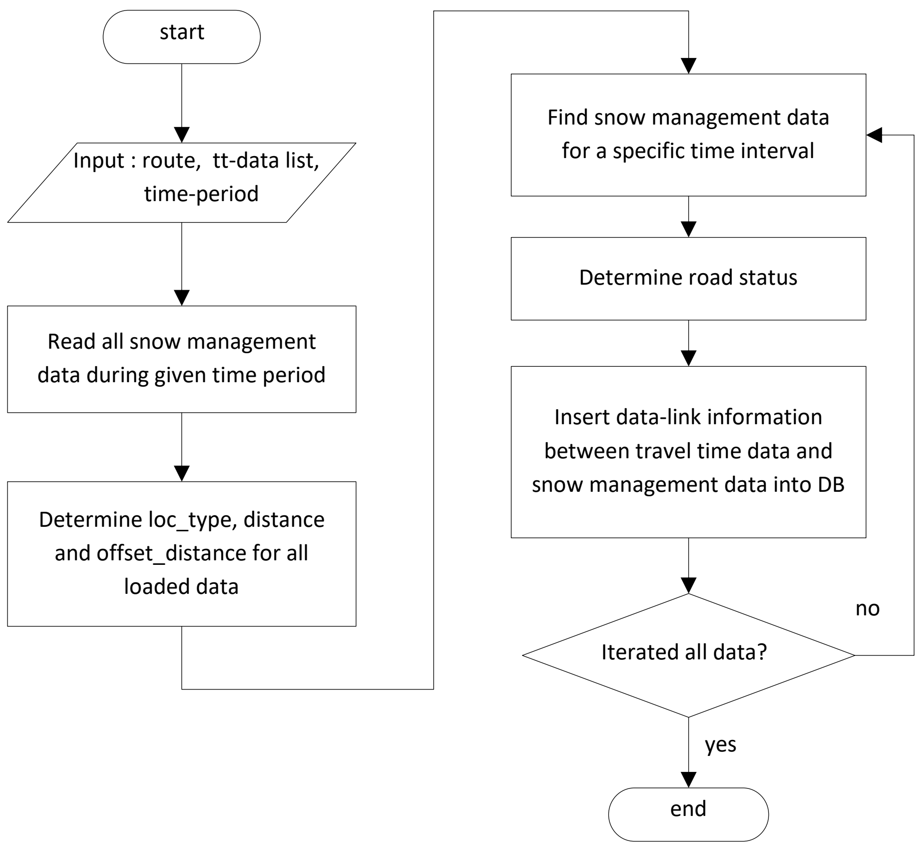


Figure 5.4.11 Flow chart of Snow-Management Categorization module

5.5 DEVELOPMENT OF RELIABILITY CALCULATION MODULE

5.5.1 Structure of Reliability Calculation module

The *Reliability Calculation* module determines various types of reliability indices using travel-time data and related non-traffic data for given routes and time periods. Figure 5.5.1 shows the simplified structure and data flow of the Reliability Calculation module that has the following sub-modules:

- *Data Extraction* module reads all the travel time and non-traffic data stored by the *Data Categorization* module for a given route and time duration. The loaded data are organized according to given filters, such as “incident-only” and/or “rainy day only” filter. Multiple data filters are acceptable so that the reliability indices under different operating conditions can be calculated.

- *Reliability* module calculates various reliability indices such as travel time index, buffer index and planning time index. Figure 5.5.2 shows the flow chart of the reliability calculation module.

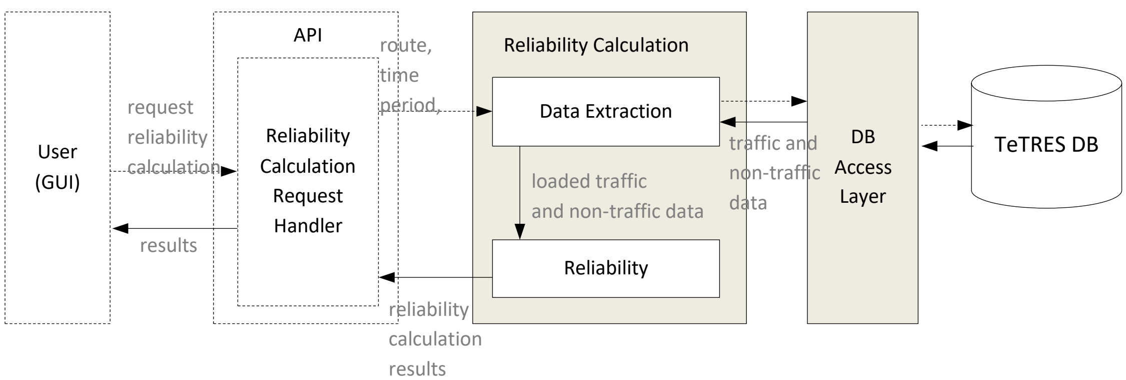


Figure 5.5.1 Structure and Data flow of the Reliability Calculation module

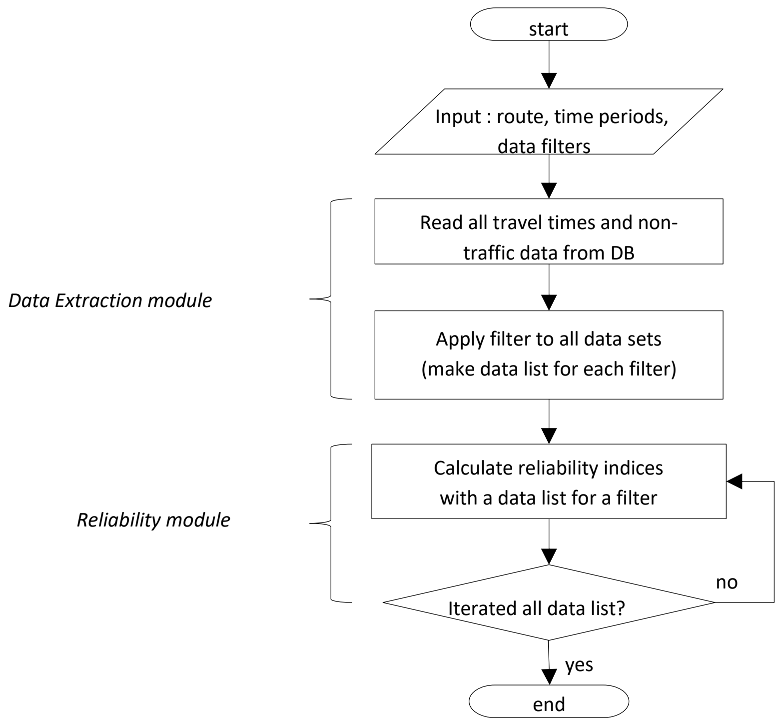


Figure 5.5.2 Flow chart of Reliability Calculation process

5.5.2 Development of Data Extraction module

To facilitate the reliability calculation process under different operating conditions, the Data Extraction module needs to have the filtering functions that can collect the data specific to given operating conditions.

Figure 5.5.3 shows a class diagram to filter the travel-time data with non-traffic, operating condition data. In this diagram,

- ExtData is the data type to contain travel time data and related non-traffic data, such as weather and incidents. One ExtData object is for travel time data at a time interval.
- IExtFilter is an interface which has a "check" function to be called by Data Extraction module.
- ExtFilter, ExtFilterGroup, And_ and Or_ are implementation of this IExtFilter interface.
- ExtFilter is a unit of filtering which is responsible for one kind of non-traffic data.
- ExtFilterGroup consists of several ExtFilters and filtered ExtData list. For example, one ExtFilterGroup can have ExtFilter for weather, ExtFilter for incident and ExtFilter for work zone.
- All filtered ExtData are put into the result list named as "results" of this class.

- And_ and OR_ are designed for logical operation among IExtFilters, which can be ExtFilter or another And_ or Or_.

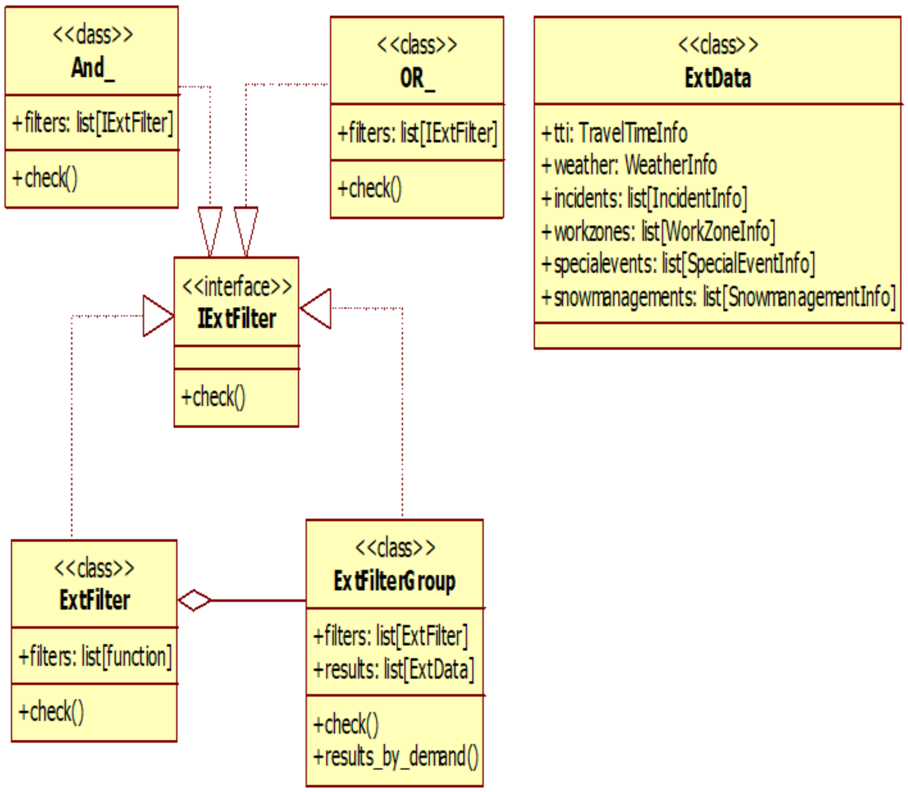


Figure 5.5.3 Class diagram for data filter

Figure 5.5.4 shows a class diagram of the filter-generation functions, which create *ExtFilter* for each non-traffic data type. Further, each module has the following functions to create 'filtered non-traffic data' described in *Data Categorization* module.

- *no_incident()* function of the *incident* module returns a filter object (instance of *ExtFilter* class) to pass only if there is no incident at the specific time interval,

- *type_rain()* function of the *weather* module returns a filter object to pass only if precipitation type of weather data is *RAIN*.

Using these functions, filter objects can be created and these filters are combined into *ExtFilterGroup* in order to collect data with respect to multiple operating conditions. Further, these filter objects can be combined into the logical operator *And_* and *Or_*

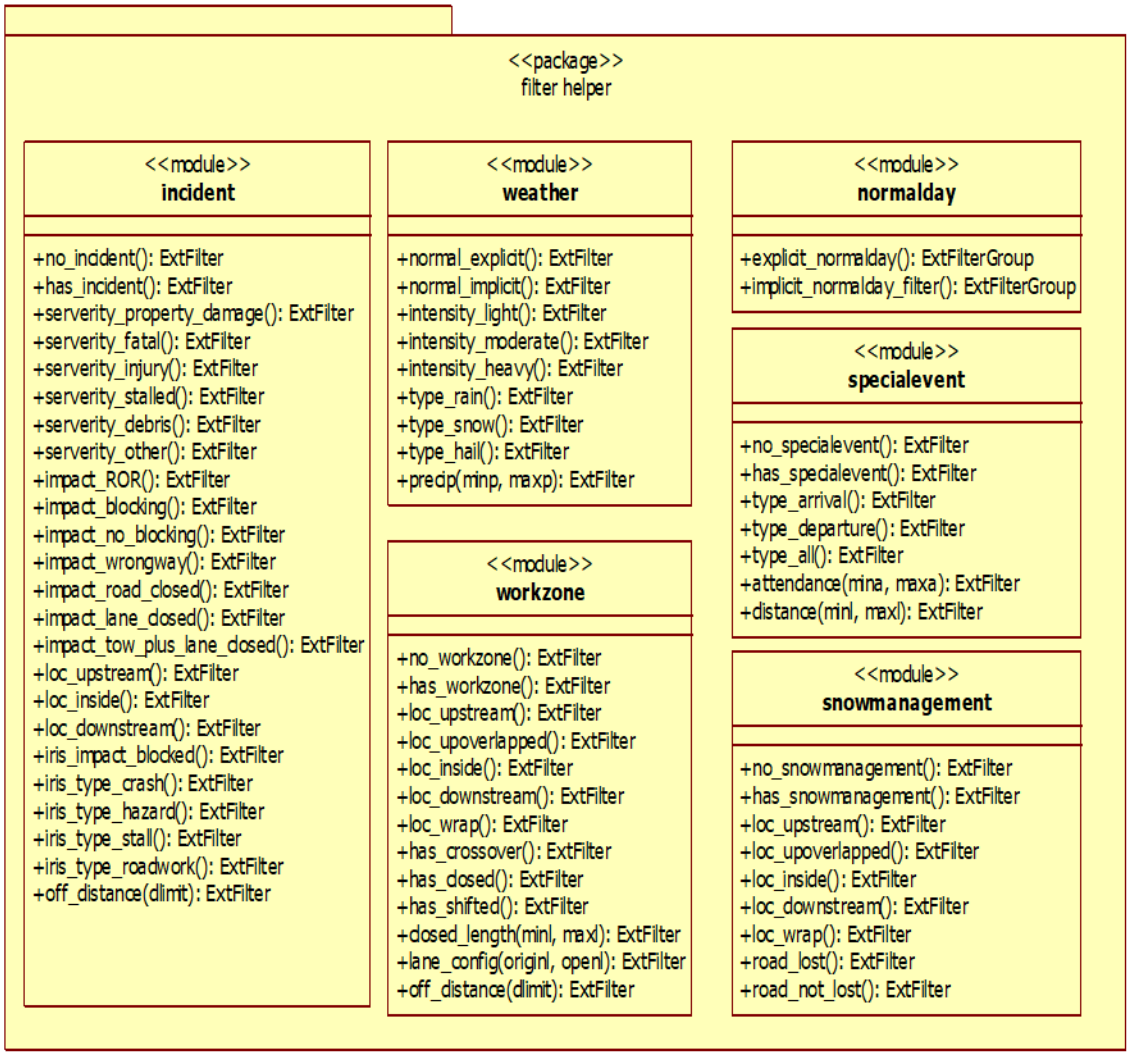


Figure 5.5.4 Class diagram of filter generation functions for each data type

5.5.3 Development of Reliability Calculation module

The *Reliability-Calculation* (RC) module calculates the travel-time reliability indices with those data collected by the Data Extraction module. The current version of the RC module developed in this research can determine the following reliability indices:

- Travel Time Index (TTI) = $\frac{TT_{avg}\ during\ congested\ hours}{TT_{Free\ Flow}}$,

where TT_{avg} = Average Travel Time, $TT_{Free\text{-}Flow}$ = Travel Time under free-flow condition.

- Planning Time Index (PTI) = $\frac{TT_{n\%}}{TT_{Free\ Flow}}$, where $TT_{n\%}$ = nth %-ile travel time.
- Buffer Index (BI) = $\frac{TT_{n\%} - TT_{avg}}{TT_{avg}}$
- Misery Index (MI) = $\frac{TT_{97.5\%}}{TT_{Free\ Flow}}$
- On-Time-Arrival = $\frac{C(TT_i > 1.5*TT_{avg}, i=1->n)}{n}$, where *C(x)* = number of occurrence of x
- Semi-Variance = $\frac{1}{n}\sum_{i=1}^{n}(TT_i - TT_{avg})^2, \exists\ TT_i > TT_{avg}$

In the above formulations, *free-flow travel time* is calculated with the posted speed limit on each route. Further, the ‘congested hours’ for TTI is defined as the time duration satisfying the following condition,

$TT_i > TT_{free\text{-}flow}$ * CH_FACTOR

where, CH_FACTOR is a user-defined parameter and currently set to 1.3. Further, PTI and BI can be calculated with 80 – 95th %-ile travel times.

Figure 5.5.5 shows the internal process of the travel-time reliability calculation module, whose data list includes both travel time and non-traffic data. First, only travel time data are extracted from the data list, which is the list of *ExtData* described in the previous section. Next, from the travel-time data list, average speed, *n*-th percentile travel times and travel times during congested hours are collected.

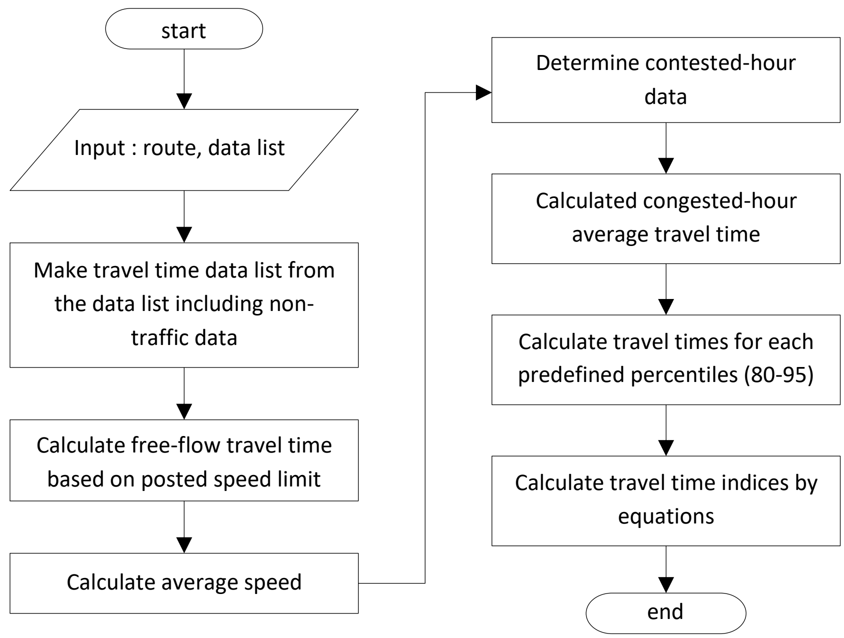


Figure 5.5.5 Flow chart of reliability calculation process

5.5.4 Example of Reliability Calculation

The reliability calculation module is tested by calculating a set of indices for an example travel-time route on I-35E NB shown in Figure 5.5.6. Figure 5.5.7 shows the source code developed in this task to calculate the travel time reliability indices. The non-traffic operating conditions tested in this example include ‘normal weather’, ‘incident’ and ‘rain or snow’. Further, the VMT-based categorization is also conducted and included in the output. Figure 5.5.8 shows the output results, which include all the calculated travel-time indices categorized for each type of operating condition as well as the VMT level.

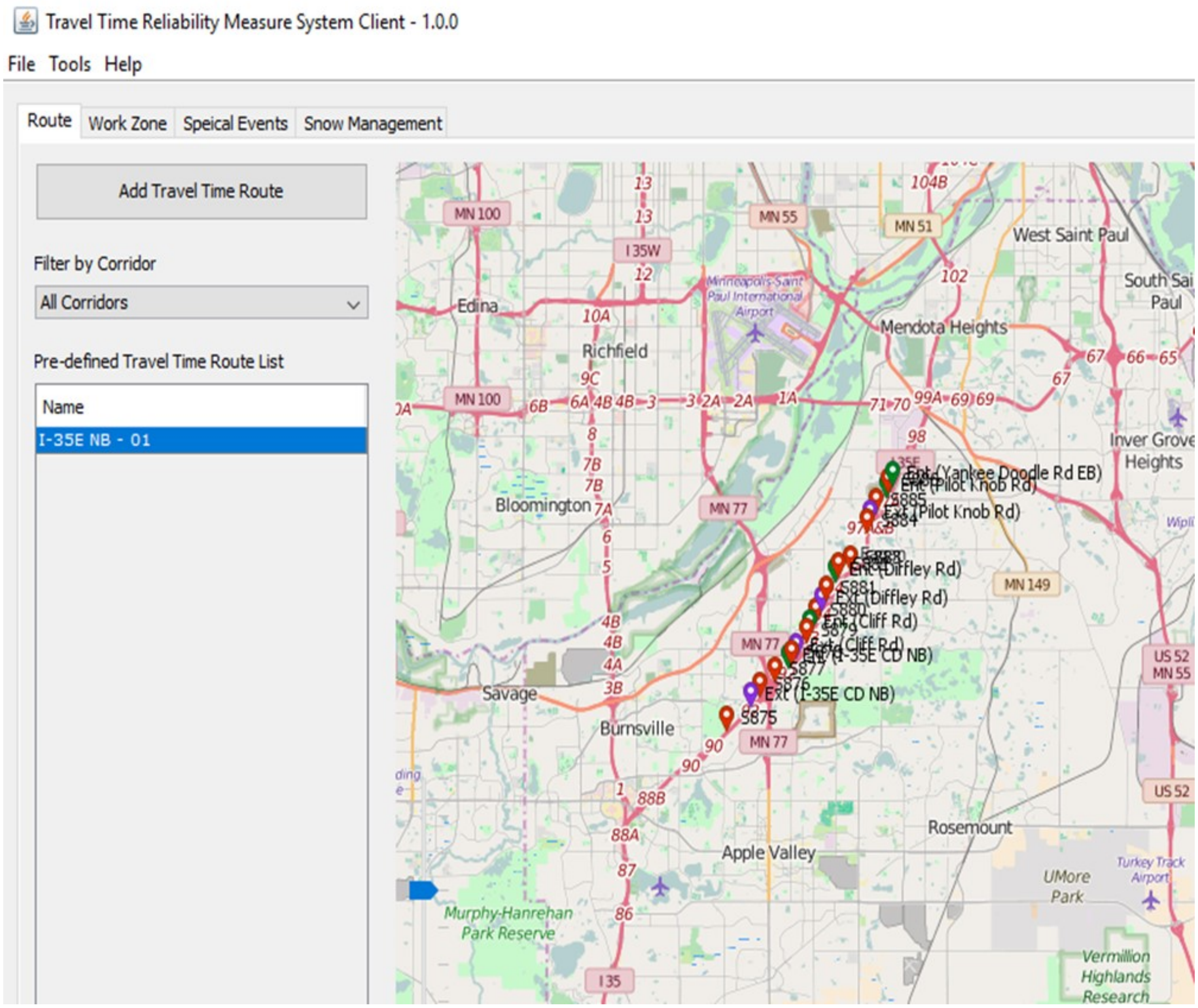


Figure 5.5.6 Example travel time route on I-35E (NB)

```
# DB Access module
da_route = route.TTRouteDataAccess()
# select the first route (to test)
ttri = da_route.list()[0]
da_route.close()
# data categorization filters
filters = [
    normalday.implicit_normalday_filter(label='Only Normal Day'),
  ExtFilterGroup([
    # no-workzone days
    workzone.no_workzone(),
    # incidents which offset_distance is less than 2 mile
    incident.has_incident(distance_limit=2),
    # no-special-event days
    specialevent.no_specialevent(),
    # normal day (not reported weather condition is considered as normal)
    weather.normal_implicit(),
  ], 'Incident'),

  ExtFilterGroup([
    Or_(
      # snow days
      weather.type_snow(),
      # or rainy days
      weather.type_rain()
    ),
  ], 'Rain or Snow'),
]
# time duration
sdate = datetime.date(2013, 1, 1)
edate = datetime.date(2013, 12, 31)
stime = datetime.time(5, 0, 0)
etime = datetime.time(11, 0, 0)
# optional parameters
target_days = [0, 1, 2, 3, 4] # Mon - Fri
remove_holiday = True
# extract all traffic and non-traffic data
# (filters are applied in this function)
extraction.extract_tt(ttri, sdate, edate, stime, etime, filters,
             target_days=target_days, remove_holiday=remove_holiday)
# each `ExtFilterGroup` has filtered `ExtData` list
for filter in filters:
  print('# ', filter.label)
  # call “Reliability” module
```

```
res = reliability.calculate(ttri, filter.results)
pprint.pprint(res)
print('  > Demand Cutlines', filter._vmt())
L, M, H = filter.results_by_demand()
print('  > Low : ')
for idx, extdata in enumerate(L):
    print('    : id=%s, time=%s, vmt=%s, tt=%s, speed=%s' % (
        extdata.tti.id, extdata.tti.time, extdata.tti.vmt, extdata.tti.tt, extdata.tti.speed))
print('  > Moderate : ')
for idx, extdata in enumerate(M):
    print('    : id=%s, time=%s, vmt=%s, tt=%s, speed=%s' % (
        extdata.tti.id, extdata.tti.time, extdata.tti.vmt, extdata.tti.tt, extdata.tti.speed))
print('  > High : ')
for idx, extdata in enumerate(H):
    print('    : id=%s, time=%s, vmt=%s, tt=%s, speed=%s' % (
        extdata.tti.id, extdata.tti.time, extdata.tti.vmt, extdata.tti.tt, extdata.tti.speed))
```

Figure 5.5.7 Example program to perform travel time reliability calculation

```
  ....
# Incident
{'avg_tt': 5.898422216373923,
 'buffer_index': {80: 0.05933045569987334,
                  85: 0.12079556770020708,
                  90: 0.30066387580050313,
                  95: 0.5970637489832032},
 'congested_avg_tt': 7.676708768762868,
 'congested_count': 91,
 'congested_hour_factor': 1.1,
 'count': 329,
 'misery_index': 2.123753990475553,
 'on_time_arrival': 0.9057750759878419,
 'on_time_arrival_count': 298,
 'planning_time_index': {80: 1.1757701091578348,
                         85: 1.2439913531116722,
                         90: 1.4436304544999148,
                         95: 1.772610056069236},
 'semi_variance': 3.7783484521146713,
 'semi_variance_count': 88,
 'travel_time_index': 1.4445419726166666,
 'tt_by_ffs': 5.314285714285722}
  > Demand Cutlines (192.27010498823893, 1375.8428571428572, 2559.4156092974754)
```

```
 > Low :
   : id=27038, time=2013-04-05 05:00:00, vmt=157.3, tt=5.644760145333022, speed=65.31044109010875
   : id=56025, time=2013-07-15 05:00:00, vmt=171.55, tt=5.531120949181454, speed=67.19233026486648
 > Moderate :
   : id=1824, time=2013-01-07 08:30:00, vmt=1463.5, tt=5.142052733644625, speed=71.58372613402621
   : id=1825, time=2013-01-07 08:35:00, vmt=1405.65, tt=5.929287093800081, speed=75.25099445497403
   : id=1826, time=2013-01-07 08:40:00, vmt=1315.6500000000008, tt=5.0676565653823396,
speed=72.40713706601464
   : id=1827, time=2013-01-07 08:45:00, vmt=1423.2999999999995, tt=5.133540132677518,
speed=71.37316550993613
   : id=1828, time=2013-01-07 08:50:00, vmt=1495.1500000000005, tt=5.32814191253344,
speed=69.52230072998147
   : id=1829, time=2013-01-07 08:55:00, vmt=1303.9999999999998, tt=5.00060983611013,
speed=73.0741272852253
   : id=1830, time=2013-01-07 09:00:00, vmt=1167.05, tt=5.830640308125936, speed=75.93197463580118
   : id=1831, time=2013-01-07 09:05:00, vmt=977.4000000000001, tt=5.9419823849146285,
speed=75.20242524950046
   : id=1832, time=2013-01-07 09:10:00, vmt=1072.2000000000003, tt=5.178616587708128,
speed=70.96455602004282
   : id=1833, time=2013-01-07 09:15:00, vmt=1033.6499999999994, tt=5.079442284365358,
speed=72.30713324913864
   : id=1834, time=2013-01-07 09:20:00, vmt=1026.5999999999997, tt=5.231895906617949,
speed=70.23874890121296
   ....
 > High :
   : id=15876, time=2013-02-25 07:35:00, vmt=2585.749999999999, tt=5.014524242197838,
speed=73.19709787462331
   : id=30228, time=2013-04-16 07:45:00, vmt=2621.8, tt=5.08946883590134, speed=71.25368634418763
   : id=38265, time=2013-05-14 07:50:00, vmt=2635.0999999999995, tt=6.777045113148593,
speed=60.572141710624805
   : id=38266, time=2013-05-14 07:55:00, vmt=2633.6, tt=6.497281606780513, speed=60.90148652210063
  ....
```

Figure 5.5.8 Output from example application

CHAPTER 6: DEVELOPMENT OF A TRAVEL-TIME INFORMATION MODULE

6.1 INTRODUCTION

This chapter develops *the Travel Time Information module (TTIM),* whose main objective is to estimate the expected travel times for predefined routes using reliability measures. Further, the connectivity of the TTIM to the existing driver-information system of MnDOT is also examined by developing an example travel-time webpage that can be used by the MnDOT system. The types and the functionalities of the major modules developed in this task are as follows:

- Travel-Time Information (TTI) Module
  - *Calculates travel-time reliability measures for each time of day (TOD) for all pre-defined routes depending on weather and dates.*
  - *Estimates expected travel times for a given route and departure time using the average travel time and TOD travel-time reliability measures*.
- Public Service API (PS-API) Module
  - *Receives and conveys the travel time information requests from the external clients, i.e., the users of MnDOT's driver-information system.*
  - *Returns the travel-time estimation results from the TTIM to the clients.*

In addition to the above modules, a web application is developed to demonstrate the travel-time information service by using an open-source chart library and external-map service.

Figure 6.1.1 shows the locations of the TTI and the PS-API modules in the overall architecture of Travel-Time Reliability Measurement System (TTRMS). The rest of this chapter describes the details of the above modules and the example travel-time webpage developed to be embedded into the existing driver-information system of MnDOT.

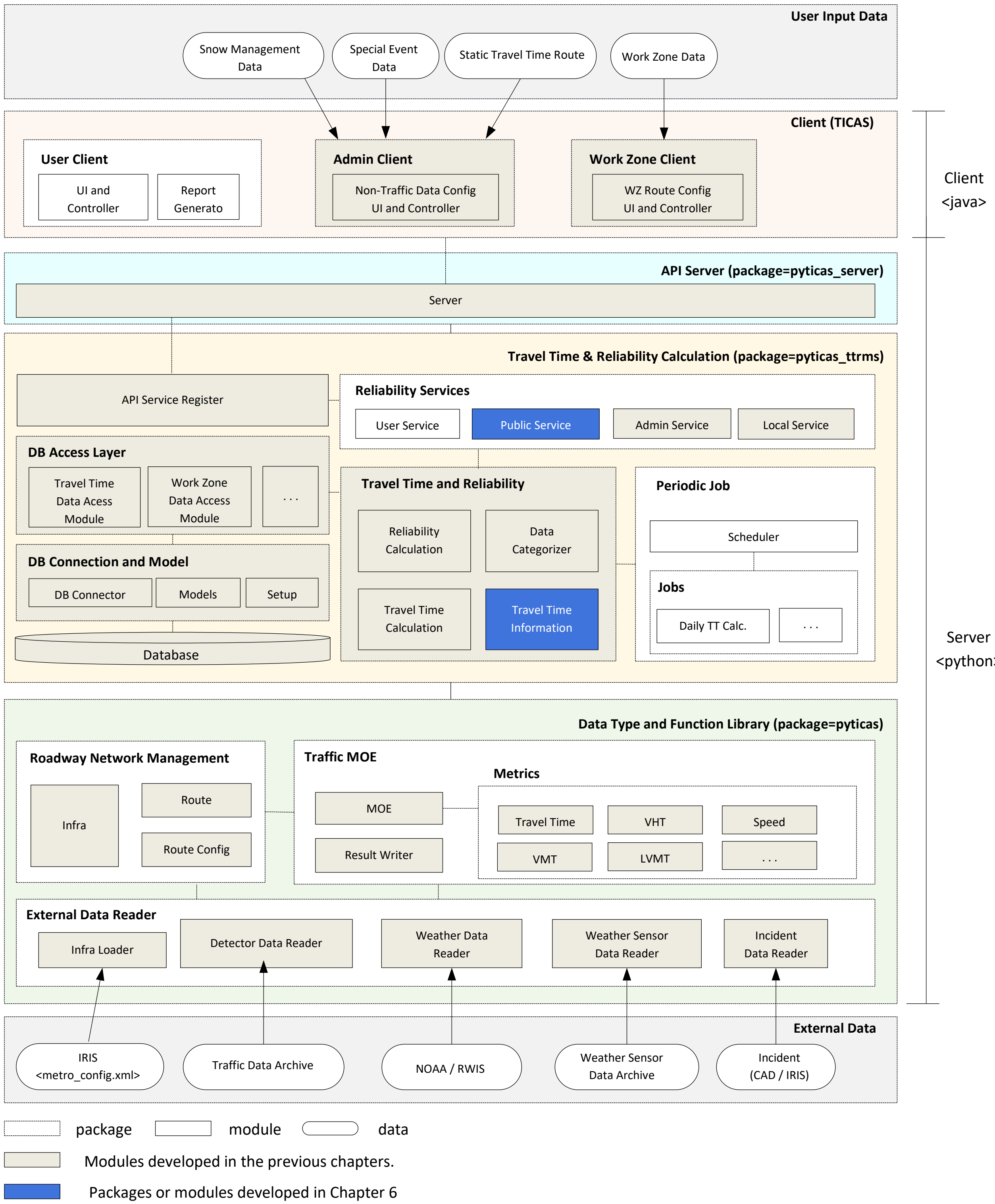


Figure 6.1.1 TTI and PS-APT Modules in the TTRMS Architecture

6.2 DEVELOPMENT OF TRAVEL-TIME INFORMATION AND PUBLIC SERVICE API MODULES

The TTI module developed in this chapter uses the travel-time data stored in the TTRMS database for each predefined route for the past year and calculates the following travel times for each time of day, i.e., every 5 minute, for each combined type of weather and date:

- Average travel time
- 95% buffer travel time
- 85% buffer travel time

Table 6.2.1 includes 12 regime types used in the travel-time calculation by the TTI module developed in this chapter.

Table 6.2.1 Regimes used in the Travel-Time Information Module

Code	Description	Code	Description
1	Dry, Monday	7	Rain, Friday
2	Dry, Tuesday-Thursday	8	Rain, Saturday-Sunday
3	Dry, Friday	9	Snow, Monday
4	Dry, Saturday-Sunday	10	Snow, Tuesday-Thursday
5	Rain, Monday	11	Snow, Friday
6	Rain, Tuesday-Thursday	12	Snow, Saturday-Sunday

Figure 6.2.1 shows the interrelationship between the TTI and the PS-API modules along with the other modules relevant to the calculation of expected travel times for given corridors and departure times. The main functionalities of each module in Figure 6.2.1 are summarized as follows:

- *The TTI* Module is responsible for the calculation of the TOD reliability measures as a pre-process and extracts the stored information in the database according to a given request.
- *The Periodic Job* module runs the function of the TOD reliability calculation in the TTI Module and the calculated data are stored into the database through the DB Access Layer.
- *The Public Travel Time Information Service,* a client of the TTI Module*,* is an external service to provide travel-time information to public. In this task, an example webpage for a client is developed to examine the feasibility of this service.
- *The API* module receives the request from a client and performs the process by calling the TTIM.
- *The Reliability Calculation* module developed in the previous task is used in the pre-process to calculate TOD reliability measures.

- *The Weather Reader* module provides weather information near a given route for a given date.
- *The MOE* module is used to produce current travel times of a given route.
- The Database stores the travel-time calculation results for each route for each regime. Table 6.2.2 shows the database table description used for the TTI Module.

Table 6.2.2 Database Table for Travel Time Information module

Table Name	Data Fields
tod_results	- route_id : route ID - regime_type : regime type, e.g. Dry-Monday, Rain-Tue./Wed./Thu, Snow-Saturday - hour : hour in time of day - minute : minute in time of day - result : JSON string containing travel time information at each time of day such as average travel time and 95percentile / 85percentile travel time for each

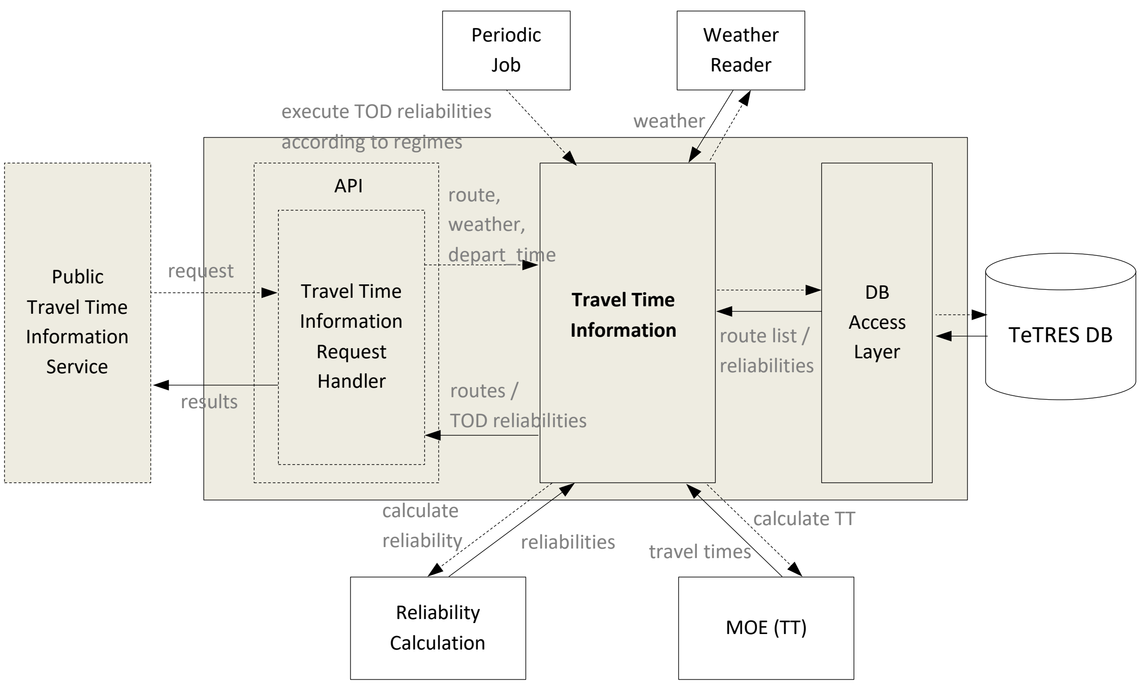


Figure 6.2.1 The Relationship between Travel-Time Information and other relevant modules

Procedure for Calculating Historical Reliability Measures

Figure 6.2.2 shows the process to calculate the reliability measures for a given predefined route. The source code of this process is included in Figure 6.2.3.

(1) Retrieve the travel-time route list from the database.

- The reliability measures are calculated for all the predefined routes using the historical data.

(2) Determine a route to process.

(3) Determine a regime to process.

(4) Read the route's travel-time data calculated by the *Periodic Job* module on a daily basis for the regime.

- Travel-time data are stored with weather information in the database.

(5) Calculate the travel-time reliability for the route by using the *Reliability Calculation* module.

(6) Save the calculated reliability measures in the database.

(7) Go to step (3) if calculation for all regimes are not completed.

(8) Got to step (2) if calculation for all routes are not completed.

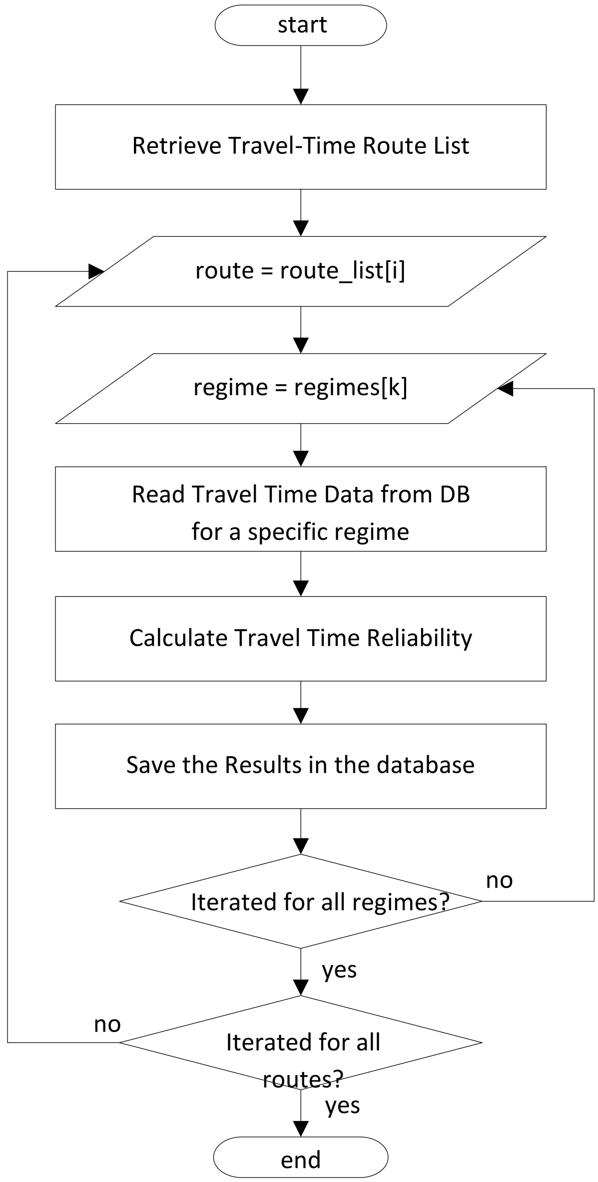


Figure 6.2.2 Process to Calculate Reliability Measures using Historical Data

```
def calculate_TOD_reliabilities(ttr_id, today):
    """
    :type ttr_id: int
    :type today: datetime.datetime
    """
    ttri = _tt_route(ttr_id)
    sdate, edate, stime, etime = _time_period(today)

    _calculate_for_a_regime(ttri, TOD_REGIME_N_0, sdate, edate, stime, etime, (0,))  # Normal, Monday
    _calculate_for_a_regime(ttri, TOD_REGIME_N_123, sdate, edate, stime, etime, (1, 2, 3))  # Normal, Tuesday-
Thursday
    _calculate_for_a_regime(ttri, TOD_REGIME_N_4, sdate, edate, stime, etime, (4,))  # Normal, Friday
    _calculate_for_a_regime(ttri, TOD_REGIME_N_56, sdate, edate, stime, etime, (5, 6))  # Normal, Saturday-
Sunday
    _calculate_for_a_regime(ttri, TOD_REGIME_R_0, sdate, edate, stime, etime, (0,))  # Rain, Monday
    _calculate_for_a_regime(ttri, TOD_REGIME_R_123, sdate, edate, stime, etime, (1, 2, 3))  # Rain, Tuesday-
Thursday
    _calculate_for_a_regime(ttri, TOD_REGIME_R_4, sdate, edate, stime, etime, (4,))  # Rain, Friday
    _calculate_for_a_regime(ttri, TOD_REGIME_R_56, sdate, edate, stime, etime, (5, 6))  # Rain, Saturday-Sunday
    _calculate_for_a_regime(ttri, TOD_REGIME_S_0, sdate, edate, stime, etime, (0,))  # Snow, Monday
    _calculate_for_a_regime(ttri, TOD_REGIME_S_123, sdate, edate, stime, etime, (1, 2, 3))  # Snow, Tuesday-
Thursday
    _calculate_for_a_regime(ttri, TOD_REGIME_S_4, sdate, edate, stime, etime, (4,))  # Snow, Friday
    _calculate_for_a_regime(ttri, TOD_REGIME_S_56, sdate, edate, stime, etime, (5, 6))  # Snow, Saturday-
Sunday
def _calculate_for_a_regime(ttri, regime_type, sdate, edate, stime, etime,
                            target_days=(1, 2, 3), except_dates=(), remove_holiday=True):
    """
    :type ttri: pyticas_ttrms.ttrms_types.TTRouteInfo
    :type regime_type: int
    :type sdate: datetime.date
    :type edate: datetime.date
    :type stime: datetime.time
    :type etime: datetime.time
    :type target_days: tuple[int]
    """
    # Regime Filter
    ext_filter = _ext_filter(regime_type)

    # Retrieve travel time data for a regime from DB
    extractor.extract_tt(ttri, sdate, edate, stime, etime, [ext_filter],
                         target_days=target_days,
                         remove_holiday=remove_holiday,
```

```
                except_dates=except_dates)
# create DB Access module instance
da = TODReliabilityDataAccess()

# delete existings
for ttwi in da.list_by_route(ttri.id, regime_type):
    da.delete(ttwi.id, autocommit=False)
da.commit()

# iterate for time of day
cursor = datetime.datetime.combine(datetime.date.today(), stime) # indicator of TOD
cursor += datetime.timedelta(seconds=cfg.TT_DATA_INTERVAL)
edatetime = datetime.datetime.combine(datetime.date.today(), etime)
while cursor <= edatetime:
    ctime = cursor.strftime('%H:%M:00')
    # collect data for a regime
    res = [ extdata for extdata in ext_filter.results if ctime in extdata.tti.time ]
    # calculate reliabilities
    ttr_res = reliability.calculate(ttri, res)
    # put the result into DB
    todri = TODReliabilityInfo()
    todri.regime_type = regime_type
    todri.route_id = ttri.id
    todri.hour = cursor.hour
    todri.minute = cursor.minute
    todri.result = json.dumps(ttr_res)
    da.insert(todri, autocommit=True)
    # move the cursor
    cursor += datetime.timedelta(seconds=cfg.TT_DATA_INTERVAL)

da.close()
```

Figure 6.2.3 Source code for Reliability Measure Calculation using Historical Data

Procedure for Travel-Time Information Service

Figure 6.2.4 shows the sequence diagram of the travel-time information service, which provides estimated travel times to public for given routes and departure times. Figure 6.2.5 includes the source code of the information service function in the TTI Module. The process can be summarized as follows:

(1) User accesses the public travel-time information (TTI) service, which requests predefined travel-time route list.

(2) The API in the server receives the request and produces the results by calling the corresponding function in the TTI Module. The TTI Module retrieves the travel-time route list from the database.

(3) User selects a travel-time route and departure time, then requests travel-time information.

(4) The API module receives the request from the TTI service and calls the corresponding function in the TTI Module, which performs the following steps:

- Calculates travel times for current time.
- Get current weather information through the *Weather Data Reader* module
- Retrieves the travel-time reliability measures for a given route and regime from the database.
- The reliability measures are sent back to the TTI service in a Jason format.

(5) The TTI Service displays the reliability measures.

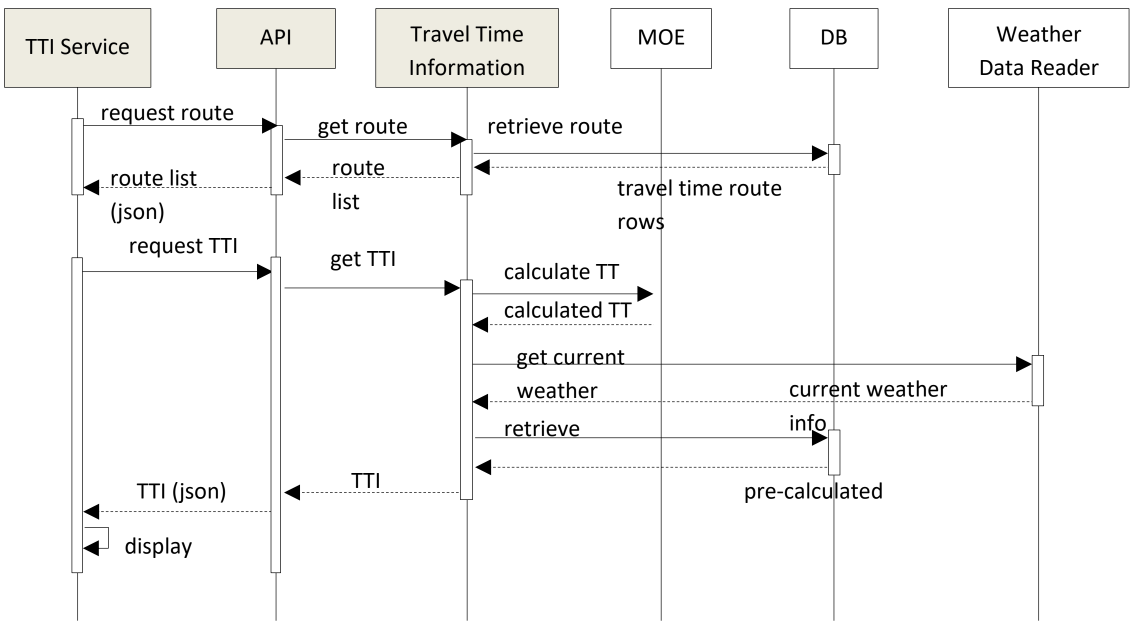


Figure 6.2.4 A Sequence diagram of travel-time information service process

```
def traveltime_info(ttr_id, weather_type, depart_time):
    """
    :type ttr_id: int
    :type weather_type: int
    :type depart_time: datetime.datetime
    :rtype: list[dict], list[float]
    """
    # create DB Access module instance for travel time route
```

```
ttrda = TTRouteDataAccess()
ttri = ttrda.get_by_id(ttr_id)
ttrda.close()

# get weather information
if not weather_type or weather_type not in [WC_NORMAL, WC_RAIN, WC_SNOW]:
    weather_type = _weather(depart_time, ttri.route)

# decide regime according to weather and departure time
regime_type = _regime_type(weather_type, depart_time)

# create DB Access module instance for travel time reliability
da = TODReliabilityDataAccess()

# retrieve reliability data from DB and pack to list
tods = da.list_by_route(ttr_id, regime_type)
res = []
for idx, tod in enumerate(tods):
    tod_res = json.loads(tod.result)
    res.append({'hour' : tod.hour, 'minute' : tod.minute, 'avg_tt' : _roundup(tod_res['avg_tt']),
            'p95_tt' : _roundup(tod_res['percentile_tts']['95']),
           'p90_tt' : _roundup(tod_res['percentile_tts']['90']),
           'p85_tt' : _roundup(tod_res['percentile_tts']['85']),
           'p80_tt' : _roundup(tod_res['percentile_tts']['80']),
            'count' : tod_res['count']
            })

# calculate travel time until the current time
today_to = depart_time
now = datetime.datetime.now()
if today_to >= now:
    today_to = now

today_from = datetime.datetime.combine(today_to.date(), datetime.time(0, 0, 0))
prd = period.Period(today_from, today_to, cfg.TT_DATA_INTERVAL)
tts = moe.travel_time(ttri.route, prd)
tts = moe.imputation(tts, imp_module=time_avg)
traveltimes = data_util.moving_average(tts[-1].data, 5)
traveltimes = _roundup(traveltimes)

return res[60:-12], traveltimes[60:-12]
```

Figure 6.2.5 Source code of the travel-time information service function in the TTI Module

6.3 DEVELOPMENT OF AN EXAMPLE WEBPAGE FOR MNDOT DRIVER-INFORMATION SYSTEM

Figure 6.3.1 shows the example webpage developed in this chapter to examine the connectivity of the TTI Module to the existing driver-information system in MnDOT. The example travel-time webpage is developed as a single web application, so that it can be embedded efficiently into the existing web site of the MnDOT driver-information system. The process to obtain expected travel times for a route is as follows:

1) User can select a route and specify expected departure time using the combo box.

2) The selected travel time route is shown in the map.

3) The expected travel times for the selected route, including average, 95th and 85th %-ile travel times, are displayed in the screen in both graphical and text formats. The travel time of current day and time when the request was made for a selected route is also displayed in the travel-time graph along with the reliability-based, expected travel times.

Figure 6.3.2 shows the results from an example application of the TTI Module with a route on the 35E NB corridor from the split point to the 494 interchange. Figure 6.3.2a is for a day under dry weather condition, while Figure 6.3.2b shows the expected travel times for a snow day in November 2017. The estimation results for a route on the I-35W NB for dry and snow days in November 2017 are shown in Figures 6.3.3a and 6.3.3b.

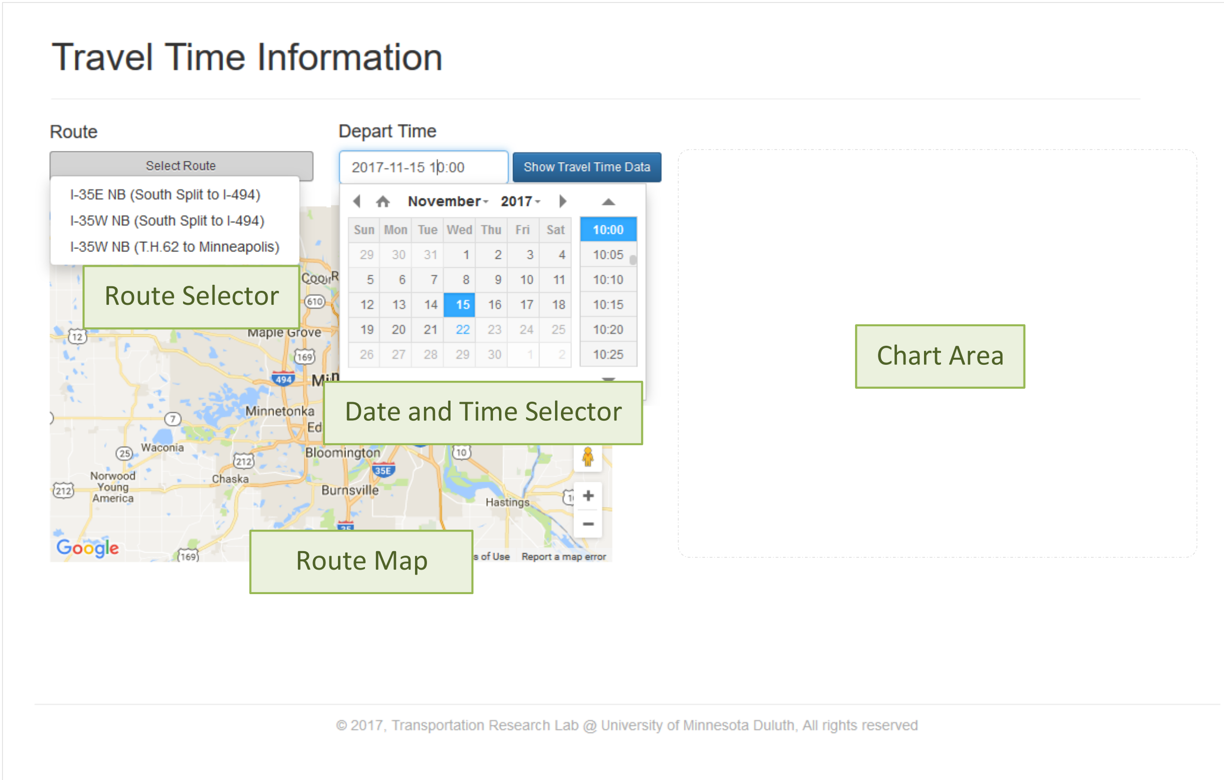
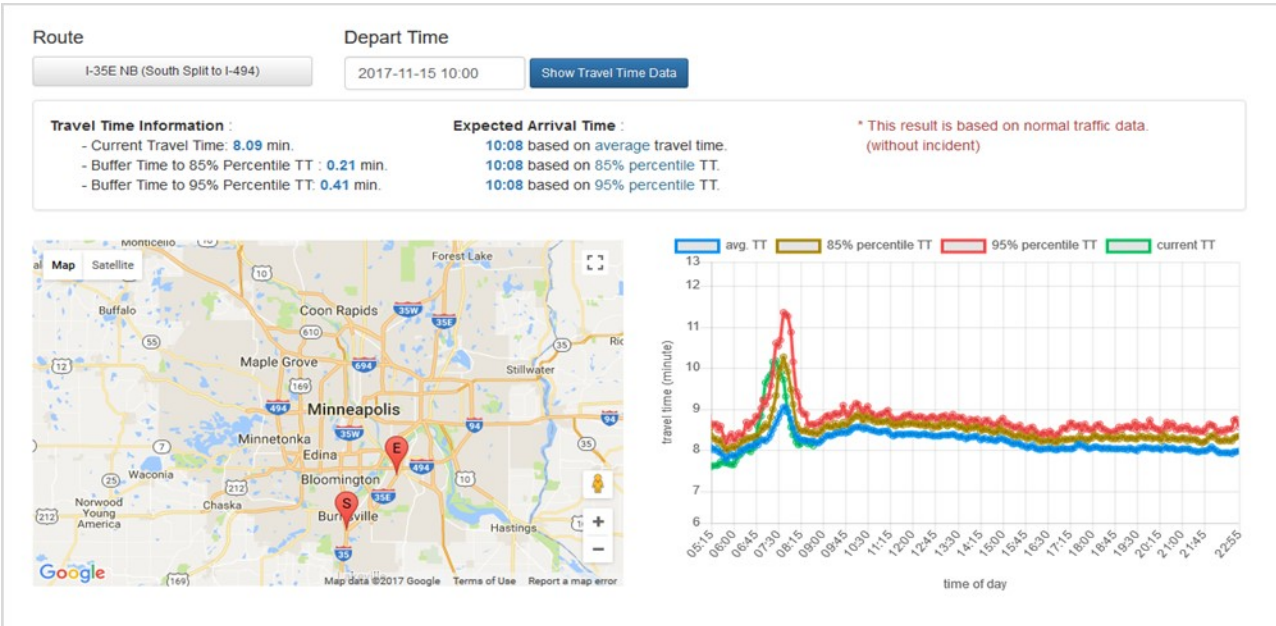
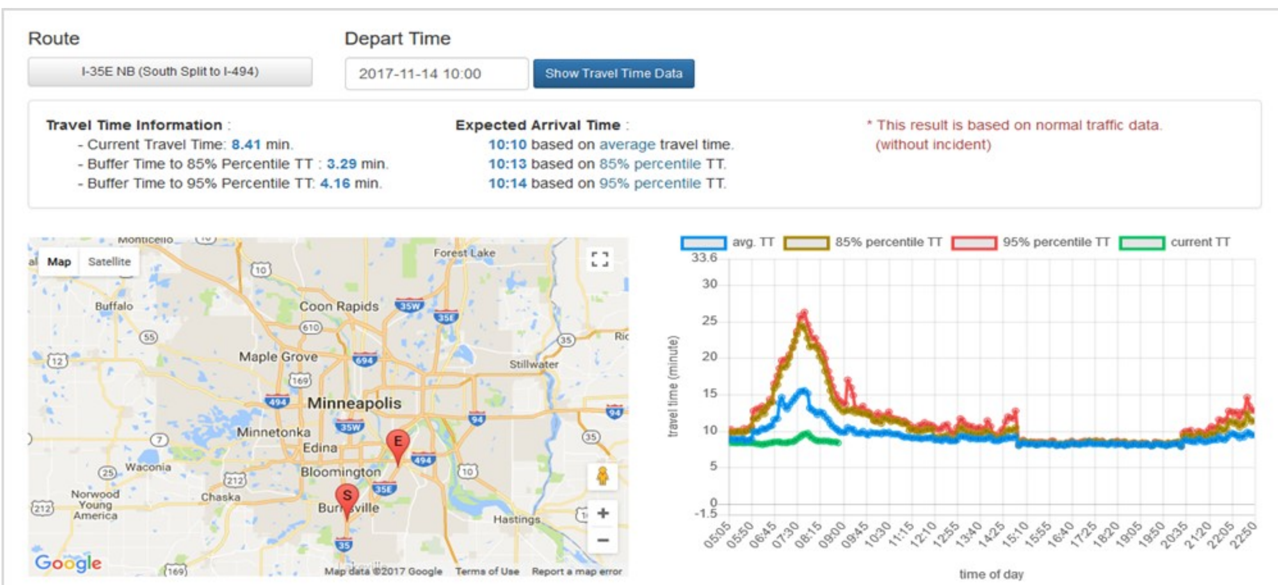


Figure 6.3.1 An Example web page for travel-time information service

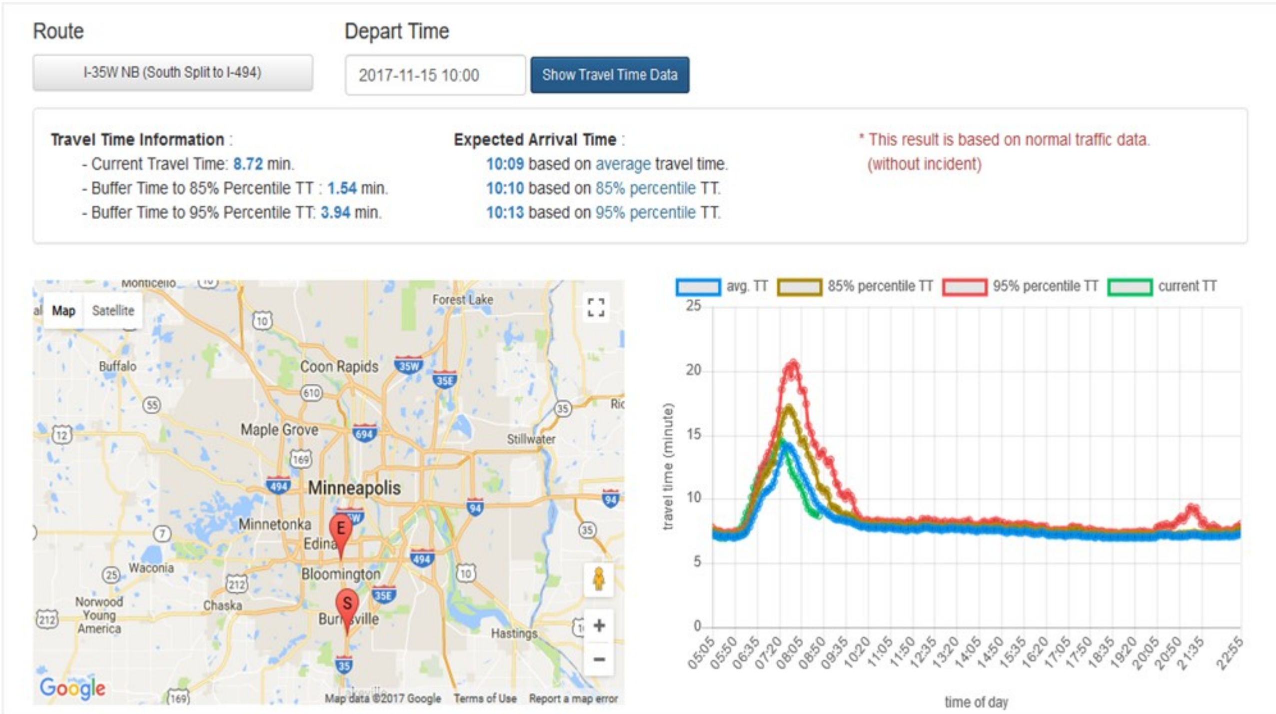


(a) Travel-time estimation results for a route on I-35E NB on a dry day

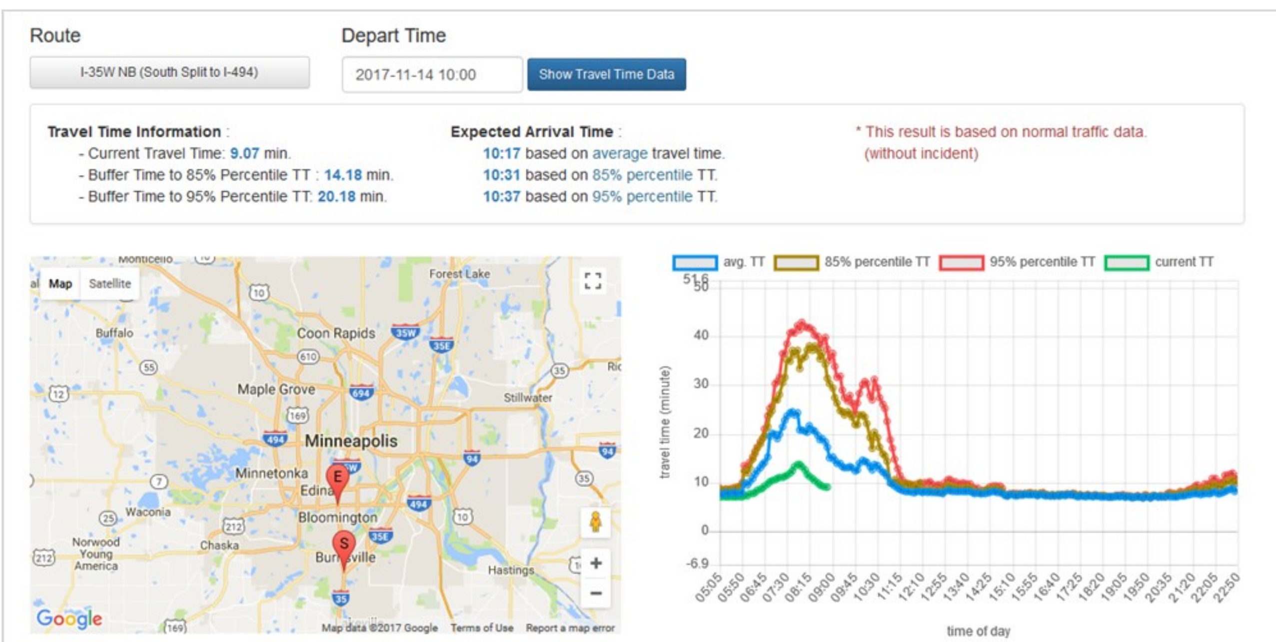


(b) Travel-time estimation results for a route on the I-35E NB during a snow day

Figure 6.3.2 Example application results of the Travel-Time Information Module for a route on I-35E (NB)



(a) Travel-time estimation results for a route on the I-35W NB for a dry day



(b) Travel-time estimation results for a route on I-35W NB during a snow day

Figure 6.3.3 Example Application Results of the Travel-Time Information Module for a route on I-35W (NB)

CHAPTER 7: DEVELOPMENT OF THE USER-INTERFACE AND REPORT-GENERATION MODULE

7.1 Introduction

In this chapter, the *User-Interface and Report-Generation Module* is developed to facilitate the input and output processes of TTRMS. Using the User-Interface, the user can specify a set of freeway routes, time periods and specific operating conditions for reliability estimation. Further, the report-generation module produces reliability measures for selected corridors following user-specified format. Figure 7.1.1 shows the relationships between the new modules developed in this chapter and the other modules in the *Travel-Time Reliability Measurement System (TTRMS)*. The major functionalities of the new modules are as follows:

- User- Interface Module:
  - Manages of the identification and grouping process for the routes whose reliability measures need to be estimated,
  - Manages of the configuration process to specify operating conditions for reliability estimation, such as weather, incident, work-zone and special events,
  - Manages the input process of the user-specified data necessary to estimate reliability measures, such as selection of a route or route groups for reliability estimation, time-periods, and operating conditions.
- User-Service Module:
  - Facilitates reliability estimation process by providing user-specified input parameters to the Reliability Estimation module.
- Reliability-Estimation Module:
  - *Creates a set of the operating-condition filters using user-specified operating conditions from the Client,*
  - *Retrieves the travel-time data for specified routes from the database using the filter functions and time-duration data,*
  - *Conducts the reliability-measures estimation process using retrieved travel time data and the reliability calculation module developed in Task 4.*
- Report-Writing Module:

- *Generates a set of spread-sheet files with the estimated reliability measures and travel-time data for user-specified routes,*
- *Generates a set of the graphs with the reliability measures for user-specified operating condition for given routes.*

The rest of this chapter describes the details of the above modules.

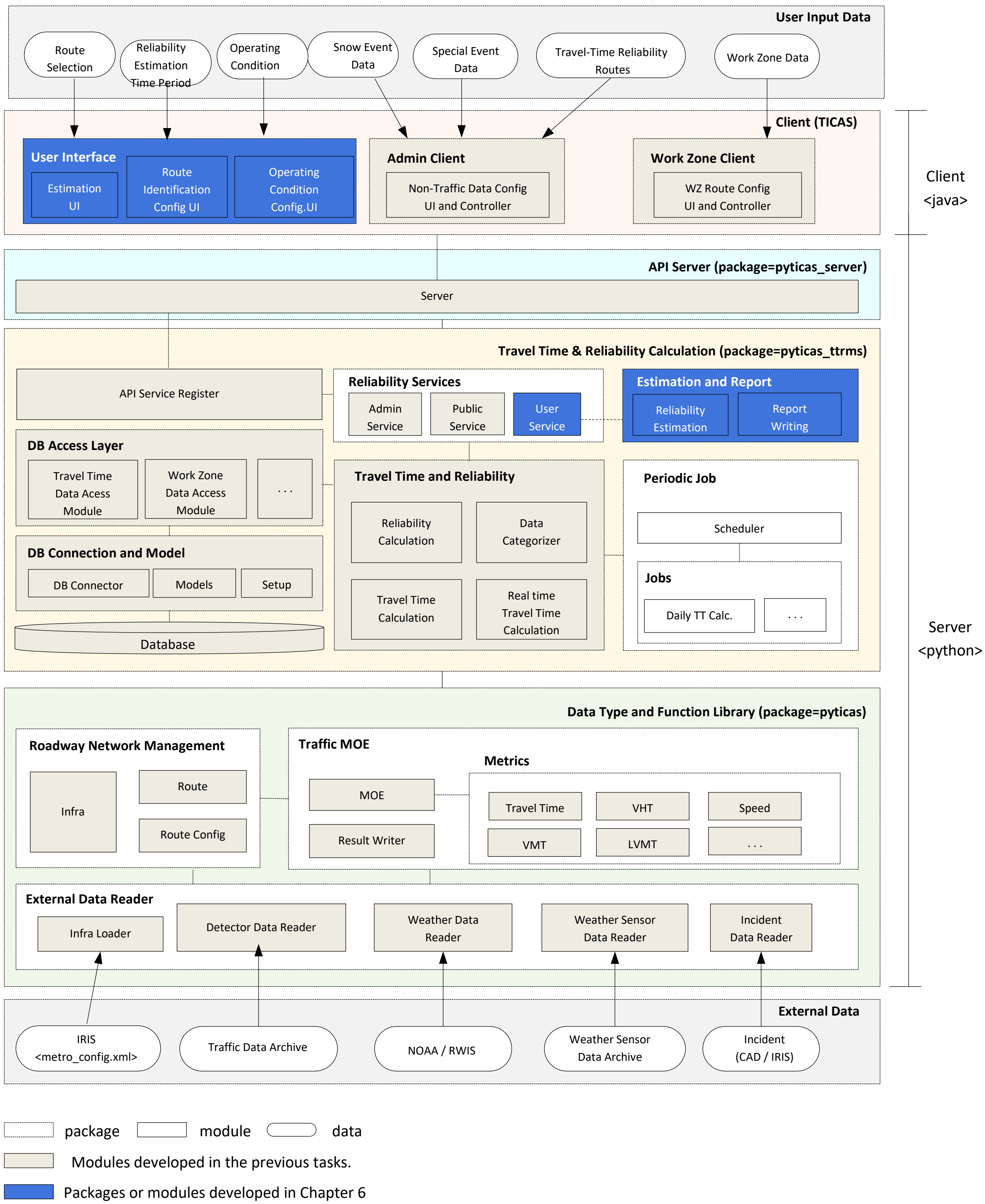


Figure 7.1.1 Architecture of TTRMS and the User-Interface Modules

7.2 DEVELOPMENT OF THE USER-INTERFACE AND USER-SERVICE MODULES

7.2.1 Overview of the User-Interface and User-Service modules

Figure 7.2.1 shows the simplified structure of the User-Interface and User-Service modules and their interrelationships. The User-Interface, written in Java, has three submodules:

- *Reliability Estimation Panel* receives user-specified input parameters needed to run the reliability estimation process, such as reliability-route selection, time periods, and operating conditions.

- *Route Identification-Configuration Panel* manages the selection process of a route or route-group whose reliability would be estimated.

- *Operating Condition-Configuration Pane* manages the configuration process of the specific operating conditions for reliability estimation, e.g. weather, incident and work zone conditions.

The user-specified data through the user interface are serialized to JSON string and delivered to the *User Service* module using HTTP by the API client modules as shown in Figure 7.2.1.

The *User service* module receives the JSON strings from the User-Interface and converts them to the Python objects to be used by other modules in TTRMS. The main functions of the two submodules in the User-Service module are as follows:

- *Reliability Estimation Handler* performs the reliability estimation process for user-specified conditions,

- *Travel Time Route Data Handler* provides travel-time route list for given corridors.

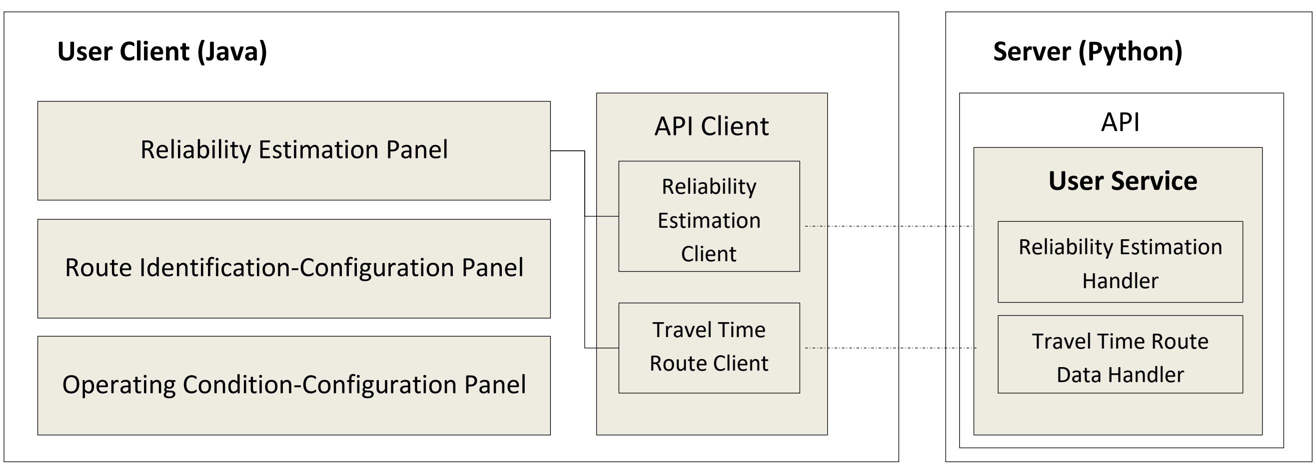


Figure 7.2.1 Overview of User-Client and User-Service module structure

7.2.2 Development of the User-Interface module

Development of the Reliability-Estimation Panel

Figure 7.2.2 shows the Reliability-Estimation Panel, which is the main user interface for entering a set of the input parameters required to estimate reliability measures. The types of the parameters that can be entered through this panel are as follows:

- Travel time routes: the pre-defined travel time routes in the *Route Identification Configuration Panel* or a single route can be selected for reliability estimation.
- Date and time information: start/end dates, time period, week day.
- Type of Reliability to be estimated:

  - Reliability for Whole-Time-Period (WTP Reliability)

    :Reliability measures calculated with all the travel time data during a given time period including Yearly, Monthly and Daily reliability measures.

  - Time of Day Reliability (TOD Reliability)

    :Reliability measures calculated for each time interval for a given time period, e.g., Reliability at 5:00, 5:05, etc., are calculated.

- Operating Conditions: specific operating conditions under which reliability measures are calculated for given corridors, e.g., weather, existence of incidents and work zones.

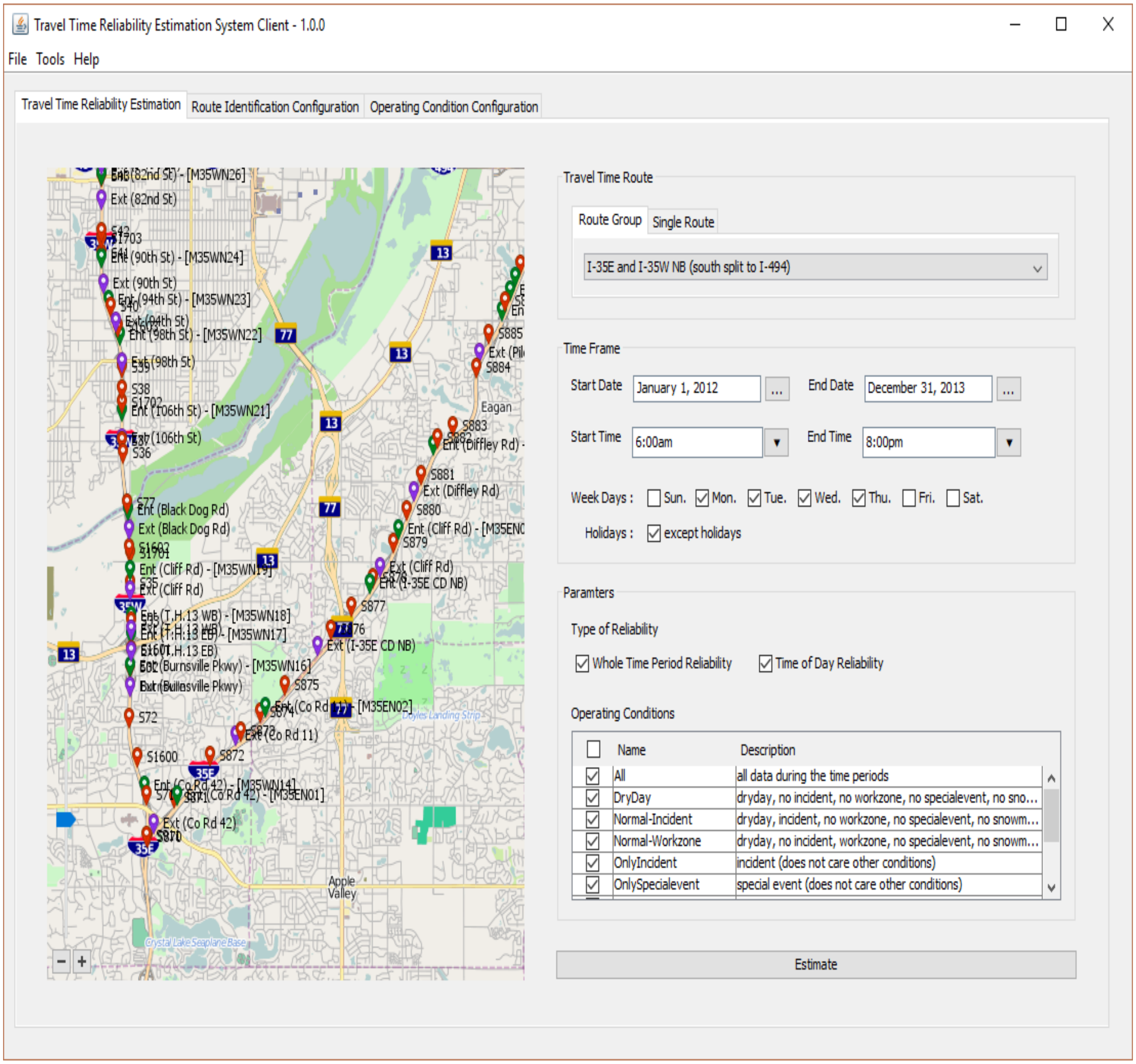


Figure 7.2.2 Estimation Panel of the User Interface

Development of the Route-Selection Panel

Figure 7.2.3 shows the screenshot of the Route-Selection Panel, where user can select a route or a route group from pre-defined routes by the Administrator Client. The predefined reliability routes for a selected corridor can be shown in the Panel as illustrated in Figure 7.2.3. Further, the list of the selected routes in a group is stored in a local disk and can be retrieved for future use.

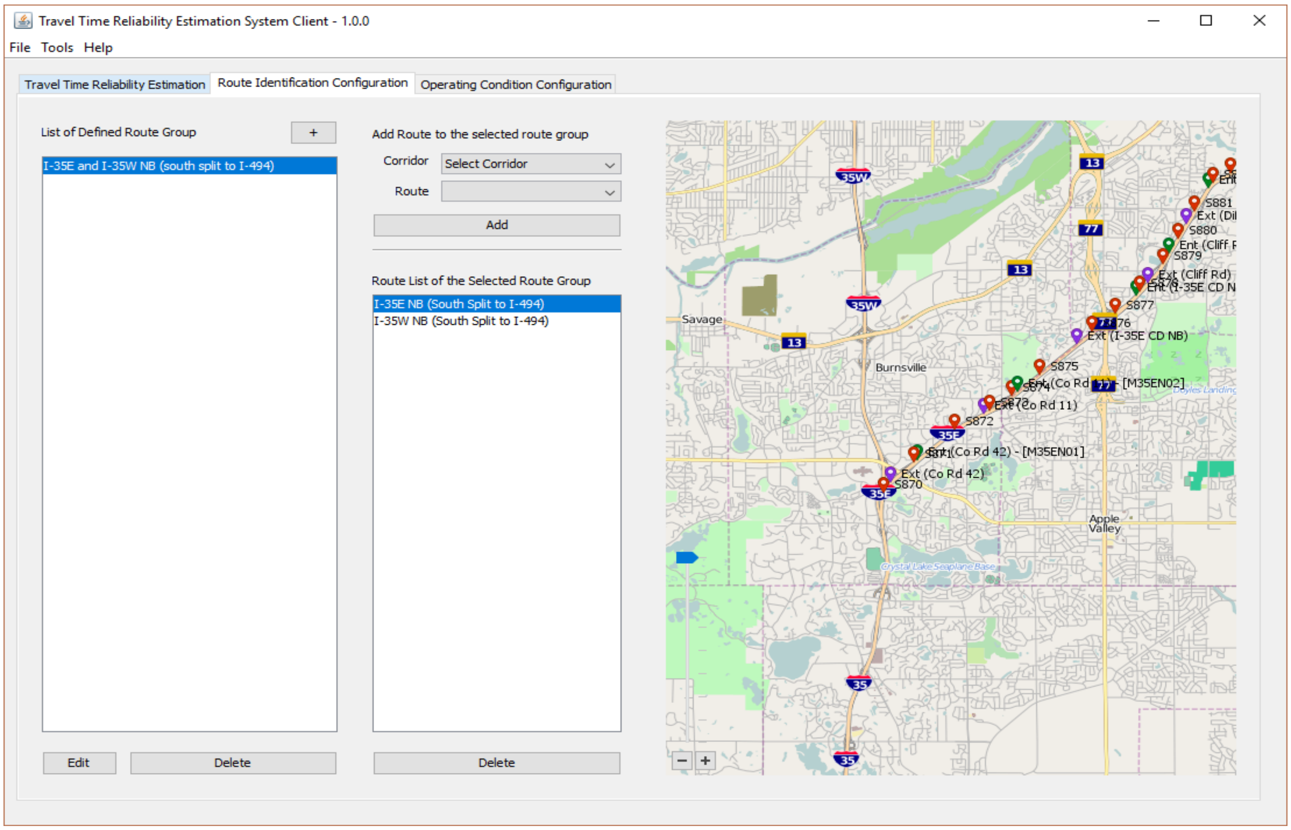


Figure 7.2.3 Route-Selection Panel of the User Client

Development of the Operating-Condition Configuration Panel

Figure 7.2.4 shows the screenshot of the Operating-Condition Configuration Panel, where user can specify a set of operating conditions under which reliability measures would be calculated. The operating conditions to be specified include types of weather, existence and types of incident, work-zone and special event. Also, the road conditions during snow events can be specified with this panel. As shown in Figure 5, the operating conditions can be specified in two ways:

(1) Specific conditions can be added by using “Add Condition” button in each tab in Figure 7.2.4.

(2) Check boxes of “Without any < *condition name*>” and “With < *condition name*>” are used for binary selection.

In the current version of the User Interface, the operating conditions specified by user are applied as follows:

- If there are multiple sub-conditions checked in a same operating condition, “OR” operator is applied.

- e.g. if “Light Rain”, “Moderate Rain” and “Heavy Rain” are checked in the weather tab, all rainy day data are used in the estimation process .

- If multiple operating conditions are set, “AND” operator is applied:

- e.g. if “Normal Dry Day” is selected in the weather tab and “Property Damage of Crash” are checked in the incident tab, only the travel-time data under normal dry-weather condition and incident(s) with property damage will be used to calculate reliability measures.

- If a certain operating condition is not specified, that operation condition is not considered when filtering travel-time data for reliability estimation.

- e.g. if “Normal Dry Day” is selected in the weather tab and any other operating condition, e.g., incident or work-zone, is not set, the travel- time data under normal dry day are used regardless of incident or work zone conditions.

Figure 7.2.5 shows various types and levels of sub-conditions for each operating condition and their possible combinations that can be specified in the current version. Figure 7.2.6 also shows one example dialog to add a sub-condition.

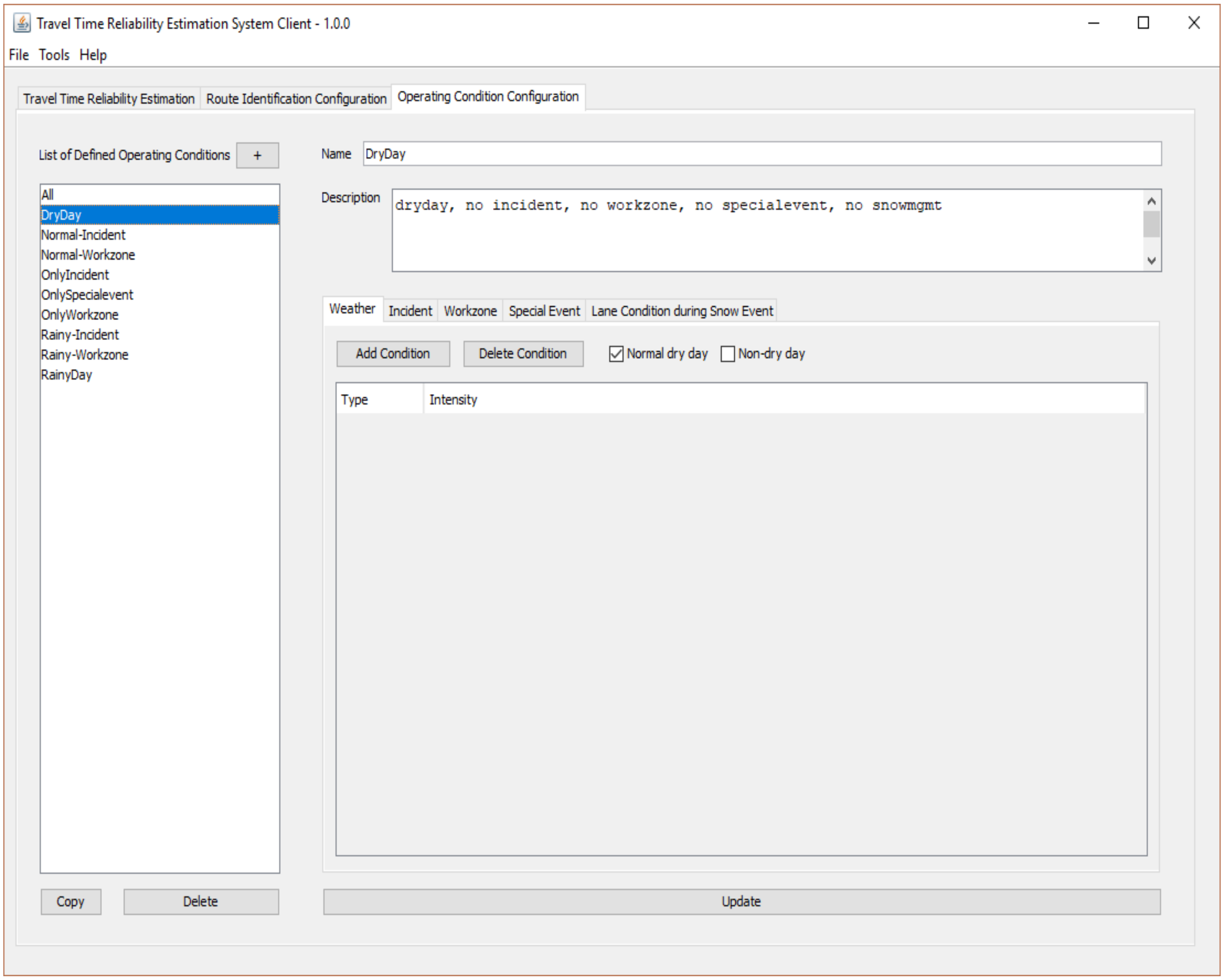


Figure 7.2.4 Operating Condition-Configuration Panel of the User Interface

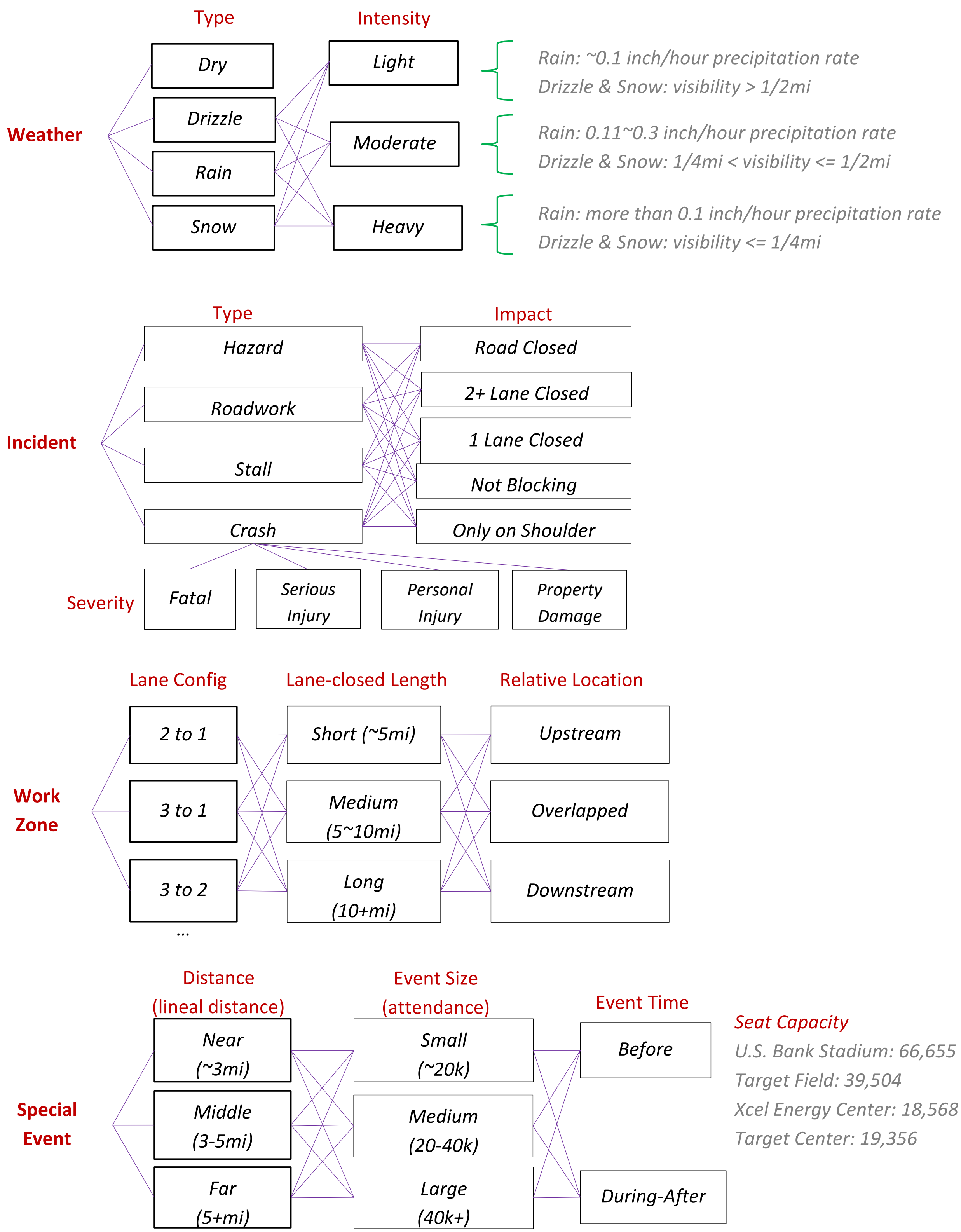


Figure 7.2.5 Combinations of the Operating Sub-Conditions

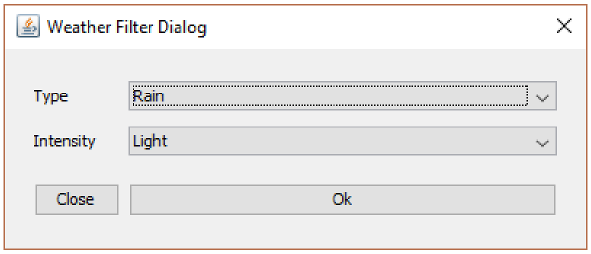


Figure 7.2.6 Dialog to Add a Weather sub-Condition

Development of API Client module

Figure 7.2.7 shows the structure of the *API Client* module, which facilitates the data exchange between the User-Interface and other modules. The main functionalities of each submodule of the API client module are as follows;

- *Reliability-Estimation Client* module delivers the user parameters from the *Reliability Estimation Panel* to the server through the *HTTP Client* module.

- *Travel-Time Route Client* module retrieves the travel-time route list from the server. The route list is then used in the *Reliability Estimation Panel* and the *Route Identification Configuration Panel*

- *HTTP-Client* module implements the POST and GET method of HTTP in both synchronous and asynchronous ways. The methods of the HTTP Client are described in Figure 7.2.8.

- *Data Types* module contains the related data types such as *ReliabilityRoute*, *OperatingConditionsInfo* and *EstimationRequestInfo* classes as shown in Figure 7.2.9.

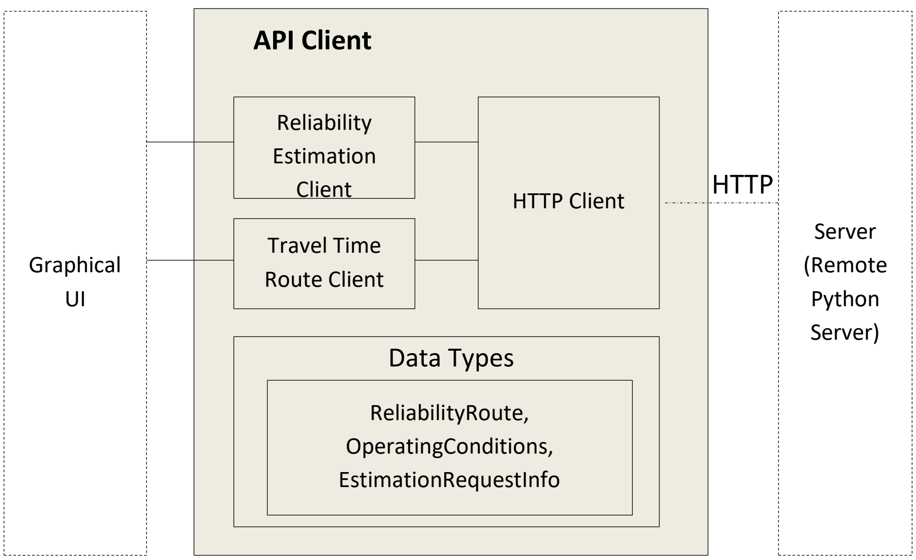


Figure 7.2.7 API Client module structure

```
 public class HttpClient {
  public static HttpURLConnection getConnection(String target_url);
  public static void get(String target_url, IHttpResultCallback callback);
  public static HttpResult get_synced(String target_url);
public static <T extends Response> void get(String target_url, IResponseCallback<T> callback, Class<T> type);
public static <T extends Response> T get_synced(String target_url,  <T> type);
  public static void post(String uri_path, PostData pd, IHttpResultCallback callback);
  public static HttpResult post_synced(String target_url, PostData postData);
public static <T extends Response> void post(String uri_path, PostData pd,
                                            IResponseCallback<T> callback,  <T> type);
public static <T extends Response> T post_synced(String target_url, pd, Class<T> type);
public static <T extends Response> T post_synced(String target_url, PostData pd, Type type);
}
```

Figure 7.2.8 HttpClient class structure

```
public class EstimationRequestInfo extends InfoBase {
   public ReliabilityRouteInfo travel_time_route;
   public String start_date;
   public String end_date;
   public String start_time;
   public String end_time;
   public WeekdayConditionInfo weekdays;
   public Boolean except_holiday;
   public ReliabilityEstimationModeInfo estmation_mode;
   public List<OperatingConditionsInfo> operating_conditions;
}
public class OperatingConditionsInfo extends InfoBase {
   public String name;
   public String desc;
   public List<WeatherConditionInfo> weather_conditions;
   public List<IncidentConditionInfo> incident_conditions;
   public List<WorkzoneConditionInfo> workzone_conditions;
   public List<SpecialeventConditionInfo> specialevent_conditions;
   public List<SnowmanagementConditionInfo> snowmanagement_conditions;
}
public class ReliabilityRouteInfo extends InfoBase {
   public String name;
   public String description;
   public String corridor;
   public Route route;
}
```

Figure 7.2.9 Data Types used in API Client module

7.2.3 Development of the User-Service module

Figure 7.2.10 shows the structure of the *User Service* module, which facilitates the data exchanges between the User-Interface module and the Reliability-Estimation-Report Generation module. The main functionalities of the two submodules are as follows:

- *Reliability Estimation Handler* module receives the user-specified parameters from the user interface and executes the travel-time reliability estimation process implemented in the *Reliability-Estimation and Report-Generation module.*

- *Travel-Time Route Data Handler* module receives the request of travel-time route list, retrieves the pre-defined travel time routes through the *DB Access Layer* module and returns the list to the client.

Figure 7.2.11 includes the source code of the *User Service* module.

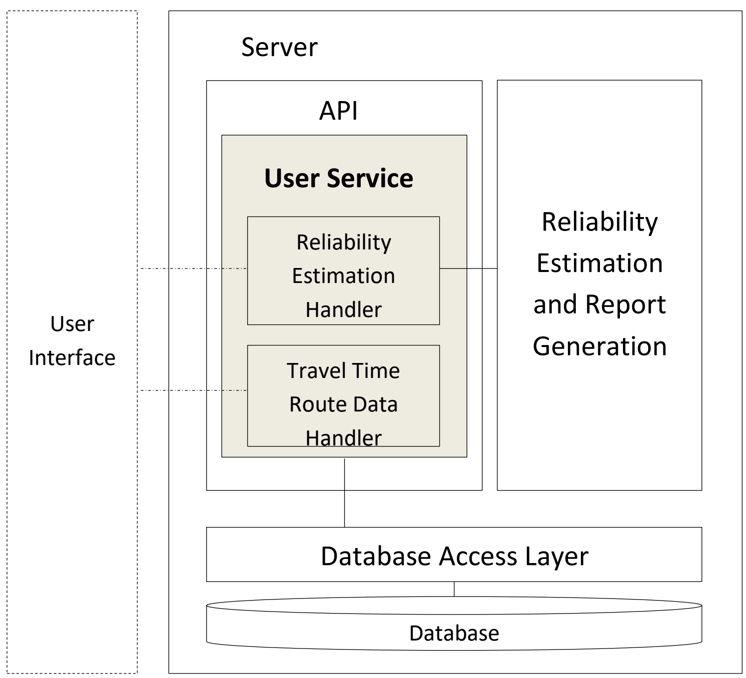


Figure 7.2.10 User-Service module structure

```
# Reliability Estimation Handler
@app.route(api_urls_user.ESTIMATION, methods=['POST'])
def tetres_user_estimation():
 # parse user parameter
  routes = request.form.get('routeIDs')
  route_ids = json.loads(routes)
  param = request.form.get('param')
  eparam = json.loads(param)
  setattr(eparam, 'travel_time_route', None)
# run estimation process for all the given routes
  ttr_da = TTRouteDataAccess()
  for a_route_id in route_ids:
    eparam.travel_time_route = ttr_da.get_by_id(a_route_id)
    if not eparam.travel_time_route:
      return prot.response_error('The travel time route does not exist')
    estimation.estimation(eparam)
# return the success message to the client
  return prot.response_success()

# Travel Time Route Data Handler
@app.route(api_urls_user.ROUTE_LIST, methods=['POST'])
def tetres_user_route_list():
 # parse user parameter
  corridor_name = request.form.get('corridor')
# retrieve route list for the given corridor through DB access module
  da = TTRouteDataAccess()
  ttris = list(da.list_by_corridor(corridor_name, order_by=('name', 'desc'), window_size=10000))
  da.close()

# return the list as JSON
  return prot.response_success({'list': ttris})
```

Figure 7.2.11 Source code of User-Service module

7.3 DEVELOPMENT OF THE RELIABILITY-ESTIMATION AND REPORT-GENERATION MODULE

7.3.1 Overview of the Reliability Estimation and Report module

Figure 7.3.1 shows the structure of the Reliability Estimation and Report Generation Module, which performs the reliability estimation process for user-specified conditions delivered through the User-Service module. After the calculation is completed, it calls the Report-Generation module, which creates a set of the output files in the spreadsheet and graph formats.

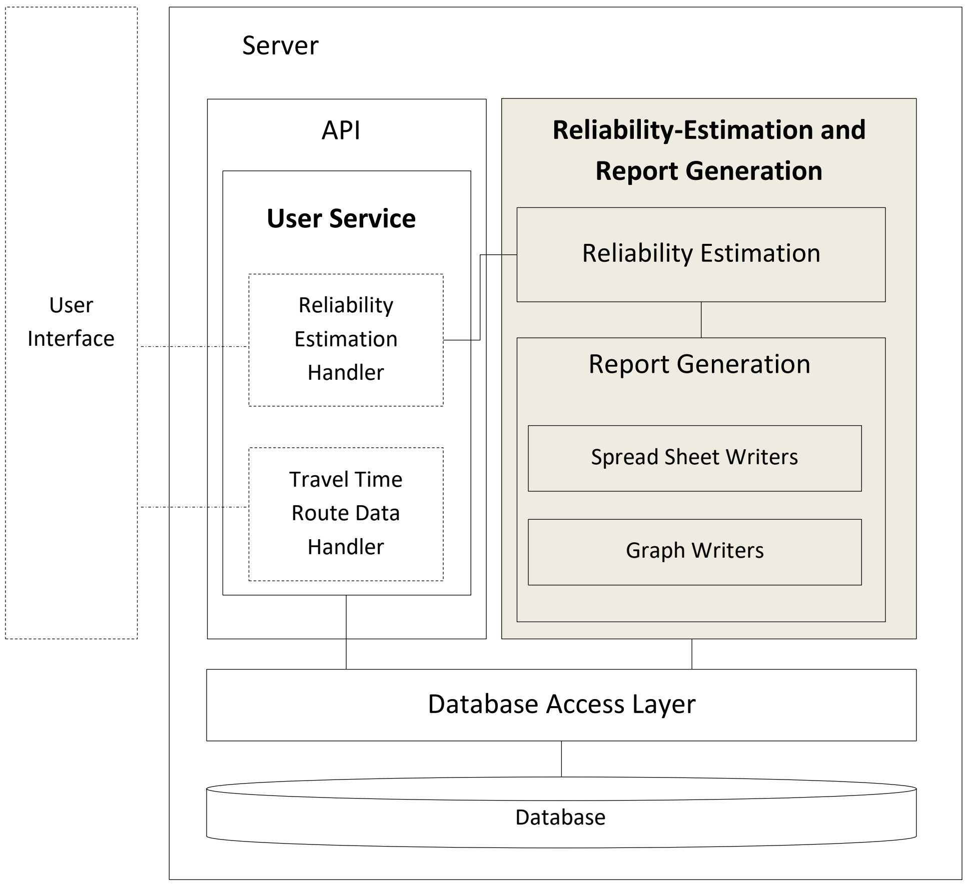


Figure 7.3.1 Estimation and Report module structure

7.3.2 Development of the Reliability-Estimation Process module

Figure 7.3.2 shows the structure of the *Reliability Estimation* process module, where the *Operating-Condition Filter Creator* first creates the filter functions for each operating-condition group specified by user in the User-Interface. Then the Reliability-Estimation module calls the *Travel Time and Reliability* module, developed in the previous chapter, and calculates the reliability measures with the filtered data for the given set of operating conditions.

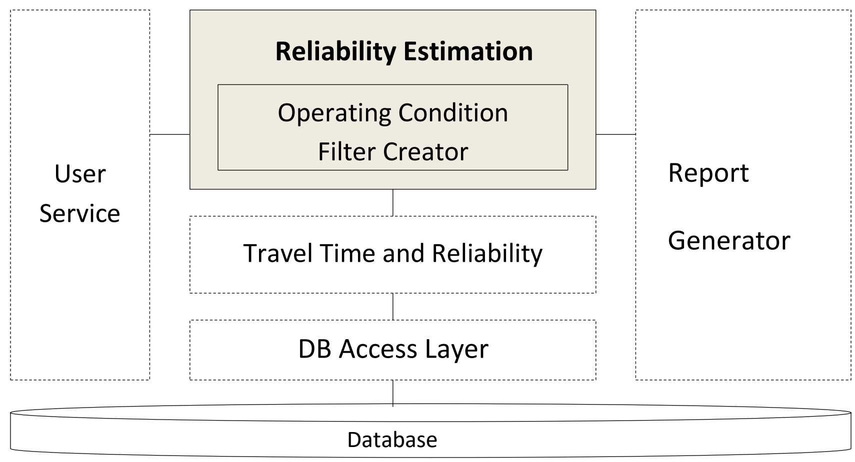


Figure 7.3.2 Reliability-Estimation module structure

Figure 7.3.3 shows the sequence diagram of the reliability estimation process performed in the Reliability-Estimation process module. The step-by-step process is as follows:

(1) Make the operating condition filter functions using the *Operating Condition Filter Creator* module.

- The filters are defined as a class containing callable function and travel-time data list for storing the travel-time data passed the filter created for the operating conditions specified by user.

(2) Retrieve travel-time data and operating condition data, such as weather and incident, during the given time period from the database.

(3) Iterate the following steps for all the given operating conditions:

(i) Iterate for all travel time data and check if it is passed by the filter function.

(ii) Store the travel time data and operating conditions data to the data list of the filter if it is passed

(4) Make yearly, monthly and daily data set with the filtered data

(5) Calculates the reliability measures with all data set including yearly, monthly and daily data sets for all the given operating conditions, depending on the selected type of the reliability measure, i.e., whole time period reliability or time of day reliability

(6) Write output files using the *Report Generation* module.

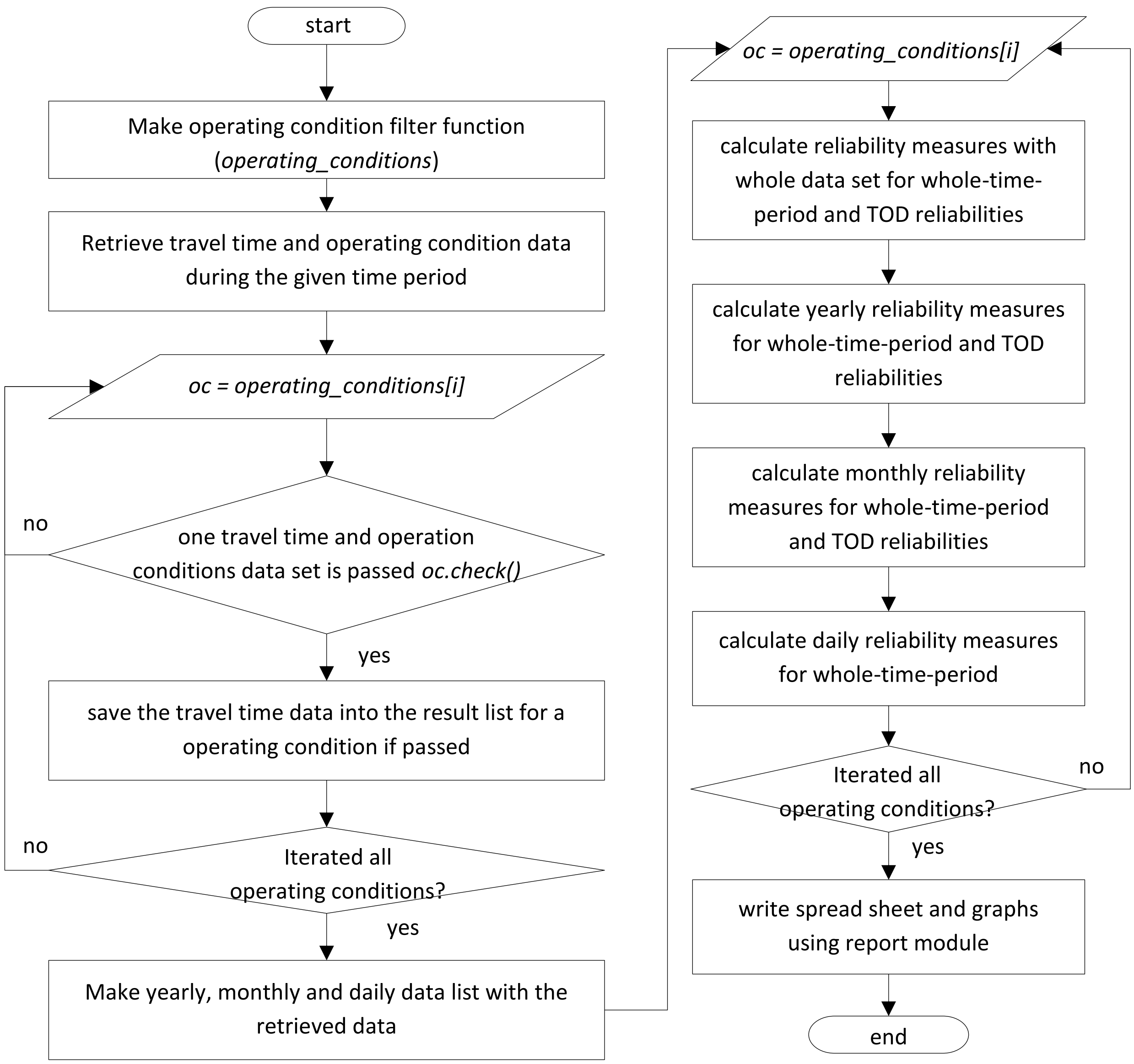


Figure 7.3.3 Sequence Diagram of Reliability-Estimation Process

7.3.3 Development of the Report-Generation module

The *Report-Generation* module saves the reliability measures calculated by the *Reliability-Estimation* module in the form of spread sheet and graph images. Specifically, the following spread-sheet and graph writers have been developed and included in this module.

Whole Time Reliability Writer creates a spread-sheet file with the reliability measures of whole-time-period data. Figure 7.3.4 shows an example spreadsheet file:

- The first sheet shows the given operating conditions as defined in the *User Interface.*

- The other sheets contain the reliability measures with all data, yearly data, monthly data and daily data sets estimated for given operating conditions.

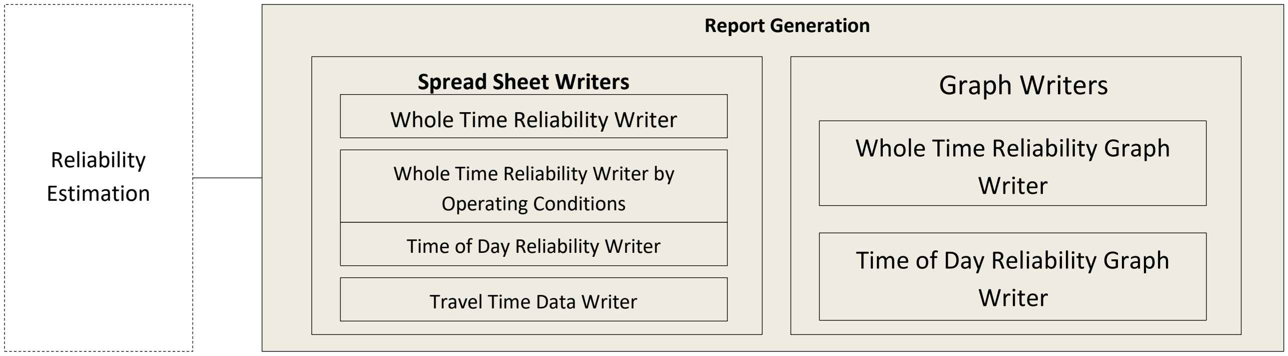


Figure 7.3.4 Sequence Diagram of Reliability Estimation Process

	A	B	C
1	Index	Name	Description
2	0	All	all data during the time periods
3	1	DryDay	dryday, no incident, no workzone, no specialevent, no snowmgmt
4	2	Normal-Incident	dryday, incident, no workzone, no specialevent, no snowmgmt.
5	3	Normal-Workzone	dryday, no incident, workzone, no specialevent, no snowmgmt.
6	4	OnlyIncident	incident (does not care other conditions)
7	5	Rainy-Incident	rain, incident, no workzone, no specialevent, no snowmgmt.
8	6	RainyDay	rain, no incident, no workzone, no specialevent, no snowmgmt.

Operating Coditions (OC) | reliabilities | yearly (OC=0) | yearly (OC=1) ...

	A	B	C	D	E	F	G	H	I	
1	OC Index	OC Name	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste(	Congeste(	80th %-ile	85
2	0	All	8.680025	0.859317	48494	8.657143	16.26591	2344	8.612107	8
3	1	DryDay	8.444492	0.835999	38991	8.657143	16.14746	1054	8.486433	8
4	2	Normal-In	9.105886	0.901477	3012	8.657143	17.63299	226	8.816319	9
5	3	Normal-W	11.2232	1.11109	1227	8.657143	14.76384	414	13.49159	1
6	4	OnlyIncid	9.393259	0.929926	3608	8.657143	17.95111	351	9.04143	9
7	5	Rainy-Inci	9.986566	0.988663	47	8.657143	14.36758	9	10.94764	
8	6	RainyDay	9.237043	0.914461	339	8.657143	17.7238	26	9.200508	9

Operating Coditions (OC) | reliabilities | yearly (OC=0) | yearly (OC=1) ...

	A	B	C	D	E	F	G	H
1	Operating Condition:	All						
2	Year	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste(	Congeste(	80th %-ile
3	2012	8.325107	0.82418	12078	8.657143	14.47084	308	8.403668
4	2013	9.08496	0.899405	12077	8.657143	15.19982	1203	9.215048
5	2014	8.763819	0.867612	12139	8.657143	18.74535	498	8.62815
6	2015	8.547166	0.846164	12200	8.657143	18.05883	335	8.547203

... yearly (OC=0) | yearly (OC=1) | yearly (OC=2) | yearly (OC=3) | yearly ...

	A	B	C	D	E	F	G	H
1	Operating Condition:	All						
2	Month	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste(	Congeste(	80th %-ile
3	2012-01	8.609097	0.852295	976	8.657143	16.75262	57	8.348761
4	2012-02	8.183108	0.810122	976	8.657143	12.80277	19	8.340704
5	2012-03	8.07191	0.799114	1037	8.657143	14.43168	9	8.24018
6	2012-04	8.191574	0.81096	1037	8.657143	13.81949	8	8.391909
7	2012-05	8.262629	0.817995	1098	8.657143	14.67636	17	8.419395
8	2012-06	8.158563	0.807692	976	8.657143	11.69879	4	8.372985
9	2012-07	8.117252	0.803603	1037	8.657143	-1	0	8.328329
10	2012-08	8.338566	0.825513	1098	8.657143	13.89817	31	8.400275
11	2012-09	8.608523	0.852238	915	8.657143	12.55323	58	8.637377

... monthly (OC=0) | monthly (OC=1) | monthly (OC=2) | monthly (OC=3) ...

	A	B	C	D	E	F	G	H	
1	Operating Condition:	All							
2	Date	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste(	Congeste(	80th %-ile	85th
3	2012-01-03	7.94016	0.786071	61	8.657143	-1	0	8.055566	8.0
4	2012-01-04	7.91225	0.783307	61	8.657143	-1	0	8.082325	8.0
5	2012-01-05	7.914535	0.783534	61	8.657143	-1	0	8.031764	8.0
6	2012-01-09	7.915031	0.783583	61	8.657143	-1	0	8.102116	8.1
7	2012-01-10	7.89106	0.78121	61	8.657143	-1	0	8.049554	8.0
8	2012-01-11	7.95658	0.787696	61	8.657143	-1	0	8.174802	8.2
9	2012-01-12	8.112981	0.80318	61	8.657143	-1	0	8.324667	8.3
10	2012-01-17	10.31598	1.021275	61	8.657143	15.00181	17	13.10364	14.
11	2012-01-18	8.016247	0.793603	61	8.657143	-1	0	8.206631	8.2

... monthly (OC=5) | monthly (OC=6) | daily (OC=0) | daily (OC=1) | da ...

Figure 7.3.5 Output File Example of Whole-Time-Reliability Writer module

Whole Time Reliability (WTP) Writer by Operating Conditions writes a spread-sheet file with the WTP reliability measures for pre-specified operating conditions. Figure 7.3.6 shows an output example from this module:

- The first sheet shows the given operating conditions specified in the User-Interface same as in the previous module.

- The other sheets contain the reliability estimates from each data set, i.e., all data, yearly data, monthly data and daily data set, for each operating condition.

	A	B	C	D	E	F	G
1	OC Index	OC Name	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste C
2	0	All	8.680025	0.859317	48494	8.657143	16.26591
3	1	DryDay	8.444492	0.835999	38991	8.657143	16.14746
4	2	Normal-Incident	9.105886	0.901477	3012	8.657143	17.63299
5	3	Normal-Workzone	11.2232	1.11109	1227	8.657143	14.76384
6	4	OnlyIncident	9.393259	0.929926	3608	8.657143	17.95111
7	5	Rainy-Incident	9.986566	0.988663	47	8.657143	14.36758
8	6	RainyDay	9.237043	0.914461	339	8.657143	17.7238

Operating Coditions (OC) | **reliabilities** | yearly AvgTT | yearly TT

	A	B	C	D	E	F	G	H
1	Year	All	DryDay	Normal-In	Normal-W	OnlyIncid	Rainy-Inci	RainyDay
2	2012	0.135341	0.078949	0.463777	-1	0.533973	-1	0.044097
3	2013	0.522338	0.325698	0.561849	0.595935	0.61397	0.10144	0.114606
4	2014	0.20161	0.102919	0.327847	-1	0.50182	0.586869	0.170514
5	2015	0.1397	0.083074	0.410631	-1	0.361485	-1	0.266406

... yearly BufferIndex(90%) | **yearly BufferIndex(95%)** | yearly Planning

	A	B	C	D	E	F	G	H
1	Month	All	DryDay	Normal-In	Normal-W	OnlyIncid	Rainy-Inci	RainyDay
2	2012-01	0.449268	0.063659	0.239152	-1	0.179842	-1	-1
3	2012-02	0.122798	0.070863	0.138592	-1	0.373489	-1	-1
4	2012-03	0.058885	0.059196	0.056554	-1	0.66339	-1	-1
5	2012-04	0.067675	0.064568	0.568564	-1	0.568564	-1	-1
6	2012-05	0.075286	0.069207	0.140977	-1	0.128482	-1	0.058796
7	2012-06	0.075009	0.063653	0.044499	-1	0.046374	-1	-1
8	2012-07	0.058787	0.056349	0.039285	-1	0.039285	-1	-1
9	2012-08	0.135421	0.064431	0.207198	-1	0.311578	-1	0.027291
10	2012-09	0.363985	0.365936	0.451728	-1	0.389409	-1	-1
11	2012-10	0.188961	0.195181	1.134463	-1	1.100776	-1	-1

... monthly BufferIndex(90%) | **monthly BufferIndex(95%)** | monthly

	A	B	C	D	E	F	G	H
1	Date	All	DryDay	Normal-In	Normal-W	OnlyIncid	Rainy-Inci	RainyDay
2	2012-01-03	0.03584	0.03584	-1	-1	-1	-1	-1
3	2012-01-04	0.04634	0.04634	-1	-1	-1	-1	-1
4	2012-01-05	0.032909	0.032909	-1	-1	-1	-1	-1
5	2012-01-09	0.030625	0.030517	-1	-1	-1	-1	-1
6	2012-01-10	0.044486	0.044486	-1	-1	-1	-1	-1
7	2012-01-11	0.056348	0.058033	-1	-1	-1	-1	-1
8	2012-01-12	0.054634	0.064078	-1	-1	-1	-1	-1
9	2012-01-17	0.67085	0.111667	0.07895	-1	0.068359	-1	-1
10	2012-01-18	0.057066	0.05642	-1	-1	-1	-1	-1
11	2012-01-19	0.050893	0.050893	-1	-1	-1	-1	-1

... daily BufferIndex(90%) | **daily BufferIndex(95%)** | daily PlanningTim

Figure 7.3.6 Output File Example of Whole Time Period Reliability Writer by Operating Conditions

Time of Day (TOD) Reliability Writer writes TOD reliability measures with all data set, yearly data set and monthly data set, for each operating condition as shown in Figure 7.3.7.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Operating	All														
2	Time	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste	Congeste	80th %-ile	85th %-ile	90th %-ile	95th %-ile	Buffer Ind	Buffer Ind	Buffer Ind	Buffer Ind	Planning T
3	06:00	8.040384	0.795993	795	8.657143	12.1454	9	8.172637	8.284777	8.478063	8.961146	0.016449	0.030396	0.054435	0.114517	0.944034
4	06:05	8.03064	0.795028	795	8.657143	12.08002	10	8.150427	8.259143	8.413118	8.918956	0.014916	0.028454	0.047627	0.110616	0.941468
5	06:10	8.057226	0.79766	795	8.657143	12.33521	10	8.177978	8.28657	8.445861	9.14233	0.014987	0.028464	0.048234	0.134675	0.944651
6	06:15	8.096678	0.801566	795	8.657143	12.39459	14	8.173175	8.294543	8.537933	9.168277	0.009448	0.024438	0.054498	0.13235	0.944096
7	06:20	8.14166	0.806019	795	8.657143	12.73564	15	8.192073	8.33011	8.603415	9.369656	0.006192	0.023146	0.056715	0.150829	0.946279
8	06:25	8.200268	0.811821	795	8.657143	12.92633	17	8.265009	8.378842	8.661321	9.500081	0.007895	0.021777	0.056224	0.158509	0.954704
9	06:30	8.241194	0.815873	795	8.657143	13.42034	17	8.311078	8.41888	8.78201	9.590109	0.00848	0.021561	0.065624	0.16368	0.960026
10	06:35	8.276562	0.819374	795	8.657143	13.84924	18	8.288002	8.418634	8.861882	9.948592	0.001382	0.017166	0.07072	0.20202	0.95736
11	06:40	8.307545	0.822441	795	8.657143	13.57953	25	8.307132	8.435013	8.919399	10.36325	-5E-05	0.015344	0.07365	0.24745	0.95957

Operating Coditions (OC) | **reliabilites (OC=0)** | reliabilites (OC=1) | re ...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Operating	All													
2	Year	Time	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste	Congeste	80th %-ile	85th %-ile	90th %-ile	95th %-ile	Buffer Ind	Buffer Ind	Buffer Ind
3	2012	06:00	7.895986	0.781697	198	8.657143	11.75676	2	7.993566	8.054325	8.141896	8.428826	0.012358	0.020053	0.031144
4	2012	06:05	7.879351	0.78005	198	8.657143	11.85472	2	7.970346	8.012473	8.135868	8.344852	0.011549	0.016895	0.032556
5	2012	06:10	7.915806	0.78366	198	8.657143	12.52266	2	8.040111	8.088145	8.135175	8.338135	0.015703	0.021771	0.027713
6	2012	06:15	7.922109	0.784284	198	8.657143	12.19403	2	7.978602	8.024165	8.140941	8.483616	0.007131	0.012882	0.027623
7	2012	06:20	7.958014	0.787838	198	8.657143	13.03028	2	8.020534	8.055473	8.122399	8.373583	0.007856	0.012247	0.020657
8	2012	06:25	7.988188	0.790825	198	8.657143	12.42181	3	8.06605	8.131385	8.229493	8.376901	0.009747	0.017926	0.030208
9	2012	06:30	8.003223	0.792314	198	8.657143	12.90381	3	8.037365	8.092477	8.147831	8.430112	0.004266	0.011152	0.018069
10	2012	06:35	8.001226	0.792116	198	8.657143	13.02715	3	8.066626	8.116173	8.170146	8.488028	0.008174	0.014366	0.021112
11	2012	06:40	8.028814	0.794847	198	8.657143	12.90593	4	8.042651	8.110929	8.272081	8.420832	0.001723	0.010228	0.030299

	P
1	
2	Buffer Ind
3	0.067482
4	0.059079
5	0.053353
6	0.070878
7	0.05222
8	0.048661
9	0.05334
10	0.060841
11	0.048826

... reliabilites (OC=5) | reliabilites (OC=6) | **yearly (OC=0)** | yearly (OC=1) ...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Operating	All														
2	Month	Time	Avg TT	Travel Tim	Data Coun	Free-Flow	Congeste	Congeste	80th %-ile	85th %-ile	90th %-ile	95th %-ile	Buffer Ind	Buffer Ind	Buffer Ind	Buffer Ind
3	2012-01	06:00	8.068372	0.798763	16	8.657143	11.30002	1	8.137909	8.23685	8.287287	9.053562	0.008618	0.020881	0.027132	0.122105
4	2012-01	06:05	8.049434	0.796889	16	8.657143	11.33584	1	7.983531	8.191818	8.305004	9.095531	-0.00819	0.017689	0.03175	0.129959
5	2012-01	06:10	8.083586	0.80027	16	8.657143	11.93522	1	7.951982	8.081384	8.169829	9.145159	-0.01628	-0.00027	0.010669	0.131325
6	2012-01	06:15	8.186031	0.810412	16	8.657143	11.90569	1	8.184247	8.188224	8.359028	9.372802	-0.00022	0.000268	0.021133	0.144975
7	2012-01	06:20	8.241967	0.815949	16	8.657143	12.4674	1	8.104864	8.200464	8.603806	9.848309	-0.01663	-0.00504	0.043902	0.194898
8	2012-01	06:25	8.199537	0.811749	16	8.657143	12.1247	1	8.260919	8.301323	8.585483	9.673306	0.007486	0.012414	0.047069	0.179738
9	2012-01	06:30	8.305444	0.822233	16	8.657143	13.28149	1	8.139707	8.158652	8.393814	9.787369	-0.01996	-0.01767	0.01064	0.178428
10	2012-01	06:35	8.330041	0.824668	16	8.657143	13.75378	1	8.192297	8.250949	8.481761	9.958212	-0.01654	-0.00949	0.018214	0.195458
11	2012-01	06:40	8.393743	0.830975	16	8.657143	14.39633	1	8.209454	8.280664	8.540673	10.18179	-0.02196	-0.01347	0.017505	0.213022

... yearly (OC=4) | yearly (OC=5) | yearly (OC=6) | **monthly (OC=0)** | mc ...

Figure 7.3.7 Output File Example of Time of Day Reliability Writer module

Travel-Time Data Writer writes travel-time data and the related non-traffic data, such weather and incident, depending on the specified operating conditions as shown in Figure 7.3.8. Using this data set, user can calculate different types of reliability measures not defined in the current version of TTRMS.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Operating Condition:	Normal-Incident													
2					weather					incident					
3	time	tt	speed	vmt	usaf	wban	precip_ty	precip	precip_int	type	impact	cdts	udts	xdts	distance
4	2012-01-09 07:15	7.712595	78.82043	3444.1	726580	14922			0	Missing	PROPERTY DAMAGE	2012-01-0	2012-01-0	2012-01-0	12.76
5	2012-01-12 08:30	7.943758	76.68258	2467.75	726580	14922			0	Missing	DEBRIS ON ROADWA	2012-01-1	2012-01-1	2012-01-1	10.15
6	2012-01-17 07:55	17.65751	44.34509	3249.65	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
7	2012-01-17 08:00	17.23645	44.34085	3212.25	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
8	2012-01-17 08:05	17.33997	46.58603	3157.95	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
9	2012-01-17 08:10	17.48691	48.28028	2879.5	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
10	2012-01-17 08:15	14.49533	49.71594	2753.7	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
11	2012-01-17 08:20	13.73941	51.82852	2614.7	726580	14922			0	Missing	MEDICAL	2012-01-1	2012-01-1	2012-01-1	12.76
12	2012-02-02 07:30	9.727437	65.88184	3595.8	726580	14922			0	Not Repor	PROPERTY DAMAGE	2012-02-0	2012-02-0	2012-02-0	14.17
13	2012-02-02 07:35	9.594685	64.42435	3580.2	726580	14922			0	Not Repor	PROPERTY DAMAGE	2012-02-0	2012-02-0	2012-02-0	14.17
14	2012-02-02 07:40	9.724336	64.03348	3571.4	726580	14922			0	Not Repor	PROPERTY DAMAGE	2012-02-0	2012-02-0	2012-02-0	14.17
15	2012-02-02 07:45	9.280591	64.33515	3627.65	726580	14922			0	Not Repor	PROPERTY DAMAGE	2012-02-0	2012-02-0	2012-02-0	14.17

Operating Coditions (OC) | data (OC=0) | data (OC=1) | **data (OC=2)** | ...

Figure 7.3.8 Output File Example of Travel Time Data Writer module

Whole-Time-Reliability (WTP) Graph Writer develops a set of multiple graph-image files with WTP reliability measures. Figures 7.3.9 - 16 show the types of the output graphs currently available. It needs to be noted that the number of graph files depends on the number of operating conditions specified by user.

The types of graphs currently available are as follows:

- Cumulative probability of travel time rate for each operating condition (Figure 7.3.9)

- Reliability index for each operating condition (Figure 7.3.10)

- Yearly reliability index variations (Figure 7.3.11)

- Yearly multiple reliability indices comparison (Figure 7.3.12)

- Monthly reliability index variations (Figure 7.3.13)

- Monthly multiple reliability indices comparison (Figure 7.3.14)

- Daily reliability index variations (Figure 7.3.15)

- Relationship of travel time rate and buffer index using daily data (Figure 7.3.16)

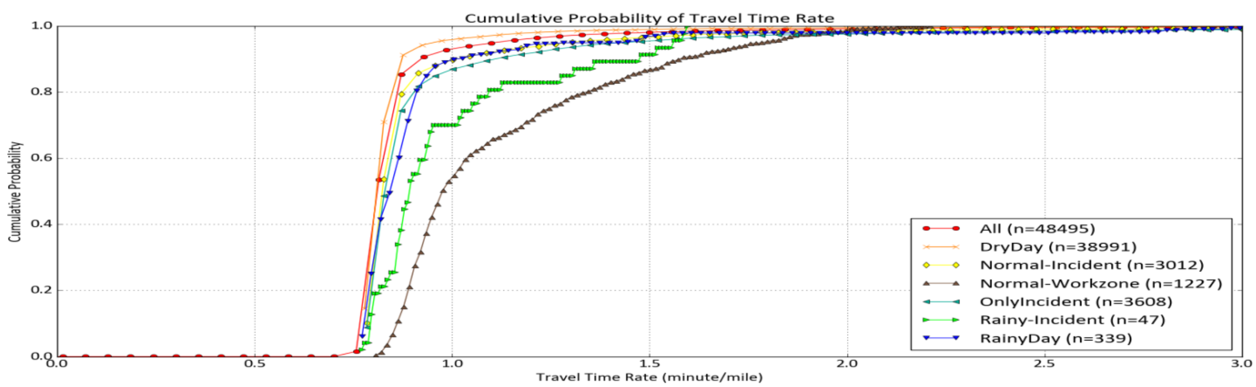


Figure 7.3.9 An Example Cumulative-Probability graph of Travel Time Rate

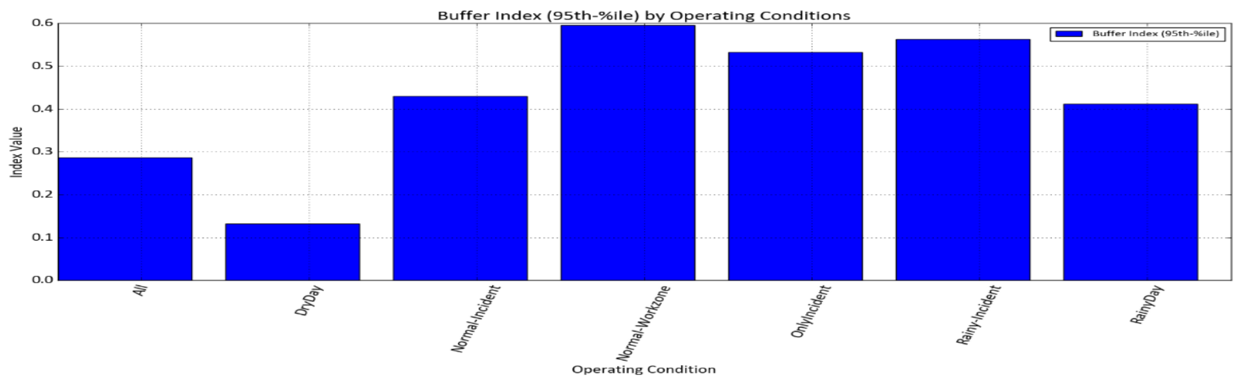


Figure 7.3.10 An Example Buffer-Index graph depending on Operating Conditions

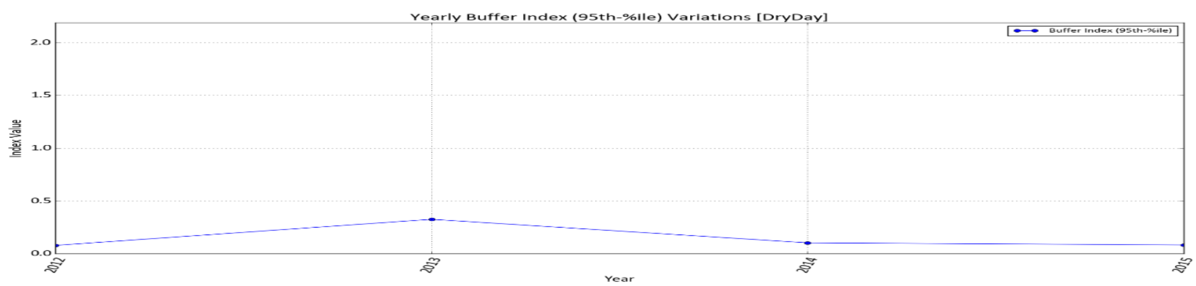


Figure 7.3.11 An Example Graph for Yearly Buffer Index Variations

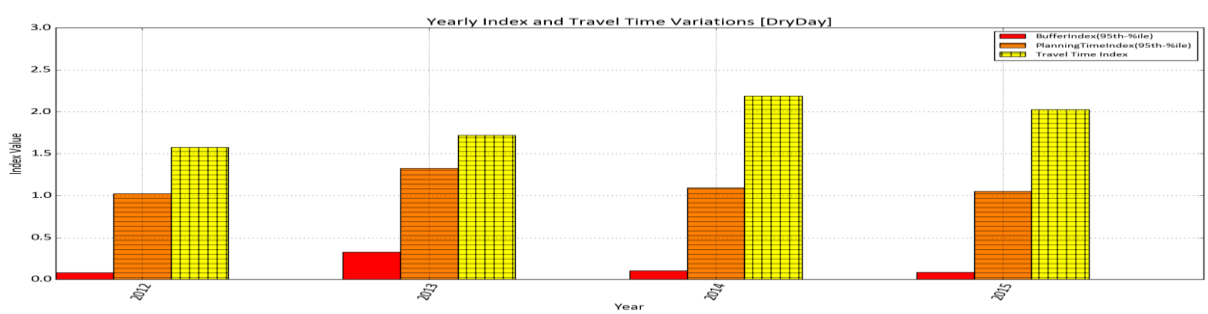


Figure 7.3.12 An Example Graph for Yearly Multiple Indices Comparison

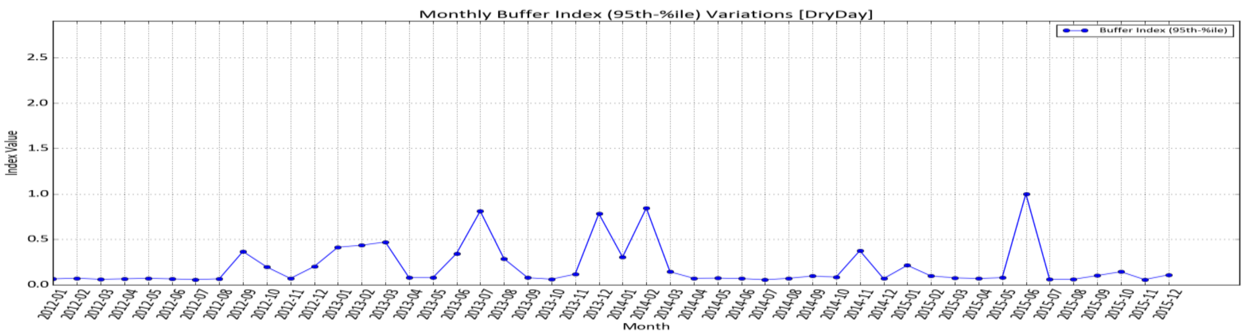


Figure 7.3.13 An Example Graph for Monthly Buffer Index Variations

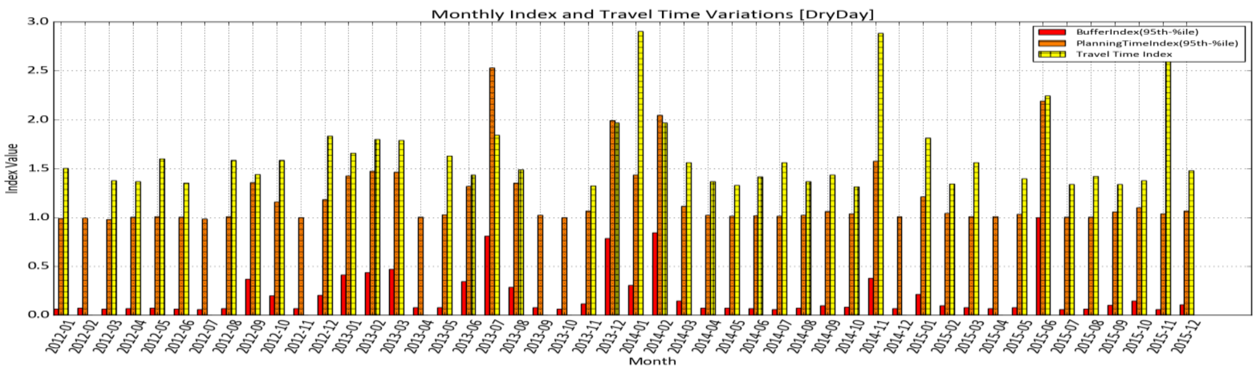


Figure 7.3.14 An Example Graph for Monthly Multiple Indices Comparison

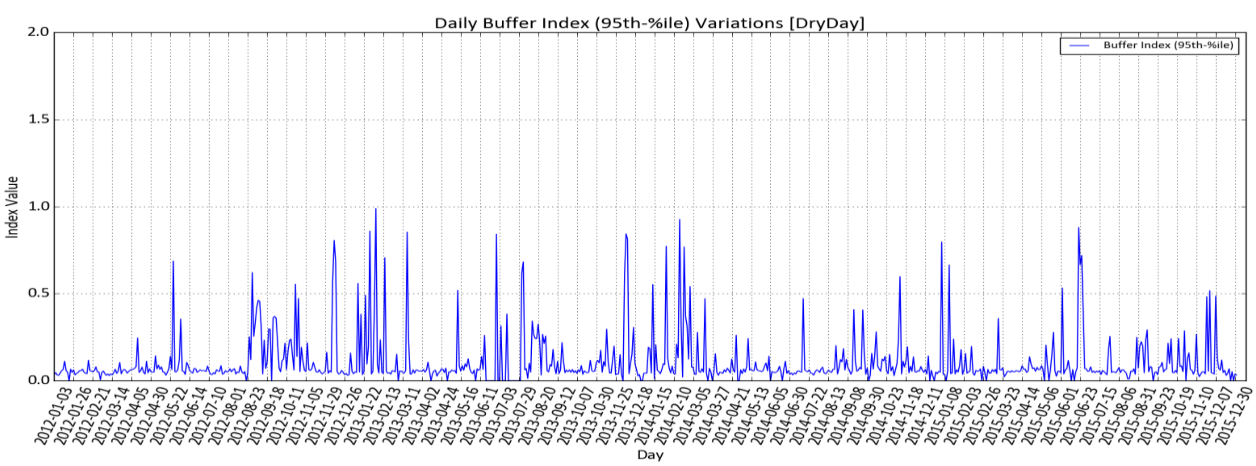


Figure 7.3.15 An Example Graph for Daily Buffer Index Variations

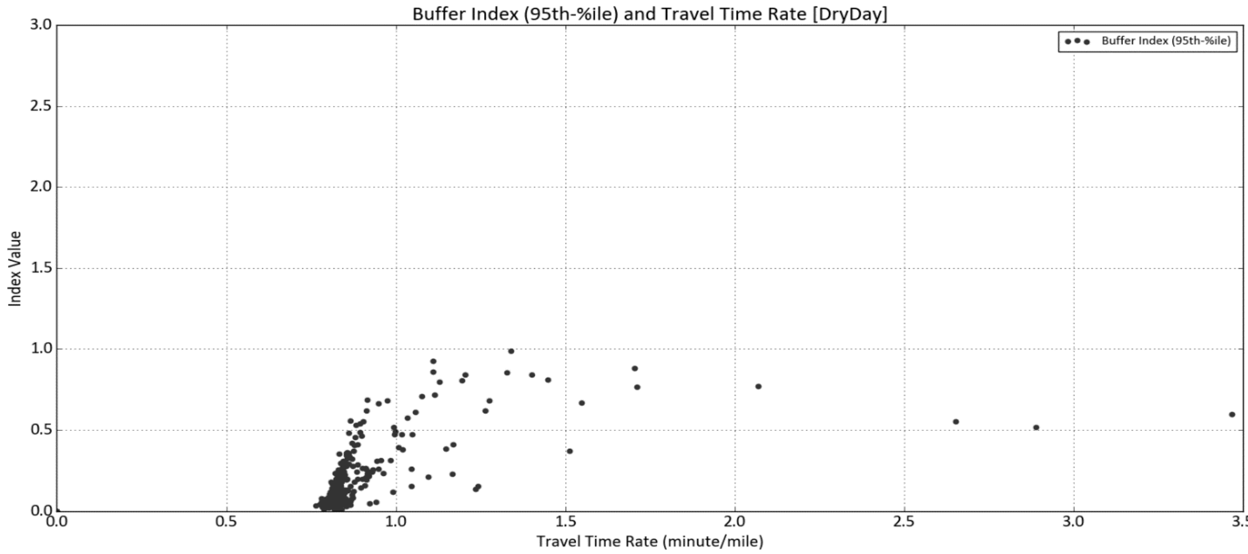


Figure 7.3.16 Variations of Daily Travel-Time Rate vs. Buffer-Index

Time of Day Reliability (TOD) Graph Writer creates TOD reliability graphs for each time interval for each operating condition with each data set, i.e., whole data set, yearly data set and monthly data set as shown in Figure 7.3.17-19. The graph types currently available are as follows:

- Travel time variations by time of day (Figure 7.3.17)
- Yearly TOD reliability indices (Figure 7.3.18)
- Monthly TOD reliability indices (Figure 7.3.19)

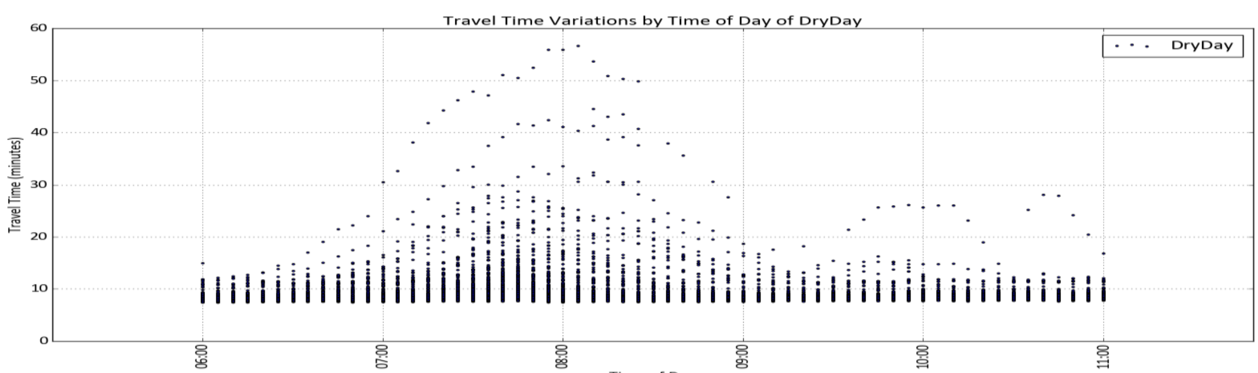


Figure 7.3.17 An Example graph for Travel-Time Variations by Time of Day

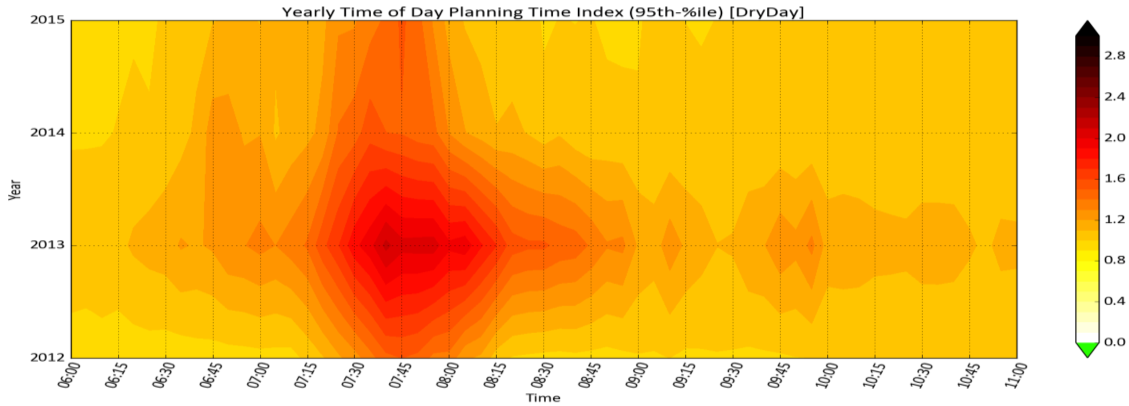


Figure 7.3.18 An Example graph for Yearly Planning-Time Index by Time of Day

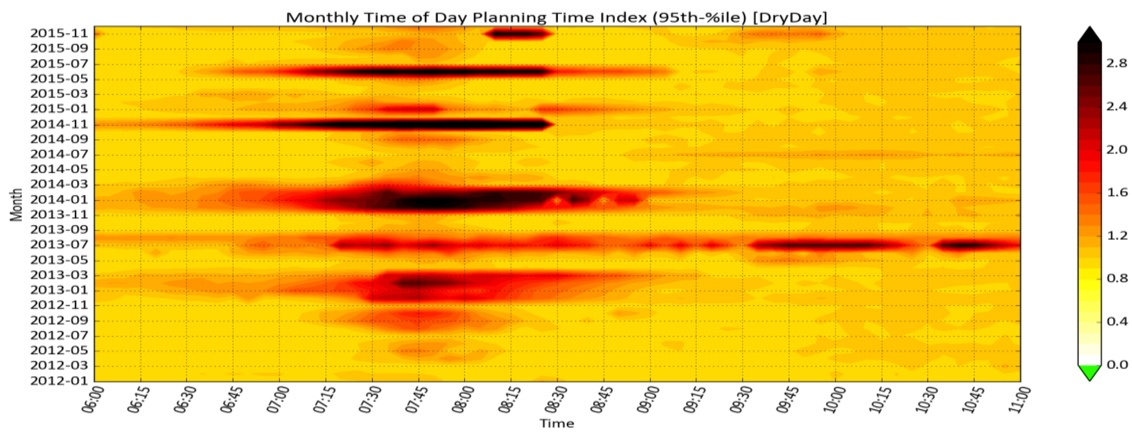


Figure 7.3.19 An Example graph for Monthly Planning-Time Index by Time of Day

CHAPTER 8: SYSTEM INTEGRATION AND TESTING

8.1 INTRODUCTION

In this chapter, all the individual modules developed in the previous chapters are integrated and the combined system is tested with real data from the metro freeway network in Twin Cities. To facilitate the integration and the data-exchange process among various modules, a set of the new functions and modules were developed. Figure 8.1.1 shows the locations and interrelationships of the new functions and modules, developed in this chapter, with the other modules in the TTRMS architecture.

First, the *Admin Client* module, developed in the previous chapter to manage non-traffic data, was enhanced with the new functions that can be used to configure the system parameters and to apply the updated data into the database. Next, the *Task Processing* module was developed for implementing the functions, including the procedures to handle the request from the *Admin Client*, to operate the entire system. The Admin Service module was also updated to connect the *Admin Client* and the *Task Processing* module. Finally, the Periodic *Data Processing* module was developed to perform the reserved tasks at pre-configured times on each day, week and month.

The functionalities of the major modules newly developed and/or enhanced in this chapter are as follows:

- *Admin Client* module, enhanced with a set of new functions:
  - provides the data-change log to the administrator.
  - sends the requests to apply the changed-data into the database,
  - configures the system parameters.
- *Periodic Data Processing* module, newly developed:
  - Performs daily, weekly and monthly tasks to prepare for travel-time data and time-of-day reliability measures, and to check missing data.
- *Task Processing* module, newly developed:
  - prepares for initial-data including weather, incident, travel times and categorization data for given-time period,
  - calculates time-of-day (TOD) reliability measures for all routes
  - calculates or categorizes travel time data for updated or inserted non-traffic data or route,

- checks if there are missing travel-time data.

The rest of this chapter summarizes the details of the above modules and the operating process of the integrated system, including initial data preparation, post processing for updated data and performing periodic tasks.

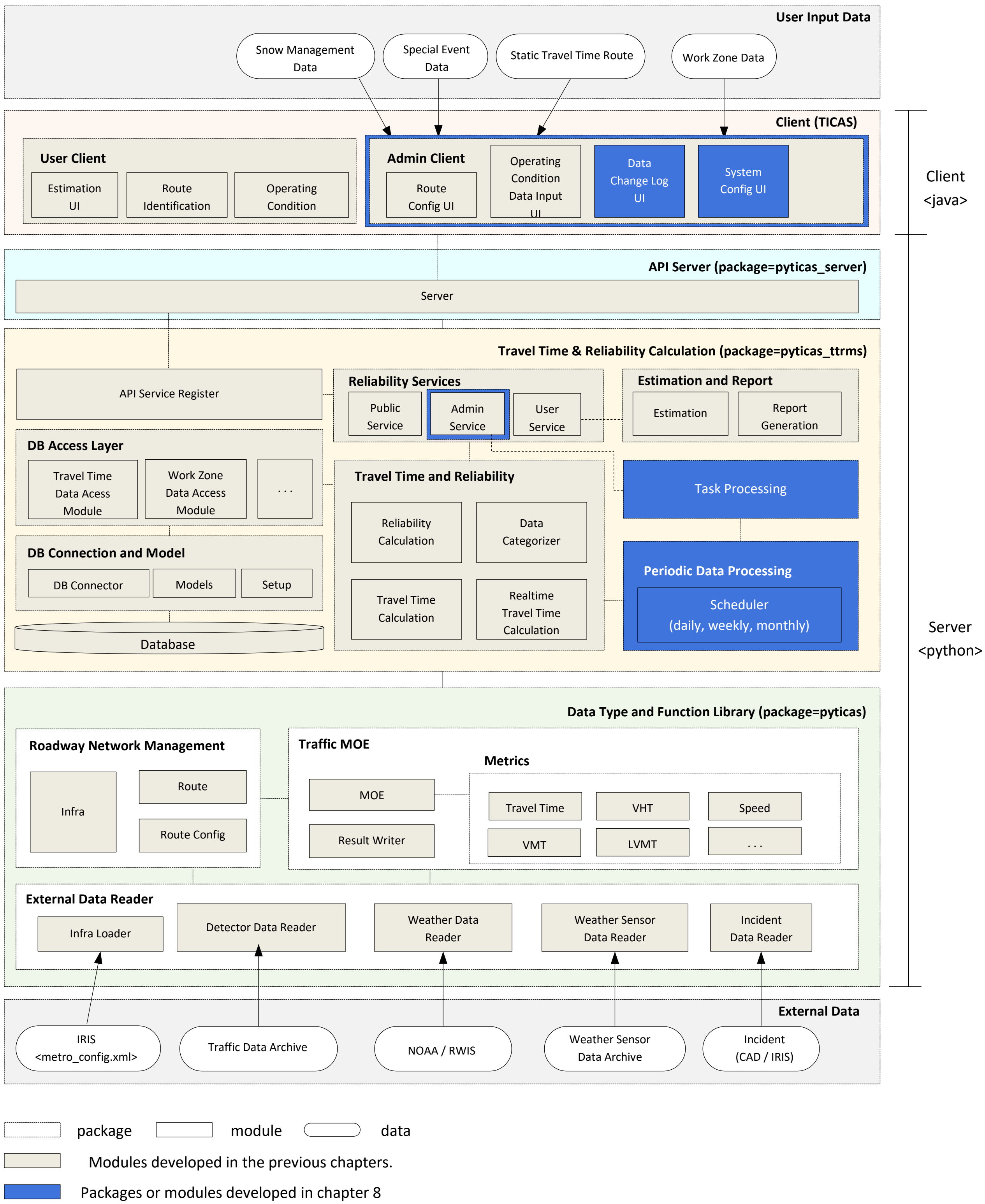


Figure 8.1.1 The Modules developed in Chapter 8 in the TTRMS Architecture

8.2 INTEGRATION OF THE ENTIRE SYSTEM

Figure 8.2.1 shows the modules developed and updated in this chapter for the system integration. The *Admin Client* module is updated with the new functions to configure the system parameters and manage the data changes by administrator. The *Task Processing* module is developed to process the requests from the *Admin Client* and the *Periodic Data Processing* module, which performs the daily, weekly and monthly tasks. The main modules in Figure 8.2.1 for calculating travel-time reliability measures have been developed in the previous chapters.

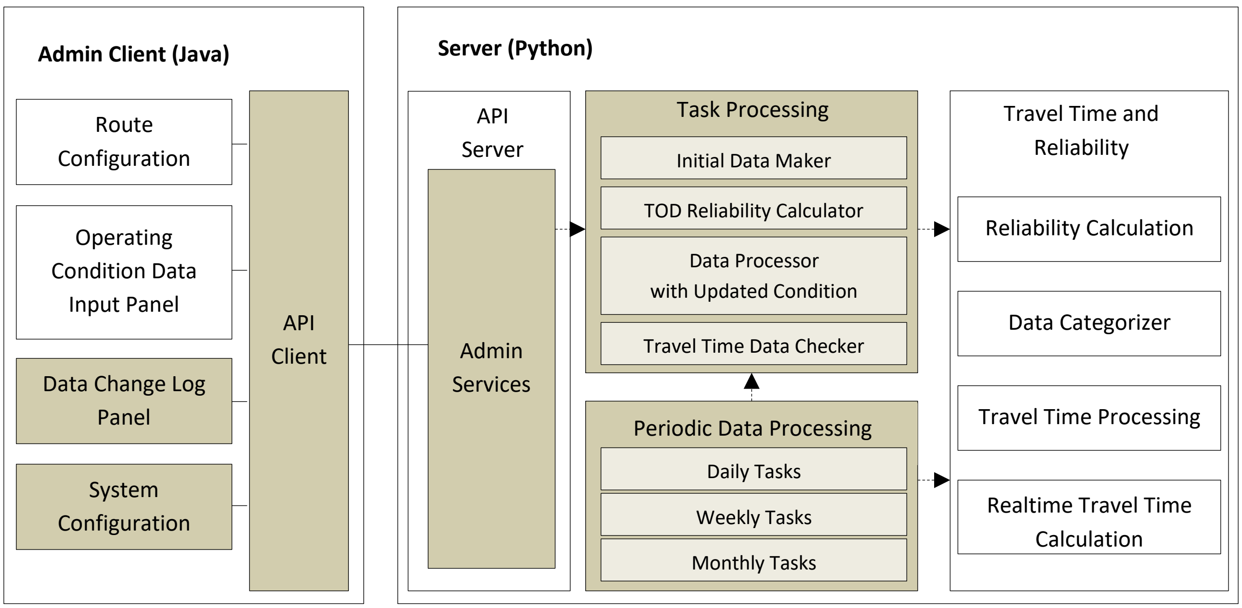


Figure 8.2.1 Updated and Added Modules for the System Integration

8.2.1 Enhancement of the Admin-Client and Admin-Service module

Figure 8.2.2 shows, the screen shot of the Admin Client, developed in the previous chapter, to manage the non-traffic data, which include travel-time route, work-zone, special-event and road condition during snow events. In this chapter, two new tabs, the *Data-Change Log Tab* and the *System-Configuration Tab,* are added to the Admin Client to facilitate the data-update and the system configuration processes. The main functionalities of each tab are as follows:

- *The **Data-Change Log Tab**,* shown in Figure 8.2.3, displays all the data-change activity logs, created in the *Admin Client* and stored in the database, in the following format:

  - Time: the time when data is changed

  - Action: type of data change action, i.e., “insert”, “delete”, or “update”

- Data: the description of changed data

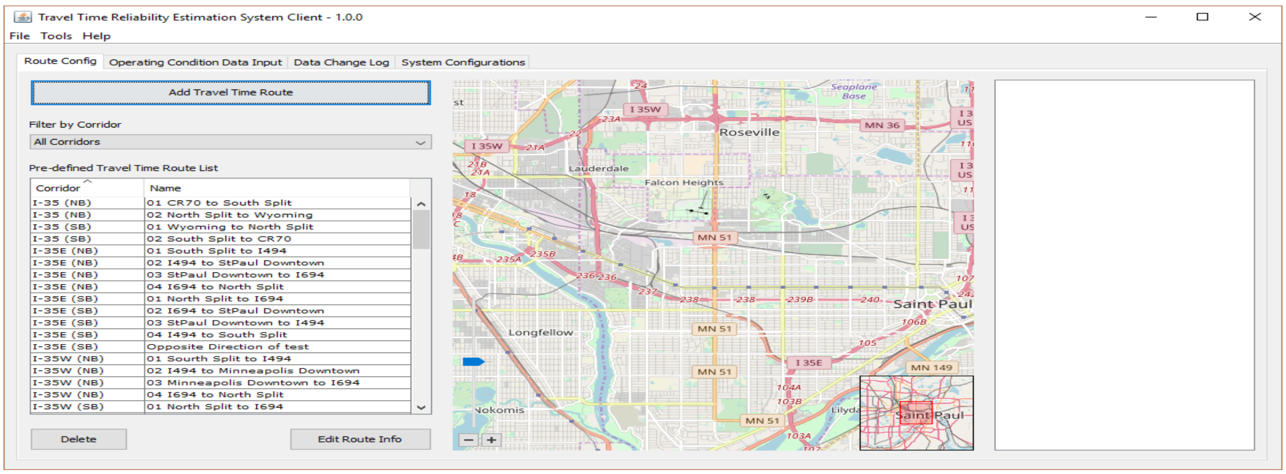
- Finished: flag showing whether the processing of the subject data change is finished or not

- Status: the status of the corresponding data processing.

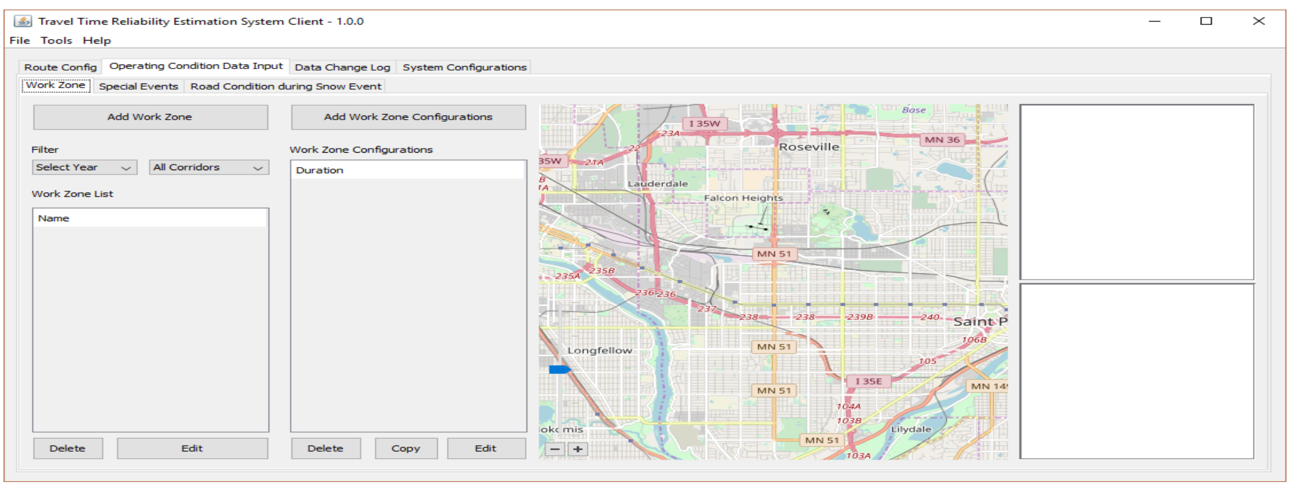
“in queue” indicates the corresponding data is waiting to be processed.

“running” denotes the changed data is being processed.

In this tab, the administrator can send the request to process all the changed data to the server by clicking “Update Database with Changed Data” button.



(a) Travel Time Route Configuration User Interface



(b) Operating Condition Data Input User Interface

Figure 8.2.2 User Interface of the Admin Client to Manage Non-Traffic Data

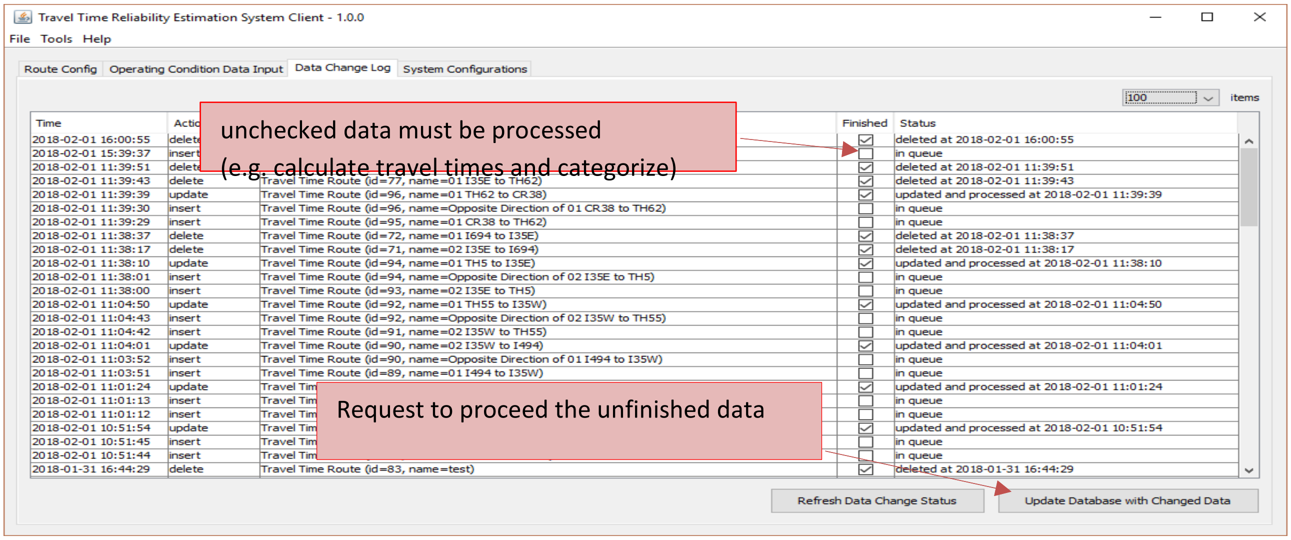


Figure 8.2.3 Data Change Log Management User Interface

- The *System-Configuration Tab,* shown in Figure 8.2.4, contains two sub-tabs to configure 1) the periodic-job scheduling and 2) the categorization parameters.

  - In the periodic-job setting tab, an administrator can set the following parameters:

    - *Data Archive-Start Year*: the start year to produce travel-time data for pre-defined routes.

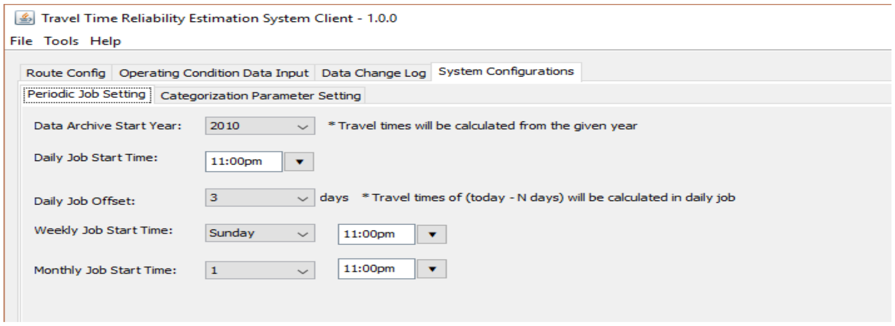
    - *Daily Job-Start Time*: the time when daily tasks are performed/

    - *Daily Job Offset*: the offset in number of days to calculate travel-time in daily tasks

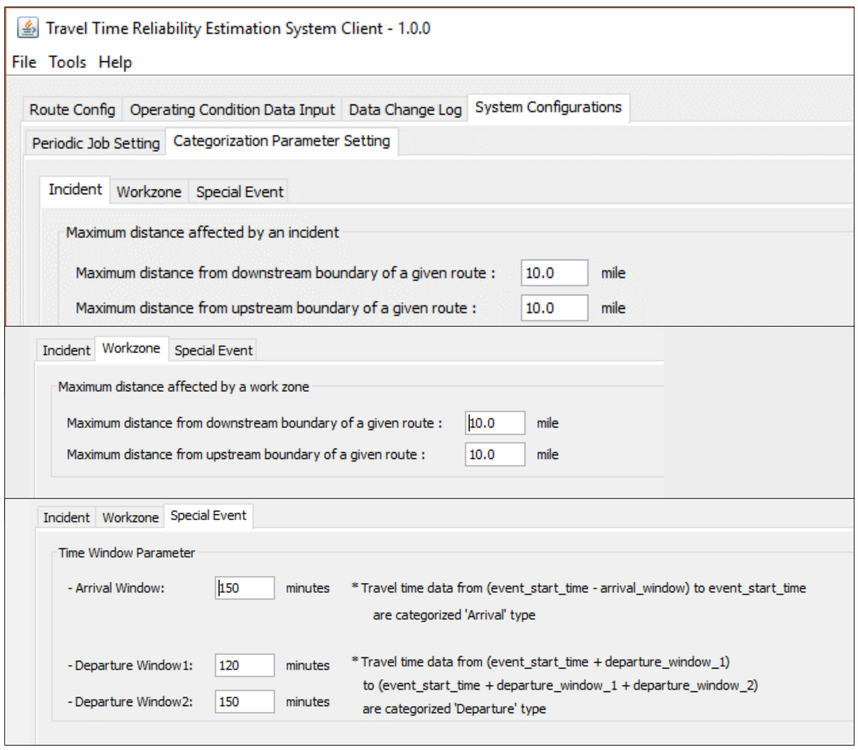
    - *Weekly Job Start Time*: the week day and time when weekly tasks are performed.

    - *Monthly Job Start Time*: the date and time when monthly tasks are performed.

  - In the categorization-parameter setting tab, an administrator can set the parameters used in categorization process as shown in Figure 8.2.4(b).



(a) Periodic-Job Configuration Tab



(b) Categorization-Parameter Configuration Tab

Figure 8.2.4 System-Configuration User Interface

8.2.2 Development of the Periodic Data-Processing module

Figure 8.2.5 shows the structure of the *Periodic Data-Processing* module, which is responsible for executing the tasks to calculate and categorize travel times at scheduled times for all pre-defined routes. This module interacts with *Admin Services*, *Task Processing* and *Travel Time and Reliability* modules. Further, all the functions that perform the tasks are implemented in the *Task Processing* and *Travel Time and Reliability* modules. The operational sequence of the *Periodic Data Processing* module is as follows:

1) Task-Scheduler and Worker-Process modules are started during the boot sequence of the system.

2) Task-Scheduler puts daily, weekly and monthly tasks to the shared queue with the Worker Process at each scheduled time.

3) Worker-Process, a separated process from the server process, receives the tasks from the shared queue and executes the tasks using the *Task Processing* and *Travel Time and Reliability* module.

4) Time schedules are updated by the *Admin Client* module. The updated-time information is saved in the database and applied to currently running scheduler through the *Admin Services* module.

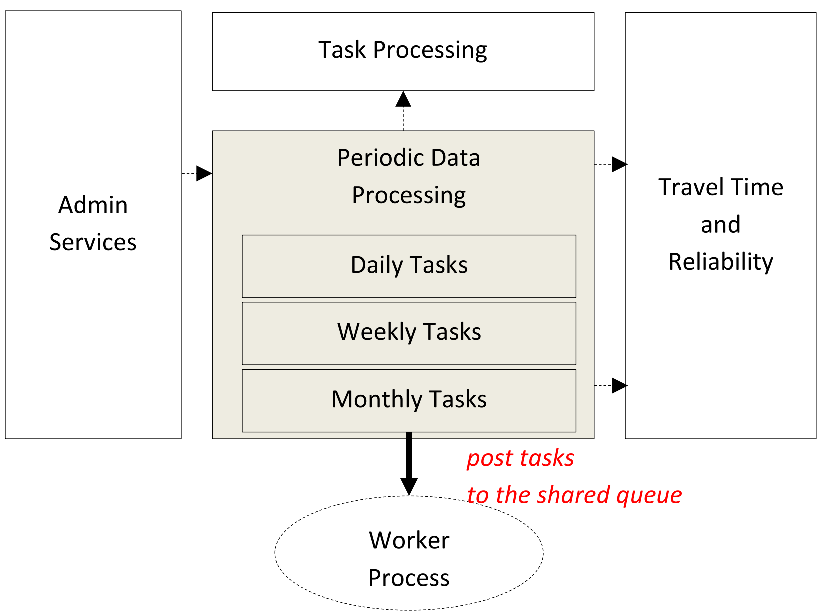


Figure 8.2.5 Periodic Data Processing module

Daily-Task Processing

In the current version of TTRMs, the travel times for all pre-defined routes are calculated on a daily basis and those travel times calculated for every day are categorized according to the operating conditions, such as weather and incident, for each day. Further, the incident and weather data for each day are loaded automatically before the categorization of the travel times is performed. Figure 8.2.6 shows the sub-modules for processing the daily tasks, whose operational sequence is as follows:

- *Scheduler* module runs the *Daily-Task* module, which calls the individual task modules in the *Daily Tasks* module in a specific order.
- Individual task modules use the *Task Processing* module and *Travel Time and Reliability* module developed in the previous tasks

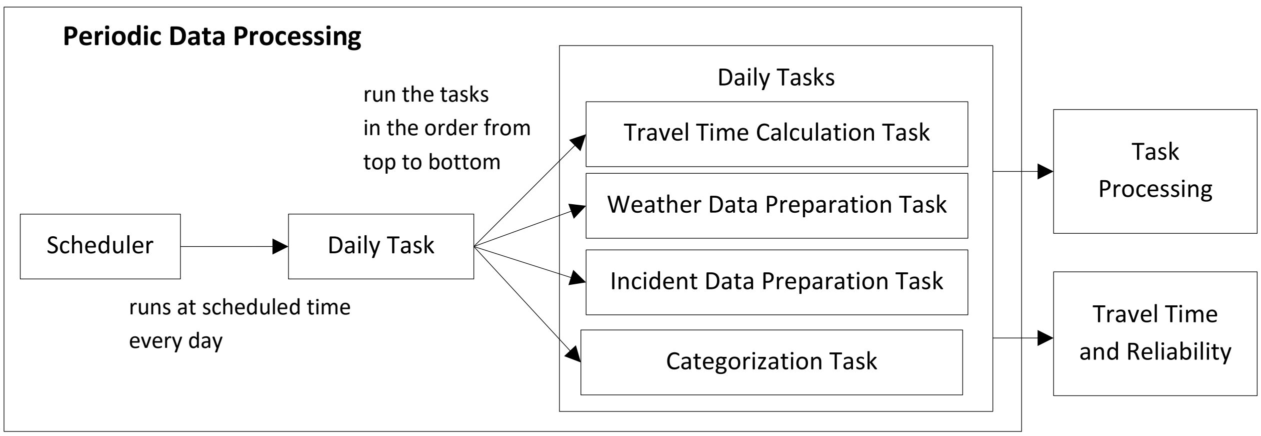


Figure 8.2.6 Sub-modules in the Daily-Tasks module in Periodic Data Processing

Weekly-Task Processing

In the current version of TTRMS, the Time-of-Day (TOD) reliability measures for all pre-defined routes are calculated on a weekly basis. Those TOD reliability measures are used as the basis for the Travel-Time Information Service, which is designed to provide public an expected travel time d for a selected route in real time. Figure 8.2.7 shows the structure and operational sequence for the Weekly-Tasks module, whose main work is performed by the *TOD Reliabilities Pre-Calculation Task* module.

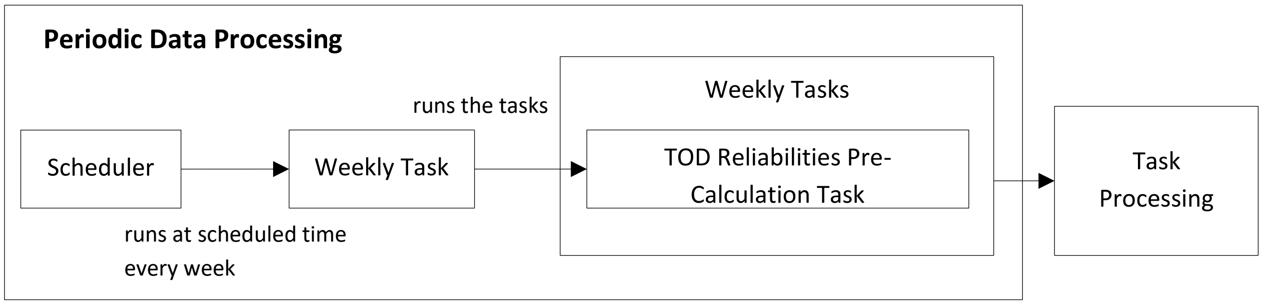


Figure 8.2.7 Structure and Sequence for Weekly-Task processing in Periodic Data Processing module

Monthly-Task Processing

The main function of the Monthly-Task module is to check if there is any missing data in the travel times calculated by the Daily-Tasks module. If any missing data are found, it runs the travel-time calculation and categorization procedures. Figure 8.2.8 shows the structure and operational sequence of the Monthly-Tasks module.

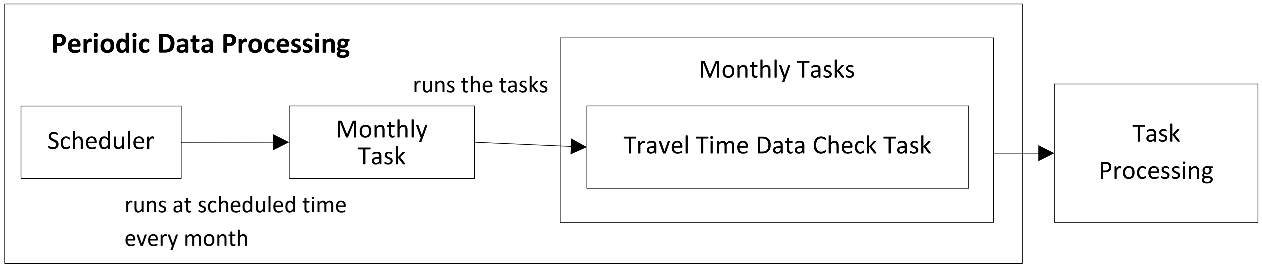


Figure 8.2.8 Structure and Operational Sequence of the Monthly-Task module in Periodic Data Processing

8.2.3 Development of the Task-Processing module

Figure 8.2.9 shows the structure of the Task-Processing module, whose main responsibility is to execute the functions that operate the entire system. The main functionalities of each submodule are as follows:

***Initial-Data Maker* module** calculates the travel-times for all predefined routes and categorizes them for given time periods before the entire system is executed.

- In the current version, the weather and incident data are imported automatically from the CAD/IRIS database and the NOAA data archives by the *Incident-Data Loader* and the *NOAA-Weather Data Loader* submodules before the travel-time data are categorized, while other non-traffic data, such as work zone, special event and road condition during snow events, are entered manually by administrator. The structure of those two submodules is shown in Figure 8.2.10.

-The calculation and categorization of the travel-time data is performed by the *Travel-Time and Reliability* module developed in the previous chapters.

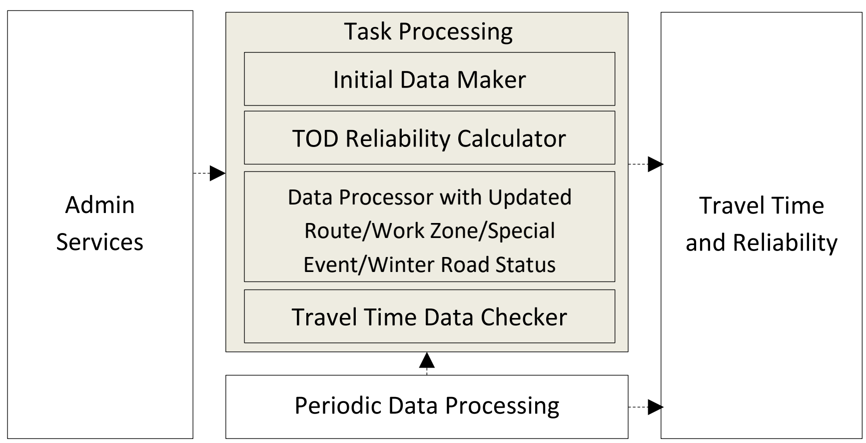


Figure 8.2.9 Structure of the Task-Processing module

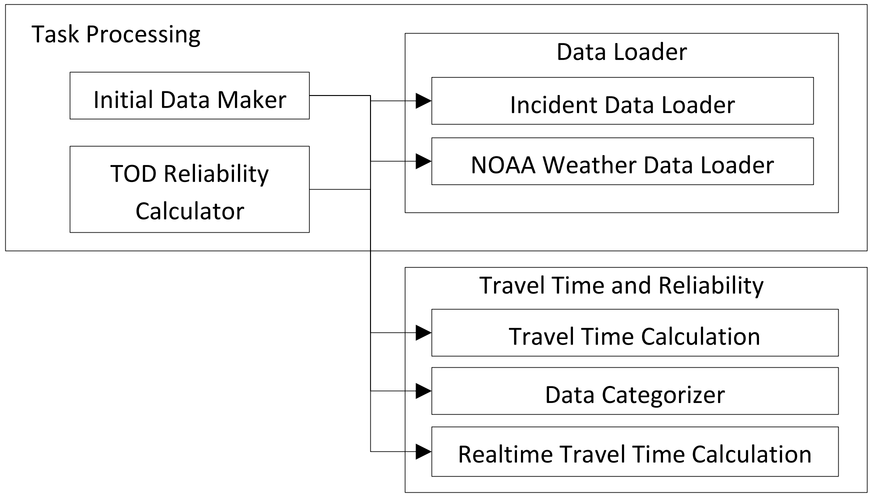


Figure 8.2.10 Sub-modules in the Initial-Data Mark module

TOD Reliability Calculator module calculates the time of day (TOD) reliability measures for given dates and for all predefined routes by using the *pre-calculation* functions of the *Real-Time Travel-Time Calculation* module.

Data-Processor module calculates and categorizes travel-time data with updated/inserted non-traffic data, such as work zone, special event and winter-road status, and also for updated routes. Figure 8.2.11 shows the flow chart of the process of this module, which has the following sequence:

1) Retrieving the data-change logs to be processed from the database.

2) Extracting the change item from the database according to the data-change log.

3) Preparing the handler to process the retrieved data.

4) Running the fetched handler with the retrieved data item

The above process is iterated for all data-change logs.

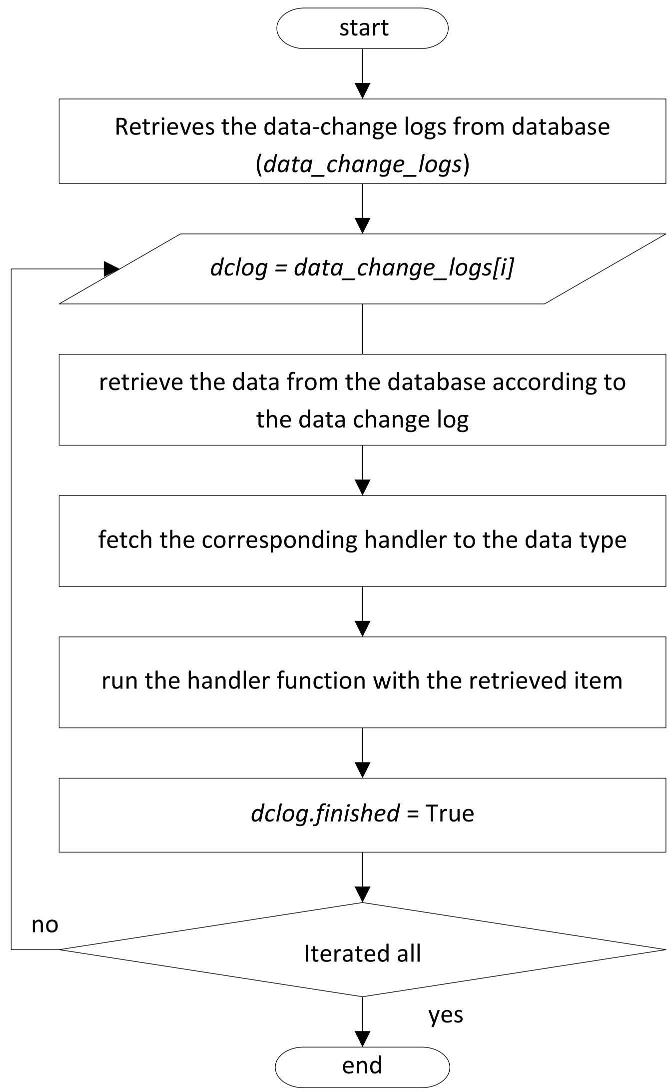


Figure 8.2.11 Flow Chart of Data Processor module

Travel-Time Data Checker module checks if there are missing travel-time data, and, if there are, it calculates and categorizes the travel-time data for those missing data.

- Checking any missing data is performed by comparing the expected number of the travel-time data for given time periods and the number of stored travel-time data. It first divides the whole data set into yearly, monthly and daily data sets and checks the existence of any missing travel-time data in a sequential manner, as shown in Figure 8.2.11.

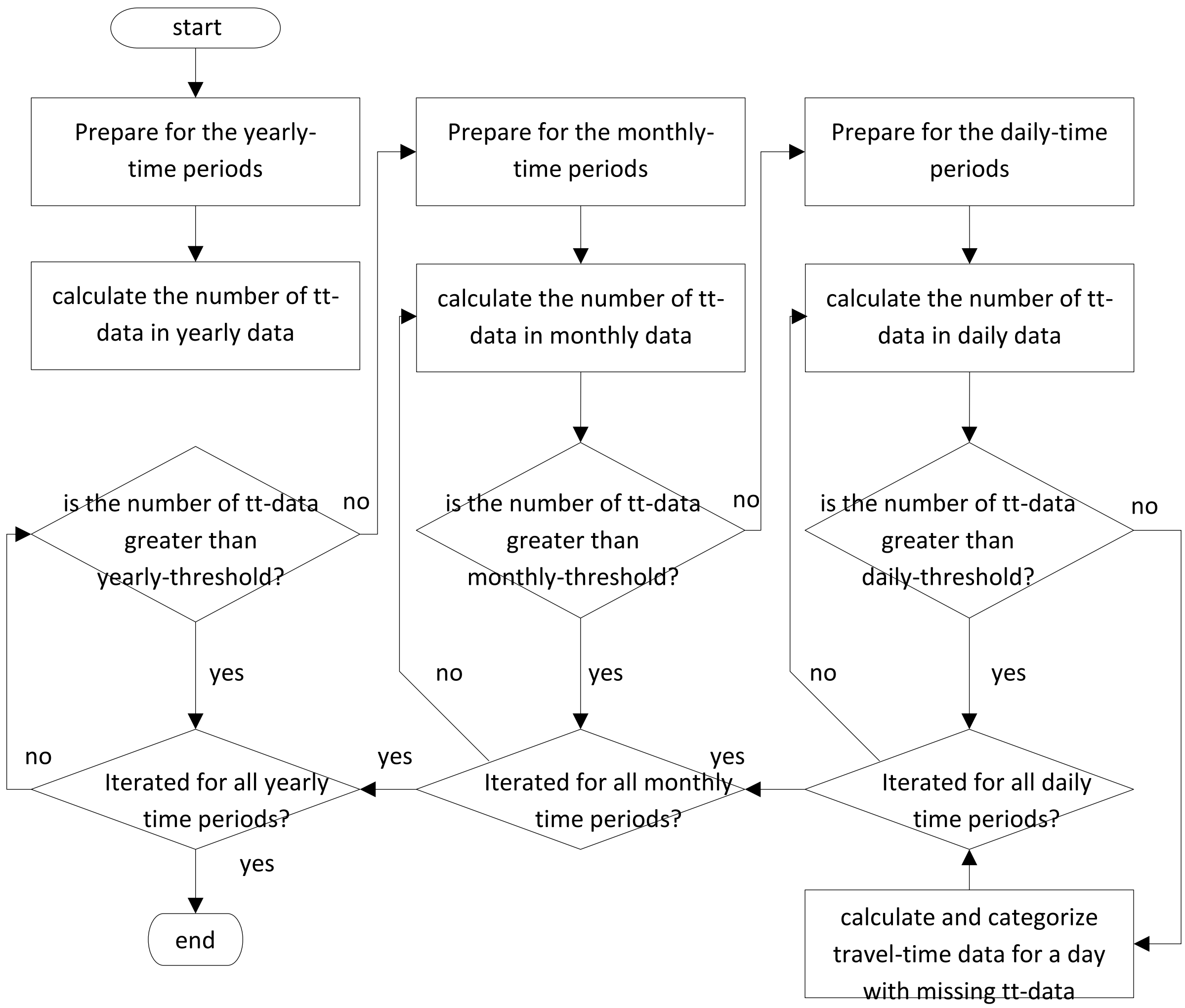


Figure 8.2.12 Flow Chart of Travel Time Data Checker module

8.3 OPERATING PROCESS OF THE INTEGRATED SYSTEM

Figure 8.3.1 shows the operating process of the integrated system, which has the following sequence:

1) Editing of non-traffic data by the Admin Client, where insert/update/delete-operations of routes, weather, incident, work-zone and road-condition data are performed.

2) Performing a set of scheduled tasks in the Server.

3) User-specification of operating conditions in the User Client and Calculation of the travel-time reliability measures for given conditions.

4) Travel-Time information service based on time-of-day reliability measures.

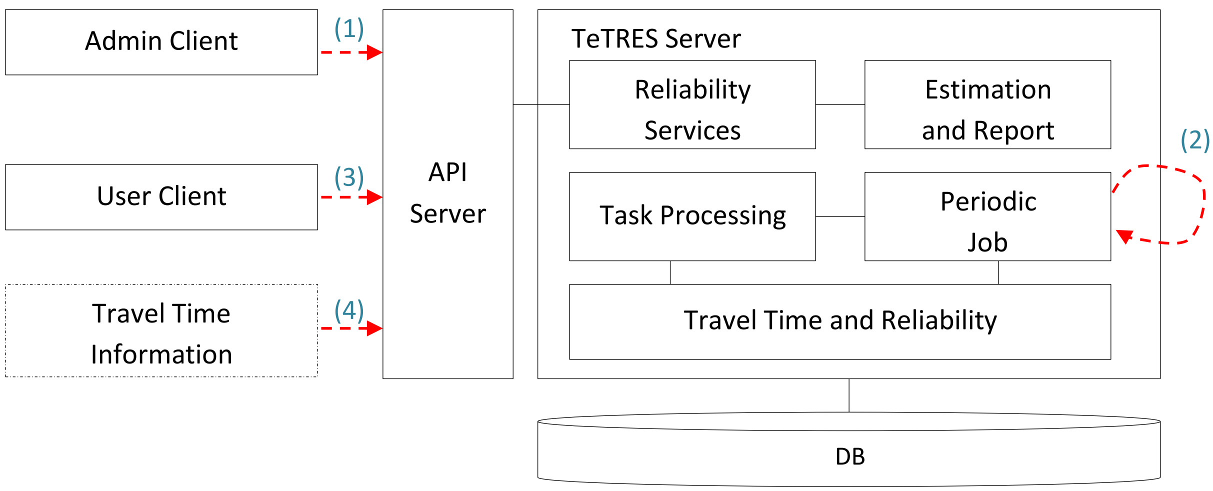


Figure 8.3.1 Operating Process of the Integrated System

Editing operations of non-traffic data in the Admin Client

In the current version of TTRMS, an administrator manages the non-traffic data, such as travel-time reliability routes, work zone, special event and road condition data, which are entered manually. Further, the system parameters, e.g., periodic job schedule, are configured in the *Admin Client,* whose screen shot is shown in Figure 8.3.2. In addition, the administrator can send the request to process the changed data to the server.

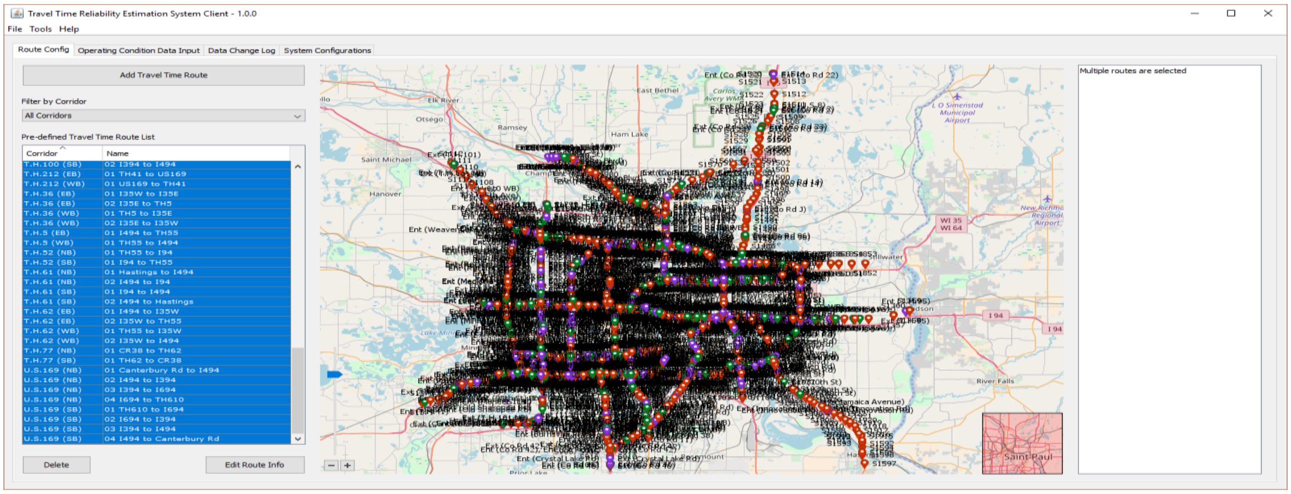


Figure 8.3.2 Screen Shot of Admin Client

Performing Scheduled Tasks in the server

The scheduled tasks described in the previous chapter are executed automatically at specified time schedules:

Daily-tasks Processing: The calculation and categorization of the travel times for all pre-defined routes are performed on a daily basis after the weather data from NOAA and the incident data from CAD and IRIS are imported.

Weekly-tasks Processing: The calculation of Time of Day (TOD) reliability measures are conducted on a weekly basis and those TOD reliability measures are stored into the database for all pre-defined routes.

Monthly-tasks Processing: The examination of any missing travel-time data is performed every month. If any travel-time data are missing, they are calculated and categorized by appropriate functions.

Travel-time reliability estimation with the user client

The travel-time reliability estimation process is triggered by the user client, whose screen shot is shown in Figure 8.3.3. The user client enables a user to set the parameters required to estimate reliability measures. Those parameters include routes, time periods, types of reliability measures, output types and operating conditions. Once user specifies those parameters, the server executes the estimation process and produces output in the format of spreadsheets and graphs.

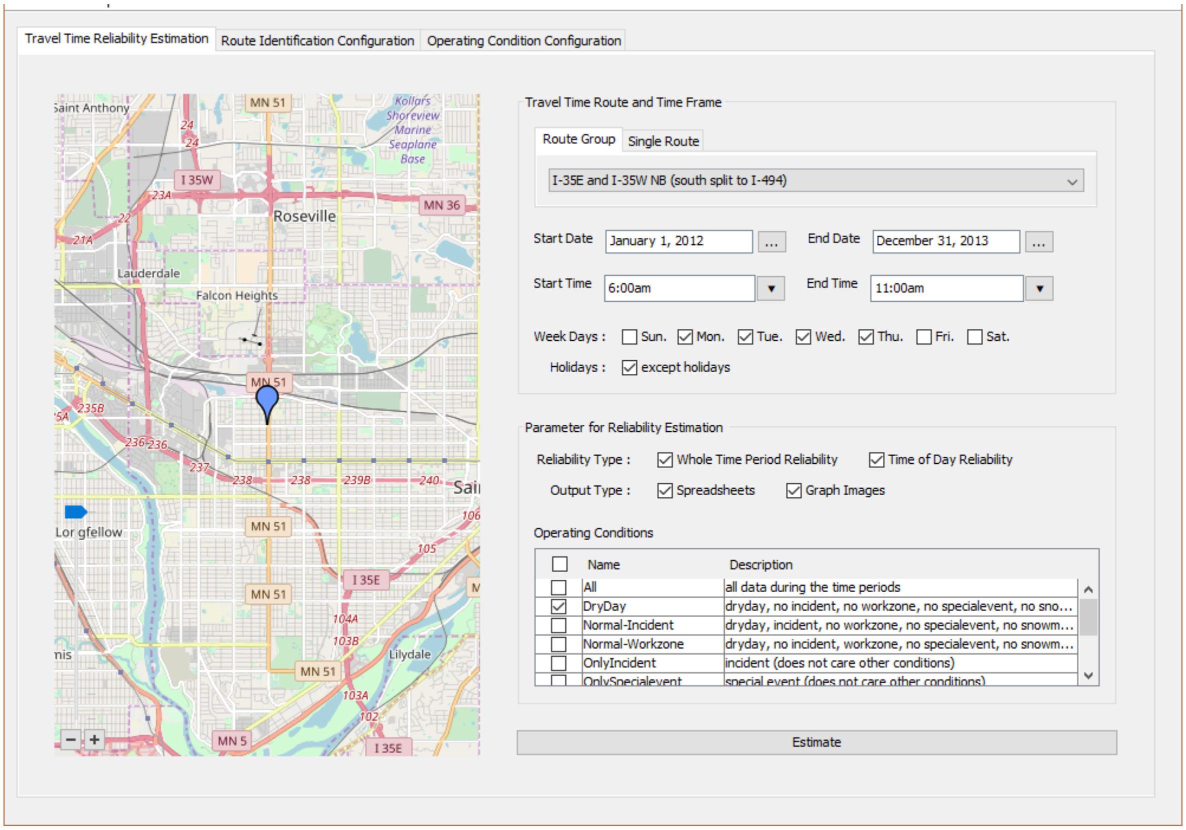


Figure 8.3.3 Screen Shot of User Client

Travel-time information service

As part of the TTRMS operations, a web page has been developed to investigate the feasibility of providing public expected travel-times for selected routes in real time. The expected travel-times are determined with the time-of-day reliability measures, which are generated by TTRMS. Figure 8.3.4 shows a screenshot of the current version of the web page, where user selects a route and enters a departure time. Then the server calculates expected travel-time for user-selected route and departure time using the time-of-day reliability measures.

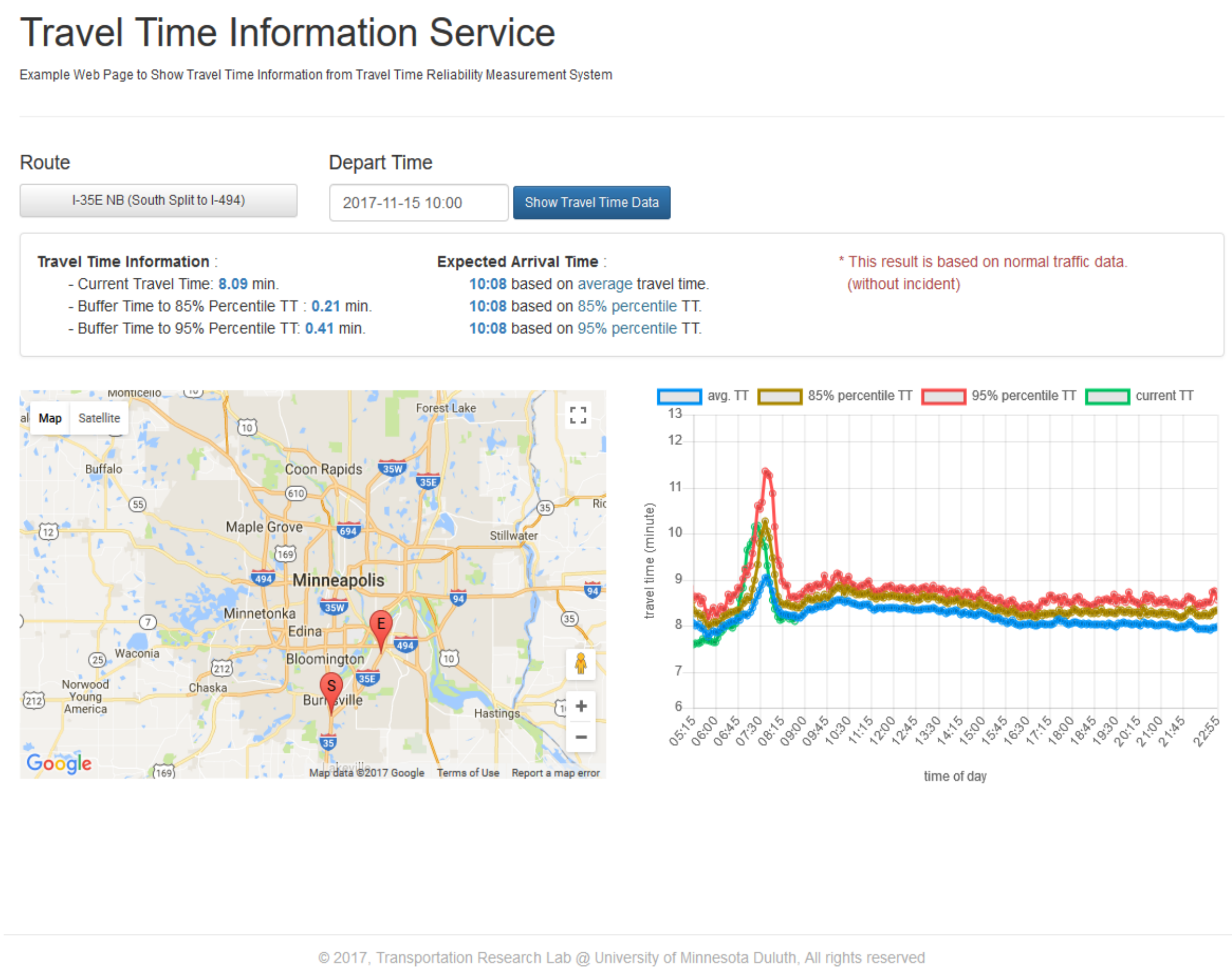


Figure 8.3.4 Example Web Page for Travel-Time Information Service

8.4 TESTING OF THE INTEGRATED SYSTEM

The integrated system is tested with real data from the metro freeway network in Twin Cities for a two-year period, i.e., from January 1, 2012, to December 31, 2013. For this testing, a total of 87 travel-time routes have been defined covering most of the metro freeway network. Figure 8.4.1 shows the freeway sections used as the basis for defining the travel-time routes in this testing, i.e., two directional routes are defined for each freeway section.

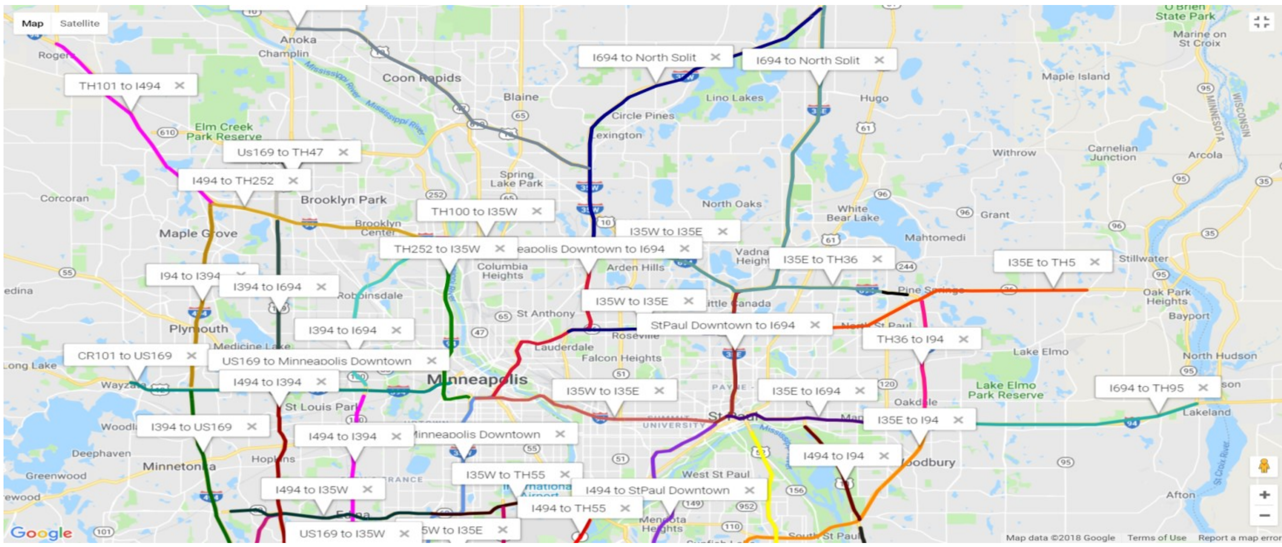


Figure 8.4.1 Freeway Sections used for defining Travel-Time Routes

In this report, the reliability estimation results for the following four routes are included. The estimation results for the other routes are available upon request.

- Selected Routes:

  - I-35E NB and I-35W NB from south split to I-494

  - U.S.169 NB and T.H.100 NB from I-494 to I-394

- Duration: 1/1/2012 -12/31/2013 (2 years)
- Time Period for Reliability Estimation: 05:00 - 11:00 a.m.
- Dates included: Tuesday - Thursday (except holiday)

Table 8.4.1 shows the types of the operating conditions used for categorizing the travel-time data for this testing. The reliability estimation results for the above 4 routes are presented in Tables 8.4.2 – 8.4.17 for each operating condition. Further, the graphical presentations of those reliability estimates are shown in Figures 8.4.2-8.2.29, which can be used for comparing the reliability measures for different operating conditions as well as analyzing the variations of reliability measures through time.

Table 8.4.1 Operating-Condition Types used in the Test

Index	Name	Description
0	All	all data during the time periods
1	DryDay	dryday, no incident, no work zone, no special event, no winter road condition
2	Normal-Incident	dryday, incident, no work zone, no special event, no winter road condition
3	Normal-Workzone	dryday, no incident, work zone, no special event, no winter road condition

Estimation Results for I-35E NB Route

Table 8.4.2 Estimated Reliability Measures by Operating Conditions of I-35E NB Route

Operating Condition (OC) Index	**0**	**1**	**2**	**3**
OC Name	All	DryDay	Normal-Incident	Normal-Workzone
Avg TT	9.39	9.05	9.50	9.87
Travel Time Rate (minute/mile)	0.85	0.82	0.86	0.90
Data Count	22191.00	13884.00	2448.00	4416.00
Free-Flow TT using Speed Limit	9.43	9.43	9.43	9.43
Congested Avg. TT	16.66	18.12	18.46	15.71
Congested Data Count	1128.00	261.00	124.00	435.00
80th %-ile TT	9.38	9.13	9.31	10.22
85th %-ile TT	9.62	9.23	9.46	10.98
90th %-ile TT	10.21	9.39	9.95	12.20
95th %-ile TT	12.31	9.87	12.31	15.07
Buffer Index (80th %-ile)	-	0.01	-	0.04
Buffer Index (85th %-ile)	0.02	0.02	-	0.11
Buffer Index (90th %-ile)	0.09	0.04	0.05	0.24
Buffer Index (95th %-ile)	0.31	0.09	0.30	0.53
Planning Time Index (80th %-ile)	0.99	0.97	0.99	1.08
Planning Time Index (85th %-ile)	1.02	0.98	1.00	1.16
Planning Time Index (90th %-ile)	1.08	1.00	1.06	1.29
Planning Time Index (95th %-ile)	1.31	1.05	1.31	1.60
Travel Time Index	1.77	1.92	1.96	1.67
Level of Travel Time Reliability	1.06	1.04	1.04	1.13
Semi-Variance	13.41	8.50	24.64	9.28
Semi-Variance Data Count	4335.00	3549.00	351.00	1094.00
On-Time Arrival	0.96	0.98	0.96	0.93
On-Time Arrival Data Count	21318.00	13668.00	2338.00	4115.00
Misery Index	1.64	1.20	1.82	1.87

Table 8.4.3 Yearly Reliability Measures of Operating Condition Type "All" of I-35E NB

Year	2012	2013
Avg TT	9.04	9.75
Travel Time Rate (minute/mile)	0.82	0.88
Data Count	11169.00	11022.00
Free-Flow TT using Speed Limit	9.43	9.43
Congested Avg. TT	15.48	18.02
Congested Data Count	261.00	866.00
80th %-ile TT	9.15	9.77
85th %-ile TT	9.25	10.23
90th %-ile TT	9.44	11.49
95th %-ile TT	10.19	14.37
Buffer Index (80th %-ile)	0.01	0.00
Buffer Index (85th %-ile)	0.02	0.05
Buffer Index (90th %-ile)	0.04	0.18
Buffer Index (95th %-ile)	0.13	0.47
Planning Time Index (80th %-ile)	0.97	1.04
Planning Time Index (85th %-ile)	0.98	1.09
Planning Time Index (90th %-ile)	1.00	1.22
Planning Time Index (95th %-ile)	1.08	1.52
Travel Time Index	1.64	1.80
Level of Travel Time Reliability	1.04	1.09
Semi-Variance	3.93	18.13
Semi-Variance Data Count	3030.00	2252.00
On-Time Arrival	0.98	0.94
On-Time Arrival Data Count	10966.00	10368.00
Misery Index	1.27	1.89

Table 8.4.4 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of I-35E NB

Year	2012											
Month	1	2	3	4	5	6	7	8	9	10	11	12
Avg TT	8.96	9.03	8.78	8.92	9.00	8.92	8.86	9.06	9.28	9.26	8.85	9.61
Travel Time Rate (minute/mile)	0.81	0.82	0.80	0.81	0.82	0.81	0.80	0.82	0.84	0.84	0.80	0.87
Data Count	949	949	949	876	1095	876	876	1022	876	1022	876	803
Free-Flow TT using Speed Limit	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43

Congested Avg. TT	18.92	14.05	15.19	15.34	15.78	13.61	-1.00	15.14	13.38	16.75	-1.00	16.84
Congested Data Count	18.00	33.00	9.00	6.00	18.00	6.00	0.00	28.00	49.00	30.00	0.00	64.00
80th %-ile TT	9.03	9.15	8.92	9.14	9.18	9.14	9.09	9.14	9.33	9.35	9.07	9.46
85th %-ile TT	9.12	9.29	9.01	9.21	9.26	9.21	9.17	9.24	9.57	9.51	9.17	10.10
90th %-ile TT	9.27	9.59	9.10	9.32	9.43	9.36	9.26	9.38	10.75	9.87	9.29	11.04
95th %-ile TT	9.48	10.67	9.24	9.51	9.74	9.64	9.45	10.08	12.42	11.18	9.59	14.36
Buffer Index (80th %-ile)	0.01	0.01	0.02	0.03	0.02	0.02	0.03	0.01	0.01	0.01	0.02	0.00
Buffer Index (85th %-ile)	0.02	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.03	0.03	0.04	0.05
Buffer Index (90th %-ile)	0.03	0.06	0.04	0.04	0.05	0.05	0.04	0.04	0.16	0.07	0.05	0.15
Buffer Index (95th %-ile)	0.06	0.18	0.05	0.07	0.08	0.08	0.07	0.11	0.34	0.21	0.08	0.49
Planning Time Index (80th %-ile)	0.96	0.97	0.95	0.97	0.97	0.97	0.96	0.97	0.99	0.99	0.96	1.00
Planning Time Index (85th %-ile)	0.97	0.99	0.96	0.98	0.98	0.98	0.97	0.98	1.02	1.01	0.97	1.07
Planning Time Index (90th %-ile)	0.98	1.02	0.96	0.99	1.00	0.99	0.98	0.99	1.14	1.05	0.99	1.17
Planning Time Index (95th %-ile)	1.01	1.13	0.98	1.01	1.03	1.02	1.00	1.07	1.32	1.19	1.02	1.52
Travel Time Index	1.90	1.49	1.61	1.63	1.67	1.44	-1.00	1.61	1.42	1.78	-1.00	1.79
Level of Travel Time Reliability	1.03	1.05	1.03	1.04	1.04	1.04	1.03	1.04	1.05	1.05	1.04	1.07
Semi-Variance	6.42	2.71	1.13	0.94	2.39	0.50	0.13	3.60	2.75	6.80	0.22	16.59
Semi-Variance Data Count	243	263	343	330	358	335	387	267	195	240	342	143
On-Time Arrival	0.98	0.97	0.99	0.99	0.99	1.00	1.00	0.98	0.97	0.98	1.00	0.94
On-Time Arrival Data Count	932	919	942	870	1080	872	876	999	854	998	876	753
Misery Index	1.07	1.43	1.01	1.10	1.10	1.07	1.03	1.33	1.41	1.38	1.06	1.94

Table 8.4.5 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of I-35E NB

Year	**2013**											
Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Avg TT	9.50	9.72	9.62	9.45	9.04	10.06	11.96	9.80	9.00	8.95	8.99	11.13
Travel Time Rate (minute/mile)	0.86	0.88	0.87	0.86	0.82	0.91	1.08	0.89	0.82	0.81	0.82	1.01
Data Count	1021	876	876	949	1022	876	949	876	876	1095	803	803
Free-Flow TT using Speed Limit	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43	9.43
Congested Avg. TT	16.64	19.59	18.83	20.81	14.04	14.09	16.63	14.67	12.64	15.80	12.33	19.51
Congested Data Count	73	62	67	42	17	101	292	71	5	8	1	127
80th %-ile TT	9.35	9.34	9.22	9.34	9.30	10.90	15.07	10.83	9.20	9.12	9.23	11.31
85th %-ile TT	9.58	9.61	9.48	9.53	9.42	11.47	16.57	11.79	9.31	9.21	9.37	12.54
90th %-ile TT	10.26	10.21	11.61	9.83	9.57	12.66	18.96	12.11	9.49	9.34	9.57	15.67
95th %-ile TT	14.13	14.40	13.76	11.48	10.00	13.81	19.50	12.65	9.97	9.65	10.02	21.27
Buffer Index (80th %-ile)	0.00	0.00	0.00	0.00	0.03	0.08	0.26	0.11	0.02	0.02	0.03	0.02
Buffer Index (85th %-ile)	0.01	0.00	0.00	0.01	0.04	0.14	0.38	0.20	0.03	0.03	0.04	0.13
Buffer Index (90th %-ile)	0.08	0.05	0.21	0.04	0.06	0.26	0.50	0.24	0.05	0.04	0.06	0.41
Buffer Index (95th %-ile)	0.49	0.48	0.43	0.21	0.11	0.37	0.63	0.29	0.11	0.08	0.11	0.91
Planning Time Index (80th %-ile)	0.99	0.99	0.98	0.99	0.99	1.16	1.60	1.15	0.98	0.97	0.98	1.20
Planning Time Index (85th %-ile)	1.02	1.02	1.01	1.01	1.00	1.22	1.76	1.25	0.99	0.98	0.99	1.33
Planning Time Index (90th %-ile)	1.09	1.08	1.23	1.04	1.01	1.34	1.90	1.28	1.01	0.99	1.02	1.66
Planning Time Index (95th %-ile)	1.50	1.53	1.46	1.22	1.06	1.46	2.07	1.34	1.06	1.02	1.06	2.26
Travel Time Index	1.76	2.08	1.89	2.21	1.49	1.49	1.76	1.56	1.34	1.68	1.31	2.07
Level of Travel Time Reliability	1.05	1.05	1.05	1.06	1.04	1.15	1.48	1.20	1.04	1.03	1.04	1.20

Semi-Variance	16.61	41.46	28.31	30.88	1.18	3.37	8.69	6.39	0.51	1.19	0.35	41.33
Semi-Variance Data Count	162	110	123	167	383	296	300	220	318	365	298	176
On-Time Arrival	0.94	0.94	0.94	0.96	0.99	0.96	0.81	0.97	1.00	0.99	1.00	0.88
On-Time Arrival Data Count	959	823	823	911	1010	838	768	846	876	1087	803	706
Misery Index	1.80	2.28	1.94	1.96	1.16	1.65	2.21	1.57	1.13	1.11	1.11	2.91

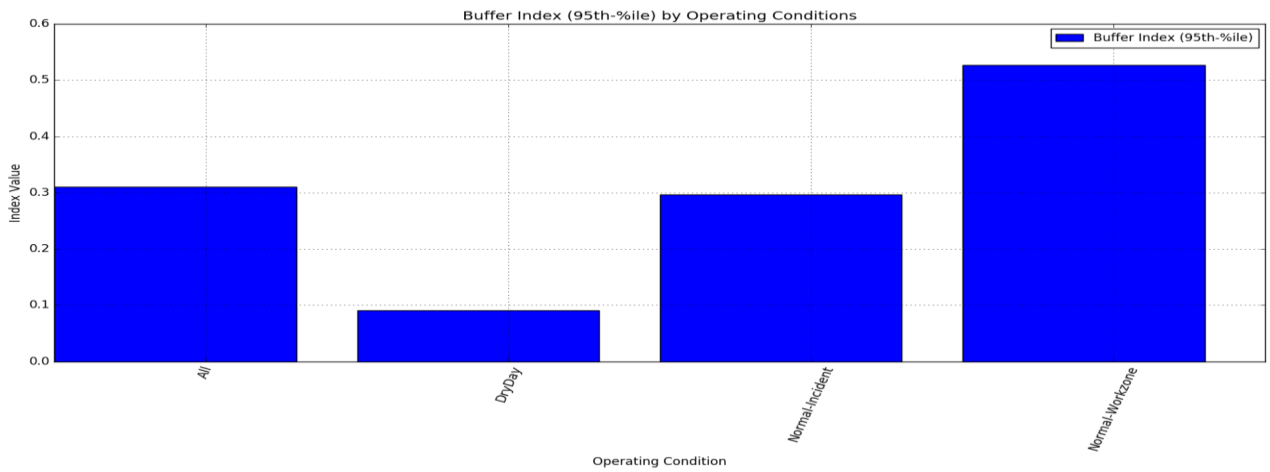


Figure 8.4.2 Buffer Index (95th-ile) by Operating Condition Types for I-35E NB

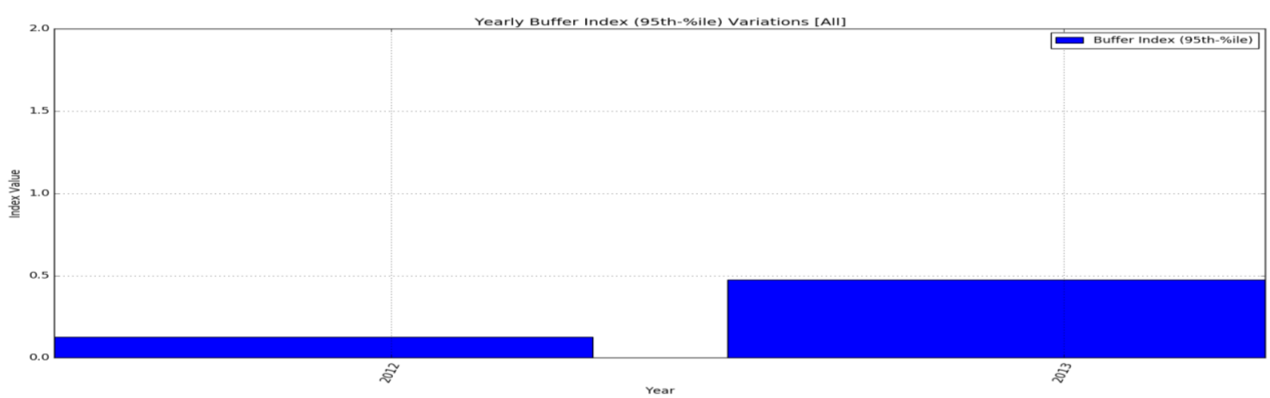


Figure 8.4.3 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of I-35E NB

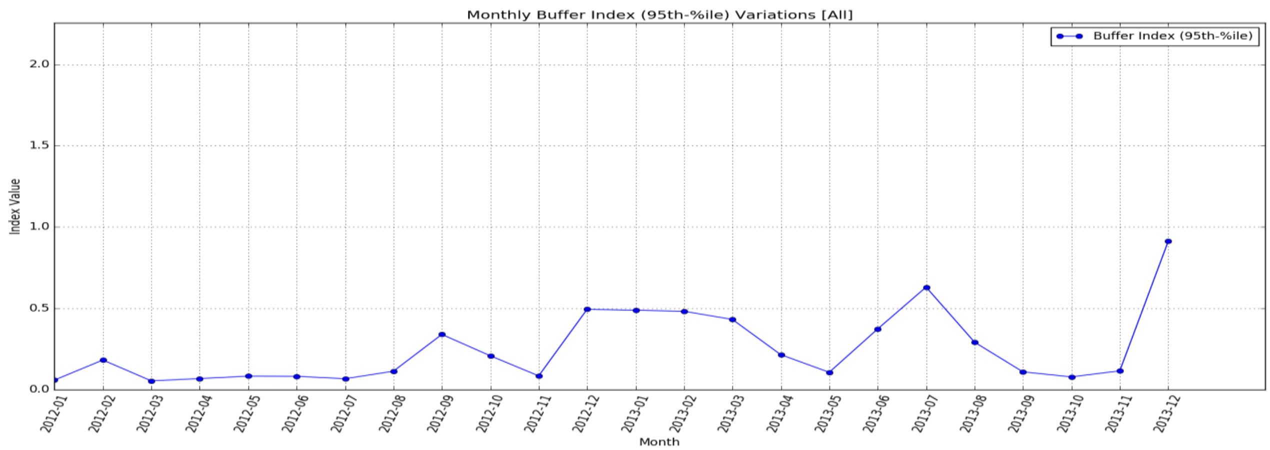


Figure 8.4.4 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of I-35E NB

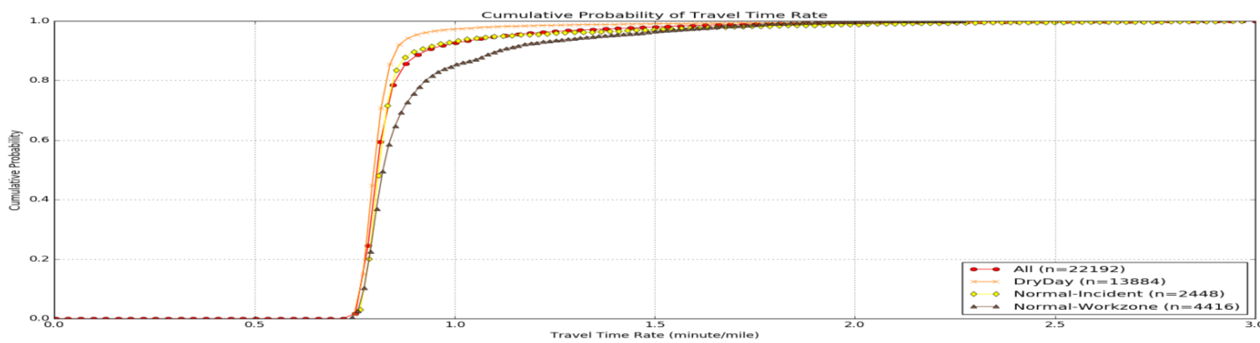


Figure 8.4.5 Cumulative Probability of Travel-Time Rate of I-35E NB

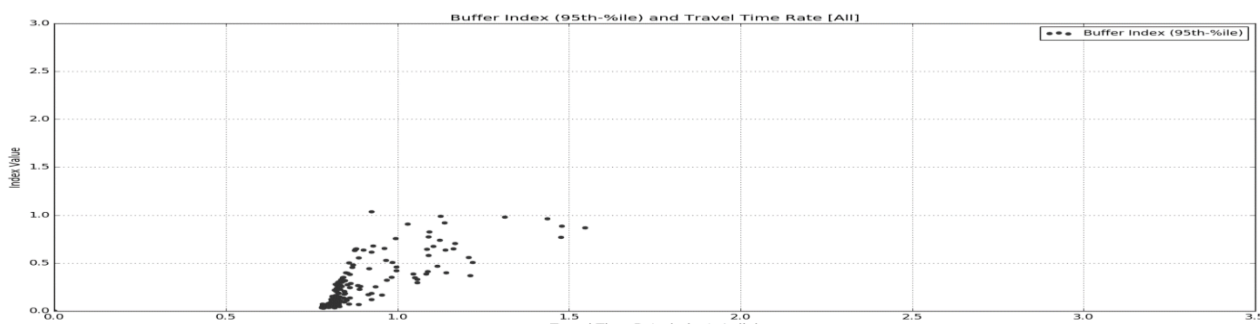


Figure 8.4.6 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for Operating Condition Type "All" of I-35E NB

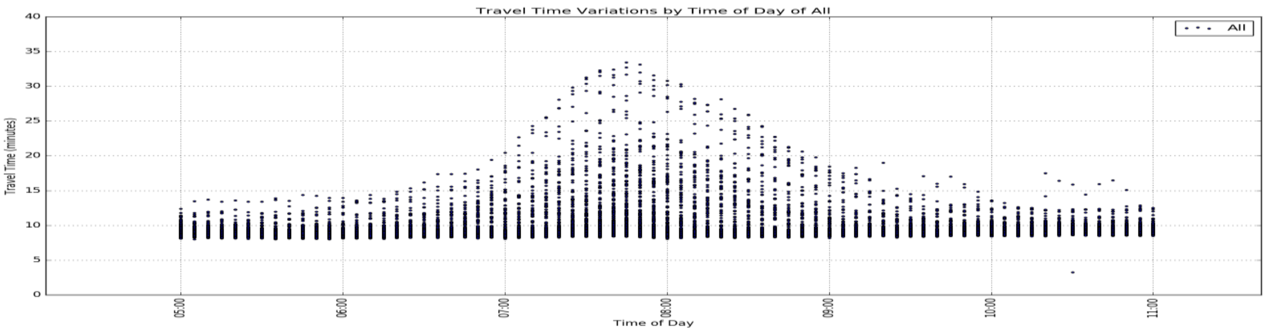


Figure 8.4.7 Time-of-Day Travel-Time Distribution of I-35E NB

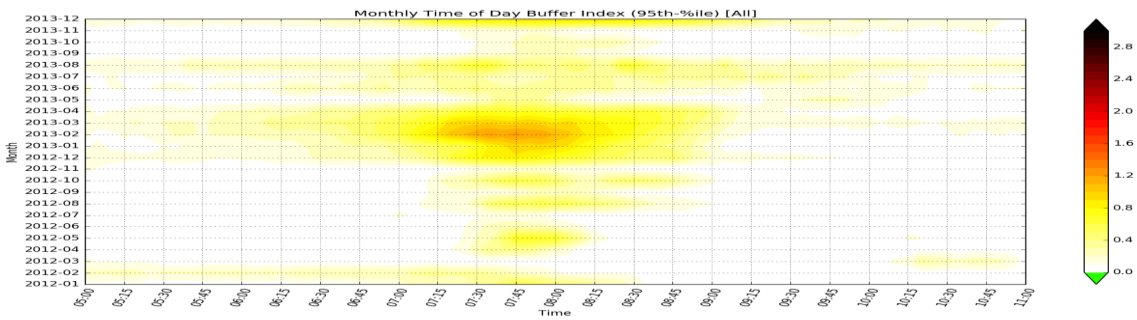


Figure 8.4.8 Monthly TOD Buffer Index (95%-ile) of I-35E NB

Estimation Results of I-35W NB

Table 8.4.6 Reliability Measures by Operating Condition Type of I-35W NB

OC Index	**0**	**1**	**2**
OC Name	All	DryDay	Normal-Incident
Avg TT	9.00	8.71	8.80
Travel Time Rate (minute/mile)	1.06	1.02	1.03
Data Count	22119.00	14714.00	10398.00
Free-Flow TT using Speed Limit	8.96	8.96	8.96
Congested Avg. TT	13.44	12.86	13.45
Congested Data Count	4798.00	2742.00	2051.00
80th %-ile TT	10.64	10.10	10.30
85th %-ile TT	11.58	11.01	11.35
90th %-ile TT	12.63	12.00	12.46
95th %-ile TT	14.25	13.37	14.00
Buffer Index (80th %-ile)	0.18	0.16	0.17
Buffer Index (85th %-ile)	0.29	0.26	0.29
Buffer Index (90th %-ile)	0.40	0.38	0.42
Buffer Index (95th %-ile)	0.58	0.54	0.59

Planning Time Index (80th %-ile)	1.34	1.27	1.29
Planning Time Index (85th %-ile)	1.46	1.38	1.43
Planning Time Index (90th %-ile)	1.59	1.51	1.57
Planning Time Index (95th %-ile)	1.79	1.68	1.76
Travel Time Index	1.69	1.62	1.69
Level of Travel Time Reliability	1.38	1.32	1.36
Semi-Variance	13.80	8.60	11.86
Semi-Variance Data Count	6590.00	4223.00	2903.00
On-Time Arrival	0.85	0.88	0.85
On-Time Arrival Data Count	18799.00	12883.00	8863.00
Misery Index	2.03	1.83	2.01

Table 8.4.7 Yearly Reliability Measures for Operating Condition Type "All" of I-35W NB

Year	**2012**	**2013**
Avg TT	8.84	9.17
Travel Time Rate (minute/mile)	1.04	1.07
Data Count	11023.00	11096.00
Free-Flow TT using Speed Limit	8.96	8.96
Congested Avg. TT	12.82	14.06
Congested Data Count	2388.00	2409.00
80th %-ile TT	10.65	10.63
85th %-ile TT	11.57	11.58
90th %-ile TT	12.48	12.91
95th %-ile TT	13.83	14.80
Buffer Index (80th %-ile)	0.21	0.16
Buffer Index (85th %-ile)	0.31	0.26
Buffer Index (90th %-ile)	0.41	0.41
Buffer Index (95th %-ile)	0.56	0.61
Planning Time Index (80th %-ile)	1.34	1.34
Planning Time Index (85th %-ile)	1.45	1.46
Planning Time Index (90th %-ile)	1.57	1.62
Planning Time Index (95th %-ile)	1.74	1.86
Travel Time Index	1.61	1.77
Level of Travel Time Reliability	1.39	1.37
Semi-Variance	5.49	21.86
Semi-Variance Data Count	3339.00	3241.00
On-Time Arrival	0.85	0.85
On-Time Arrival Data Count	9344.00	9458.00
Misery Index	1.91	2.26

Table 8.4.8 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of I-35W NB

Year	**2012**											
Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Avg TT	8.44	8.40	8.06	8.91	8.86	8.96	8.62	8.93	10.36	9.20	8.25	9.13
Travel Time Rate (minute/mile)	0.99	0.98	0.95	1.04	1.04	1.05	1.01	1.05	1.21	1.08	0.97	1.07
Data Count	949	949	949	876	1022	803	876	1022	876	1022	876	803
Free-Flow TT using Speed Limit	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96
Congested Avg. TT	12.77	12.78	12.29	12.77	11.94	12.21	12.23	12.86	13.87	12.61	11.79	14.64
Congested Data Count	160	144	87	207	249	201	159	236	368	295	117	165
80th %-ile TT	9.56	9.33	8.68	10.95	11.01	11.13	9.97	10.66	13.54	11.57	9.06	10.46
85th %-ile TT	10.66	10.40	9.34	11.83	11.59	11.87	10.93	11.90	14.19	12.03	9.97	12.18
90th %-ile TT	11.75	11.38	10.17	12.86	12.08	12.42	11.90	12.97	14.90	12.67	10.99	14.38
95th %-ile TT	13.13	13.13	11.41	14.03	12.64	13.15	12.85	14.12	16.40	14.02	12.09	16.58
Buffer Index (80th %-ile)	0.13	0.11	0.08	0.23	0.24	0.24	0.16	0.19	0.31	0.26	0.10	0.15
Buffer Index (85th %-ile)	0.26	0.24	0.16	0.33	0.31	0.32	0.27	0.33	0.37	0.31	0.21	0.33
Buffer Index (90th %-ile)	0.39	0.36	0.26	0.44	0.36	0.39	0.38	0.45	0.44	0.38	0.33	0.57
Buffer Index (95th %-ile)	0.55	0.56	0.42	0.57	0.43	0.47	0.49	0.58	0.58	0.52	0.47	0.82
Planning Time Index (80th %-ile)	1.20	1.17	1.09	1.38	1.38	1.40	1.25	1.34	1.70	1.45	1.14	1.31
Planning Time Index (85th %-ile)	1.34	1.31	1.17	1.49	1.46	1.49	1.37	1.50	1.78	1.51	1.25	1.53
Planning Time Index (90th %-ile)	1.48	1.43	1.28	1.62	1.52	1.56	1.50	1.63	1.87	1.59	1.38	1.81
Planning Time Index (95th %-ile)	1.65	1.65	1.43	1.76	1.59	1.65	1.62	1.78	2.06	1.76	1.52	2.08
Travel Time Index	1.61	1.61	1.54	1.60	1.50	1.54	1.54	1.62	1.74	1.59	1.48	1.84
Level of Travel Time Reliability	1.28	1.25	1.17	1.43	1.41	1.41	1.29	1.38	1.54	1.47	1.20	1.37

Semi-Variance	6.02	5.89	4.32	4.12	1.85	2.28	3.04	4.51	8.62	4.00	2.35	11.56
Semi-Variance Data Count	243	248	259	275	366	277	265	307	366	356	241	216
On-Time Arrival	0.87	0.89	0.94	0.83	0.86	0.84	0.89	0.84	0.78	0.82	0.91	0.83
On-Time Arrival Data Count	829	847	895	726	883	678	778	856	683	838	799	664
Misery Index	1.85	1.82	1.59	1.90	1.67	1.72	1.71	1.91	2.36	1.92	1.59	2.34

Table 8.4.9 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of I-35W NB

Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Avg TT	8.71	9.51	8.42	9.54	8.90	9.71	9.04	9.43	8.30	8.22	8.61	12.12
Travel Time Rate (minute/mile)	1.02	1.11	0.99	1.12	1.04	1.14	1.06	1.11	0.97	0.96	1.01	1.42
Data Count	1022	876	876	949	1022	876	949	949	876	1095	803	803
Free-Flow TT using Speed Limit	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96	8.96
Congested Avg. TT	13.45	15.19	13.21	16.93	12.40	13.71	12.56	12.87	11.79	11.77	12.81	18.97
Congested Data Count	179	203	102	182	236	276	237	291	103	121	141	338
80th %-ile TT	9.85	10.87	9.32	10.25	10.82	12.14	11.11	12.37	9.03	8.94	9.96	18.47
85th %-ile TT	10.94	11.72	9.95	11.01	11.52	13.32	11.91	13.10	9.90	9.82	10.75	19.33
90th %-ile TT	12.18	12.89	10.52	12.48	12.31	14.26	12.65	13.62	10.63	10.46	11.62	21.19
95th %-ile TT	14.54	16.38	12.84	14.88	13.57	15.47	13.95	14.19	11.61	11.32	12.64	24.58
Buffer Index (80th %-ile)	0.13	0.14	0.11	0.07	0.22	0.25	0.23	0.31	0.09	0.09	0.16	0.44
Buffer Index (85th %-ile)	0.26	0.23	0.18	0.15	0.30	0.37	0.32	0.39	0.19	0.19	0.25	0.60
Buffer Index (90th %-ile)	0.40	0.35	0.25	0.31	0.38	0.47	0.40	0.44	0.28	0.27	0.35	0.75
Buffer Index (95th %-ile)	0.67	0.72	0.53	0.56	0.53	0.59	0.54	0.51	0.40	0.38	0.47	1.03
Planning Time Index (80th %-ile)	1.24	1.37	1.17	1.29	1.36	1.53	1.40	1.56	1.13	1.12	1.25	2.20

Planning Time Index (85th %-ile)	1.37	1.47	1.25	1.38	1.45	1.67	1.50	1.65	1.24	1.23	1.35	2.43
Planning Time Index (90th %-ile)	1.53	1.62	1.32	1.57	1.55	1.79	1.59	1.71	1.34	1.32	1.46	2.66
Planning Time Index (95th %-ile)	1.83	2.06	1.61	1.87	1.71	1.94	1.75	1.78	1.46	1.42	1.59	3.09
Travel Time Index	1.69	1.91	1.66	2.13	1.56	1.72	1.58	1.62	1.48	1.48	1.61	2.26
Level of Travel Time Reliability	1.30	1.41	1.23	1.34	1.38	1.51	1.41	1.54	1.18	1.17	1.31	1.98
Semi-Variance	8.77	41.49	6.77	100.65	3.41	8.93	3.45	3.01	2.45	2.74	9.23	29.52
Semi-Variance Data Count	282	255	250	224	343	309	318	362	266	309	229	286
On-Time Arrival	0.87	0.84	0.92	0.87	0.87	0.79	0.84	0.79	0.95	0.95	0.88	0.68
On-Time Arrival Data Count	888	739	805	830	891	695	801	745	830	1043	710	544
Misery Index	2.08	2.97	1.93	3.39	1.85	2.19	1.88	1.84	1.62	1.58	1.74	3.77

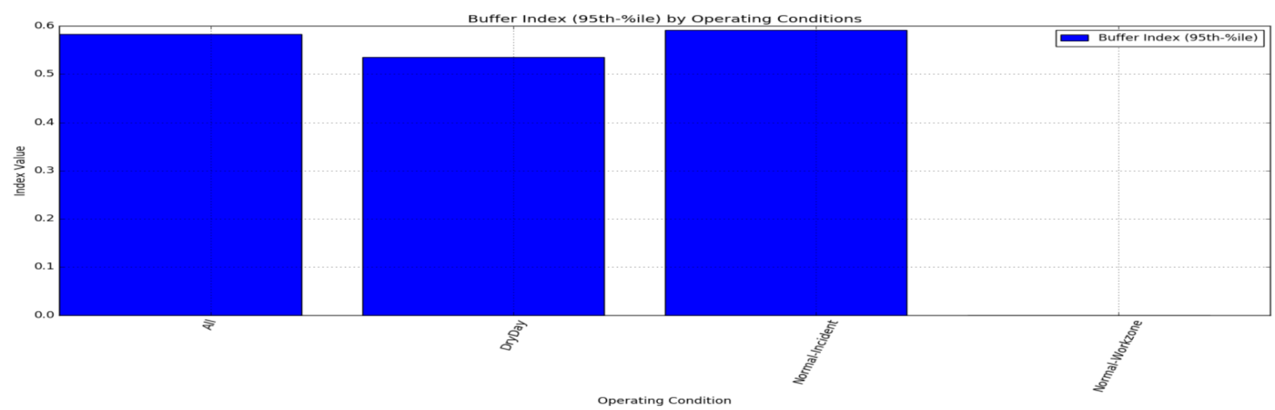


Figure 8.4.9 Buffer Index (95th-ile) by Operating Conditions of I-35W NB

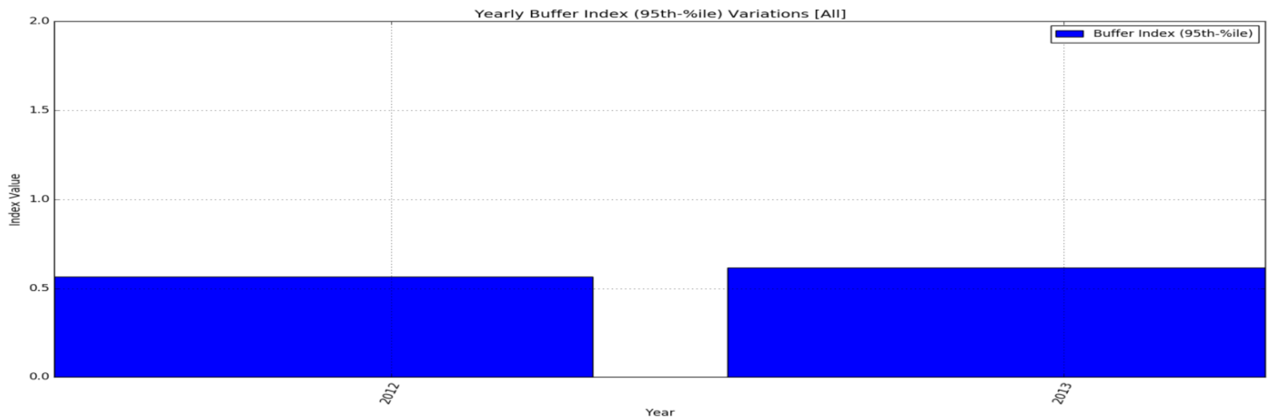


Figure 8.4.10 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of I-35W NB

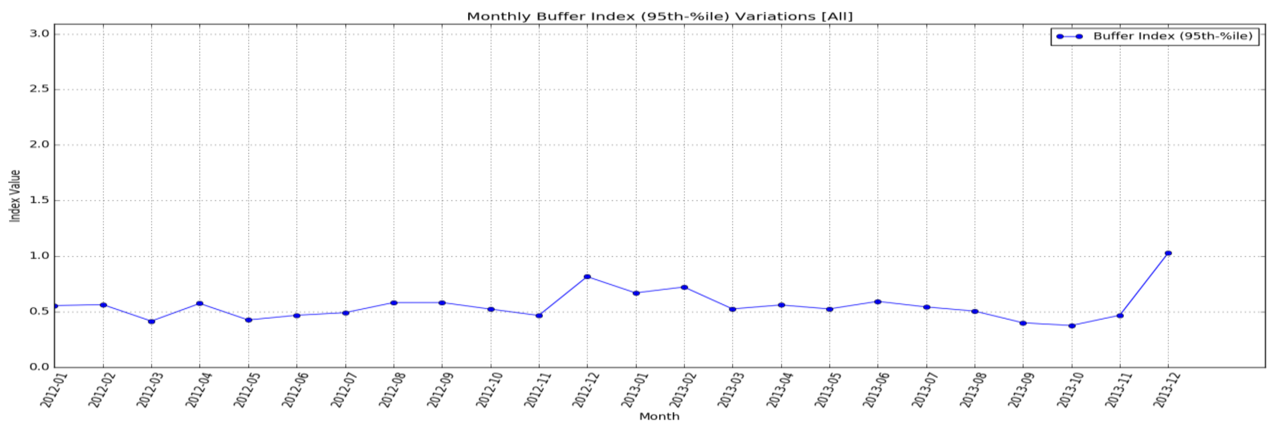


Figure 8.4.11 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of I-35W NB

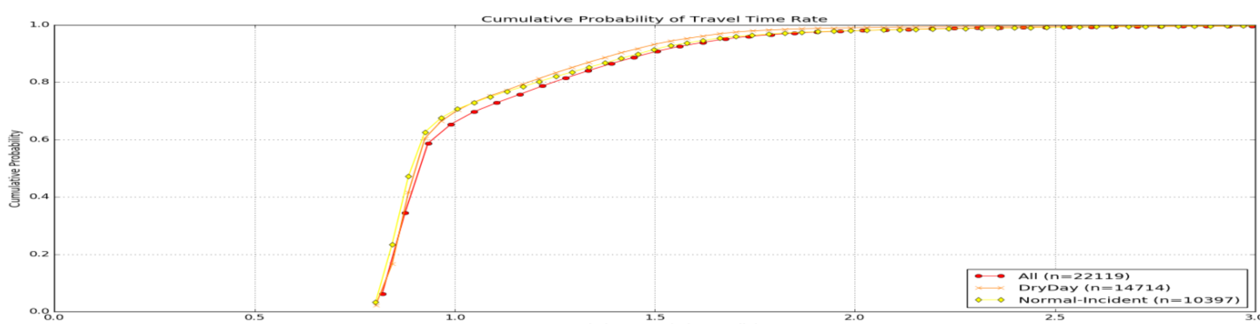


Figure 8.4.12 Cumulative Probability of Travel Time Rate of I-35W NB

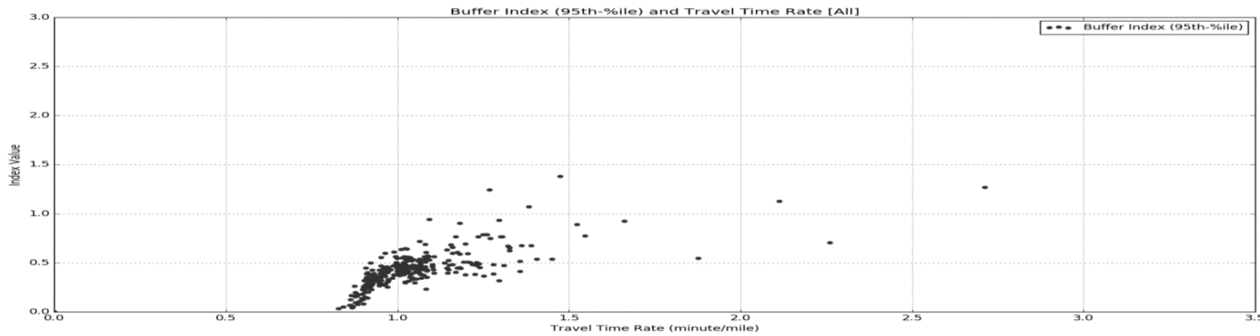


Figure 8.4.13 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for "All operating condition" of I-35W NB

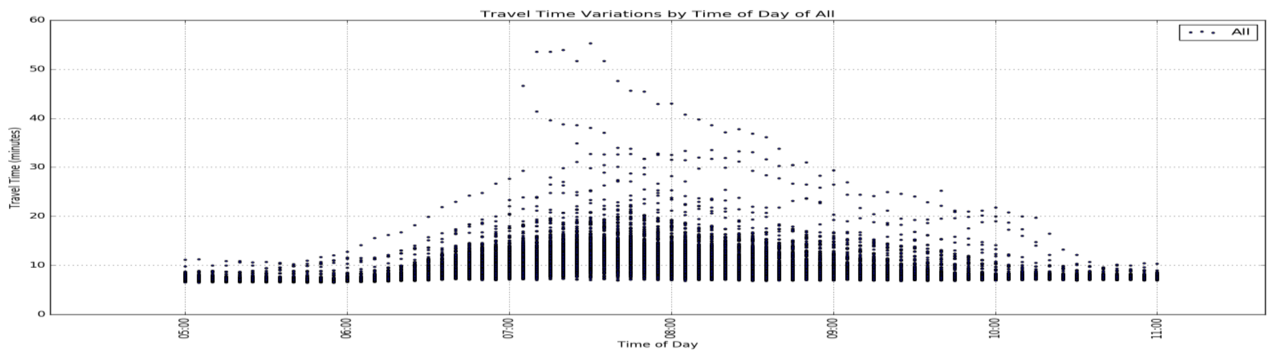


Figure 8.4.14 Time-of-Day Travel Time Distribution of I-35W NB

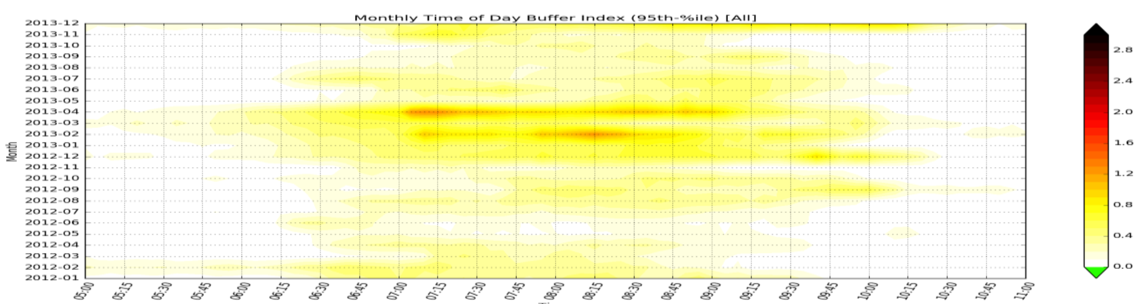


Figure 8.4.15 Monthly TOD Buffer Index (95%-ile) of I-35W NB

Estimation Results for U.S.169 NB

Table 8.4.10 Reliability Measures by Operating Conditions of U.S.169 NB

OC Index	**0**	**1**	**2**
OC Name	All	DryDay	Normal-Incident
Avg TT	8.51	8.18	8.88
Travel Time Rate (minute/mile)	0.96	0.92	1.01
Data Count	22264.00	13755.00	1854.00

Free-Flow TT using Speed Limit	8.51	8.51	8.51
Congested Avg. TT	18.90	16.83	18.60
Congested Data Count	805.00	206.00	112.00
80th %-ile TT	8.36	8.25	8.50
85th %-ile TT	8.55	8.34	8.82
90th %-ile TT	8.19	8.51	8.94
95th %-ile TT	10.02	8.34	11.91
Buffer Index (80th %-ile)	-	0.01	-
Buffer Index (85th %-ile)	0.01	0.02	
Buffer Index (90th %-ile)	0.09	0.05	0.13
Buffer Index (95th %-ile)	0.33	0.16	0.51
Planning Time Index (80th %-ile)	0.87	0.85	0.88
Planning Time Index (85th %-ile)	0.89	0.86	0.92
Planning Time Index (90th %-ile)	0.96	0.88	1.05
Planning Time Index (95th %-ile)	1.18	0.98	1.40
Travel Time Index	2.10	1.98	2.19
Level of Travel Time Reliability	1.05	1.04	1.06
Semi-Variance	32.12	8.86	59.76
Semi-Variance Data Count	3526.00	3526.00	269.00
On-Time Arrival	0.96	0.98	0.93
On-Time Arrival Data Count	21299.00	13508.00	1730.00
Misery Index	1.50	1.13	1.75

Table 8.4.11 Yearly Reliability Measures for Operating Condition Type "All" of U.S.169 NB

Year	**2012**	**2013**
Avg TT	8.23	8.78
Travel Time Rate (minute/mile)	0.92	0.99
Data Count	11169.00	11095.00
Free-Flow TT using Speed Limit	8.51	8.51
Congested Avg. TT	16.05	18.61
Congested Data Count	224.00	581.00
80th %-ile TT	8.29	8.54
85th %-ile TT	8.38	8.08
90th %-ile TT	8.57	9.16
95th %-ile TT	8.42	11.22
Buffer Index (80th %-ile)	0.01	0.00
Buffer Index (85th %-ile)	0.02	0.04

Buffer Index (90th %-ile)	0.05	0.18
Buffer Index (95th %-ile)	0.16	0.44
Planning Time Index (80th %-ile)	0.86	0.89
Planning Time Index (85th %-ile)	0.87	0.95
Planning Time Index (90th %-ile)	0.89	1.08
Planning Time Index (95th %-ile)	0.99	1.32
Travel Time Index	1.89	2.19
Level of Travel Time Reliability	1.04	1.08
Semi-Variance	8.28	45.63
Semi-Variance Data Count	2695.00	1905.00
On-Time Arrival	0.98	0.94
On-Time Arrival Data Count	10916.00	10383.00
Misery Index	1.19	1.73

Table 8.4.12 Monthly Reliability Measures in 2012 of Operating Condition Type "All" of U.S.169 NB

Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Avg TT	8.06	8.91	6.88	6.85	8.02	8.07	8.14	8.09	8.22	8.74	8.17	8.67
Travel Time Rate (minute/mile)	0.90	1.01	0.88	0.88	0.90	0.90	0.91	0.91	0.92	0.99	0.92	0.98
Data Count	949	949	949	876	1095	876	876	1022	876	1022	876	803
Free-Flow TT using Speed Limit	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51
Congested Avg. TT	14.90	20.94	12.99	-1.00	13.07	13.35	11.31	11.41	14.00	14.06	11.94	15.22
Congested Data Count	4	59	2	0	6	3	1	1	7	77	7	57
80th %-ile TT	8.18	8.27	8.07	8.08	8.23	8.30	8.35	8.31	8.40	8.66	8.38	8.53
85th %-ile TT	8.25	8.42	8.11	8.13	8.29	8.37	8.44	8.36	8.52	8.00	8.47	8.87
90th %-ile TT	8.38	8.77	8.20	8.19	8.41	8.49	8.62	8.45	8.75	9.20	8.70	9.03
95th %-ile TT	8.21	12.18	8.33	8.29	8.70	8.83	8.13	8.67	8.18	12.81	8.67	13.56
Buffer Index (80th %-ile)	0.02	0.00	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.00	0.03	0.00
Buffer Index (85th %-ile)	0.03	0.00	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.03	0.04	0.03
Buffer Index (90th %-ile)	0.05	0.11	0.05	0.05	0.06	0.06	0.07	0.05	0.07	0.19	0.07	0.18
Buffer Index (95th %-ile)	0.16	0.54	0.07	0.07	0.10	0.11	0.14	0.08	0.13	0.66	0.21	0.77

Planning Time Index (80th %-ile)	0.84	0.85	0.83	0.83	0.85	0.86	0.86	0.86	0.87	0.90	0.87	0.88
Planning Time Index (85th %-ile)	0.85	0.87	0.84	0.84	0.86	0.87	0.87	0.86	0.88	0.94	0.88	0.92
Planning Time Index (90th %-ile)	0.87	1.03	0.85	0.84	0.87	0.88	0.90	0.88	0.91	1.08	0.91	1.06
Planning Time Index (95th %-ile)	0.96	1.43	0.86	0.86	0.91	0.92	0.95	0.90	0.96	1.51	1.02	1.59
Travel Time Index	1.75	2.46	1.53	-1.00	1.54	1.57	1.33	1.34	1.65	1.65	1.40	1.79
Level of Travel Time Reliability	1.04	1.05	1.03	1.03	1.04	1.04	1.04	1.04	1.04	1.07	1.05	1.08
Semi-Variance	1.36	71.66	0.23	0.04	0.65	0.52	0.47	0.38	1.15	9.41	0.92	14.12
Semi-Variance Data Count	295	108	413	441	467	396	350	458	322	184	299	139
On-Time Arrival	0.99	0.93	1.00	1.00	0.99	0.99	1.00	1.00	0.99	0.92	0.99	0.92
On-Time Arrival Data Count	943	887	947	876	1089	870	872	1018	867	943	868	742
Misery Index	1.04	2.74	0.88	0.87	0.97	0.98	1.05	1.01	1.01	1.71	1.11	1.84

Table 8.4.13 Monthly Reliability Measures in 2013 of Operating Condition Type "All" of U.S.169 NB

Month	1	2	3	4	5	6	7	8	9	10	11	12
Avg TT	8.26	8.26	8.36	8.81	8.29	9.20	6.89	6.82	8.16	8.41	8.34	11.25
Travel Time Rate (minute/mile)	0.93	1.06	0.94	1.00	0.93	1.18	0.88	0.87	0.91	0.95	0.94	1.44
Data Count	1021	876	876	949	1022	876	949	949	876	1095	803	803
Free-Flow TT using Speed Limit	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51	8.51
Congested Avg. TT	12.21	25.97	12.87	20.49	11.91	16.24	-1.00	-1.00	14.07	12.97	14.05	22.28
Congested Data Count	33	52	33	50	7	118	0	0	14	39	40	195
80th %-ile TT	8.36	8.76	8.30	8.43	8.48	10.30	8.11	8.04	8.28	8.38	8.20	13.03
85th %-ile TT	8.57	8.22	8.83	8.89	8.66	10.82	8.15	8.10	8.37	8.57	8.33	15.66
90th %-ile TT	8.10	9.04	9.15	8.82	8.37	11.92	8.25	8.16	8.48	8.71	8.82	21.11

95th %-ile TT	9.67	11.95	10.53	11.34	9.54	15.32	8.41	8.26	8.94	10.59	10.87	28.36
Buffer Index (80th %-ile)	0.01	0.00	0.00	0.00	0.03	0.12	0.03	0.03	0.02	0.00	0.00	0.16
Buffer Index (85th %-ile)	0.04	0.00	0.06	0.01	0.05	0.18	0.04	0.04	0.03	0.02	0.00	0.39
Buffer Index (90th %-ile)	0.12	0.09	0.24	0.13	0.15	0.30	0.05	0.05	0.05	0.18	0.06	0.88
Buffer Index (95th %-ile)	0.33	0.45	0.43	0.45	0.31	0.67	0.08	0.07	0.11	0.43	0.48	1.52
Planning Time Index (80th %-ile)	0.86	0.91	0.86	0.87	0.88	1.21	0.84	0.83	0.86	0.87	0.85	1.53
Planning Time Index (85th %-ile)	0.89	0.97	0.92	0.93	0.90	1.27	0.84	0.83	0.87	0.89	0.86	1.84
Planning Time Index (90th %-ile)	0.95	1.06	1.08	1.04	0.98	1.40	0.85	0.84	0.88	1.02	0.92	2.48
Planning Time Index (95th %-ile)	1.14	1.40	1.24	1.33	1.12	1.80	0.87	0.85	0.93	1.24	1.28	3.33
Travel Time Index	1.44	3.05	1.51	2.41	1.40	1.91	-1.00	-1.00	1.65	1.52	1.65	2.62
Level of Travel Time Reliability	1.06	1.12	1.06	1.07	1.06	1.28	1.04	1.04	1.04	1.05	1.04	1.67
Semi-Variance	2.87	126.67	3.97	46.39	1.30	22.97	0.13	0.05	2.64	4.42	9.85	119.95
Semi-Variance Data Count	245	127	161	148	311	296	435	434	270	207	118	193
On-Time Arrival	0.96	0.94	0.94	0.94	0.99	0.91	1.00	1.00	0.98	0.95	0.95	0.77
On-Time Arrival Data Count	979	821	827	890	1009	794	947	949	858	1036	760	618
Misery Index	1.35	3.42	1.41	2.32	1.19	2.43	0.89	0.87	1.15	1.38	1.63	4.81

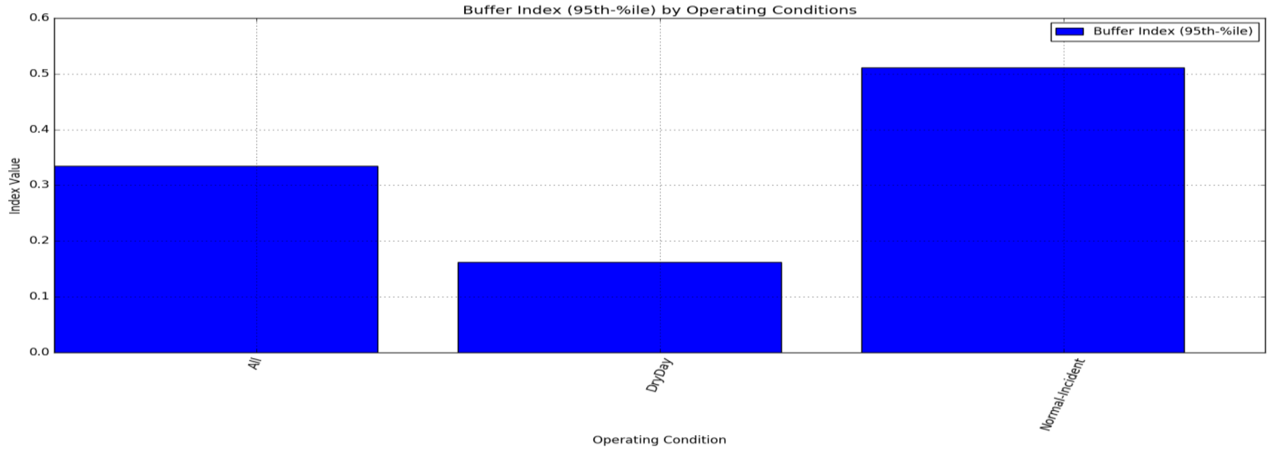


Figure 8.4.16 Buffer Index (95th-ile) by Operating Conditions of U.S.169 NB

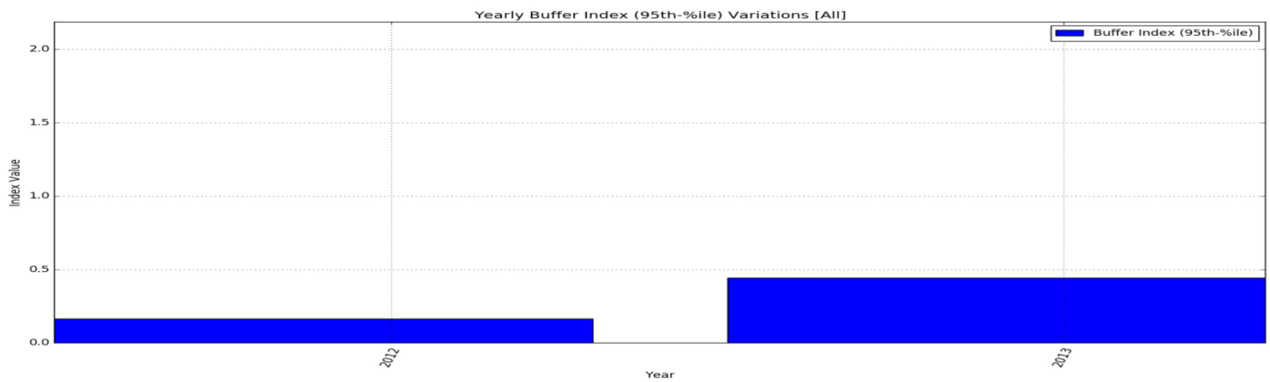


Figure 8.4.17 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of U.S.169 NB

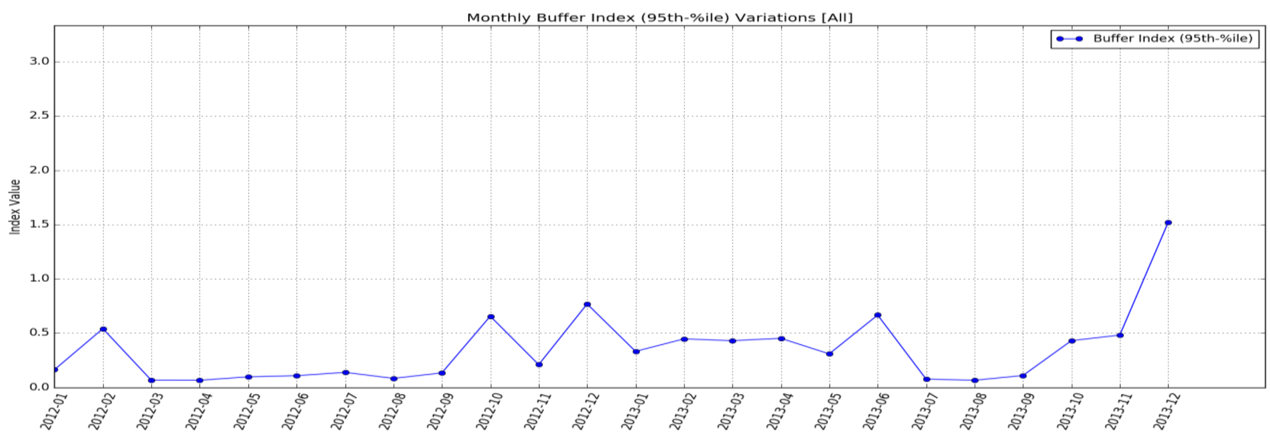


Figure 8.4.18 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of U.S.169 NB

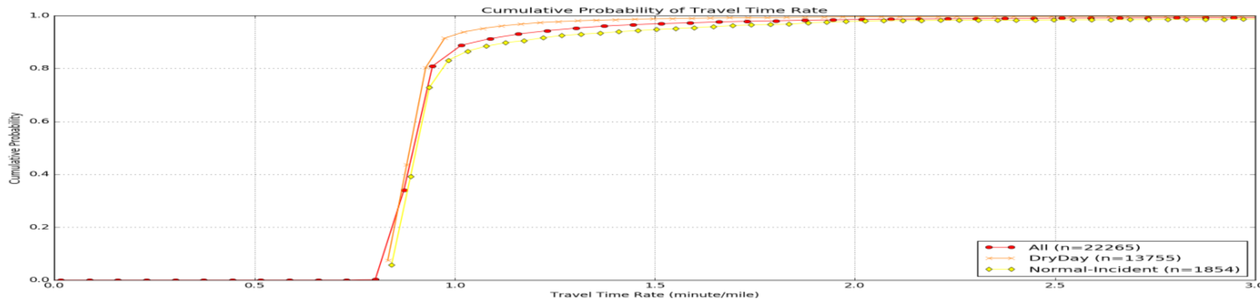


Figure 8.4.19 Cumulative Probability of Travel Time Rate of U.S.169 NB

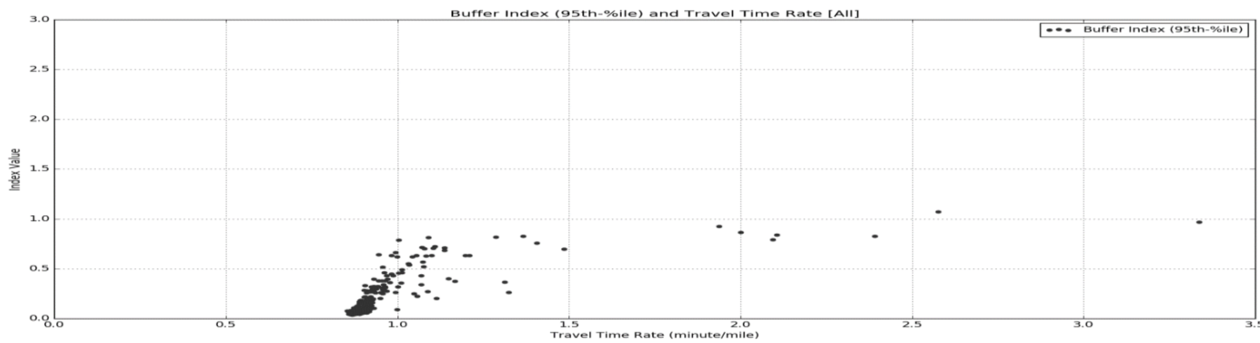


Figure 8.4.20 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for “All operating condition” of U.S.169 NB

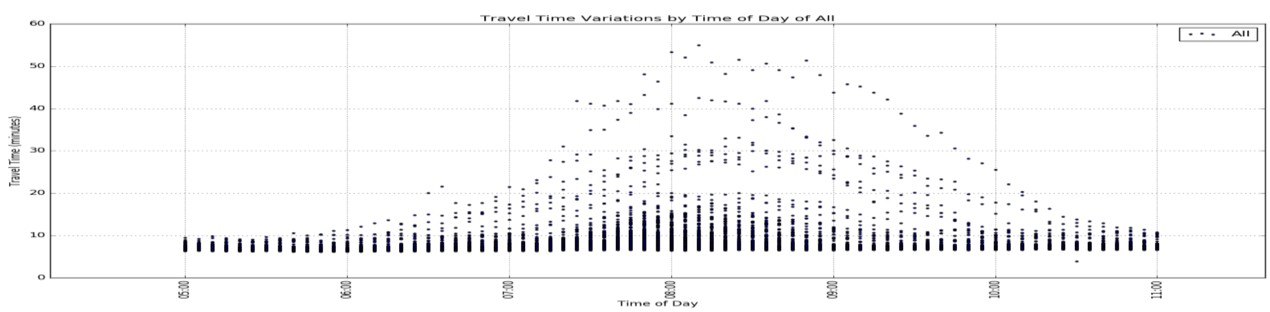


Figure 8.4.21 Time-of-Day Travel Time Distribution of U.S.169 NB

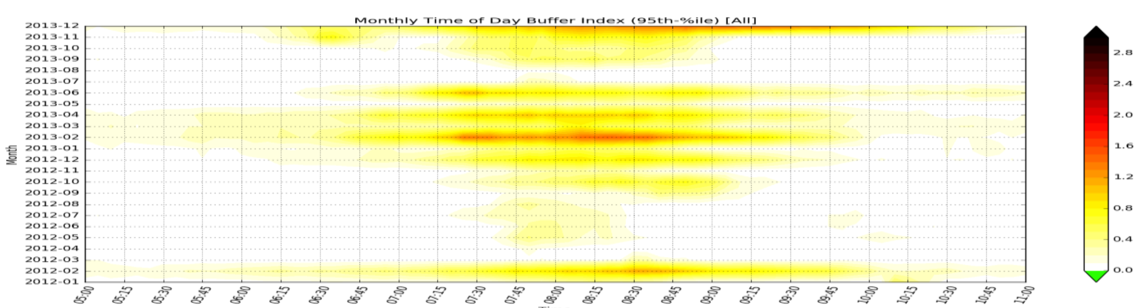


Figure 8.4.22 Monthly TOD Buffer Index (95%-ile) of U.S.169 NB

Estimation Results for T.H.100 NB

Table 8.4.14 Reliability Measures by Operating Condition Types of T.H.100 NB

OC Index	**0**	**1**	**2**
OC Name	All	DryDay	Normal-Incident
Avg TT	8.43	8.26	8.67
Travel Time Rate (minute/mile)	1.05	1.03	1.08
Data Count	22264.00	19144.00	2050.00
Free-Flow TT using Speed Limit	8.75	8.75	8.75
Congested Avg. TT	13.96	13.20	13.83
Congested Data Count	1395.00	939.00	170.00
80th %-ile TT	8.28	8.08	8.00
85th %-ile TT	8.06	8.59	8.61
90th %-ile TT	8.97	8.58	9.65
95th %-ile TT	10.57	10.03	11.89
Buffer Index (80th %-ile)	-	-	0.04
Buffer Index (85th %-ile)	0.09	0.05	0.12
Buffer Index (90th %-ile)	0.21	0.18	0.26
Buffer Index (95th %-ile)	0.42	0.38	0.55
Planning Time Index (80th %-ile)	0.94	0.91	1.03
Planning Time Index (85th %-ile)	1.04	0.98	1.11
Planning Time Index (90th %-ile)	1.16	1.11	1.25
Planning Time Index (95th %-ile)	1.37	1.29	1.54
Travel Time Index	1.80	1.70	1.79
Level of Travel Time Reliability	1.07	1.04	1.17
Semi-Variance	14.45	10.01	10.80
Semi-Variance Data Count	4166.00	3278.00	468.00
On-Time Arrival	0.94	0.95	0.92
On-Time Arrival Data Count	20941.00	18251.00	1893.00
Misery Index	1.66	1.47	1.95

Table 8.4.15 Yearly Reliability Measures for Operating Condition Type “All” of T.H.100 NB

Year	**2012**	**2013**
Avg TT	8.31	8.55
Travel Time Rate (minute/mile)	1.03	1.07
Data Count	11169.00	11095.00
Free-Flow TT using Speed Limit	8.75	8.75
Congested Avg. TT	13.67	14.16

Congested Data Count	568.00	828.00
80th %-ile TT	8.13	8.57
85th %-ile TT	8.67	8.36
90th %-ile TT	8.61	9.36
95th %-ile TT	10.09	11.24
Buffer Index (80th %-ile)	0.00	0.00
Buffer Index (85th %-ile)	0.05	0.11
Buffer Index (90th %-ile)	0.18	0.24
Buffer Index (95th %-ile)	0.38	0.49
Planning Time Index (80th %-ile)	0.92	0.98
Planning Time Index (85th %-ile)	0.99	1.08
Planning Time Index (90th %-ile)	1.11	1.21
Planning Time Index (95th %-ile)	1.30	1.45
Travel Time Index	1.77	1.83
Level of Travel Time Reliability	1.05	1.11
Semi-Variance	13.52	15.07
Semi-Variance Data Count	1939.00	2230.00
On-Time Arrival	0.95	0.93
On-Time Arrival Data Count	10632.00	10314.00
Misery Index	1.50	1.83

Table 8.4.16 Monthly Reliability Measures in 2012 for Operating Condition Type "All" of T.H.100 NB

Year	**2012**											
Month	1	2	3	4	5	6	7	8	9	10	11	12
Avg TT	8.40	8.03	6.96	8.03	8.38	6.88	8.00	6.92	8.35	8.51	8.08	8.23
Travel Time Rate (minute/mile)	1.05	1.14	0.98	0.99	1.04	0.97	0.99	0.98	1.04	1.06	1.00	1.16
Data Count	949	949	949	876	1095	876	876	1022	876	1022	876	803
Free-Flow TT using Speed Limit	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75
Congested Avg. TT	12.30	16.71	11.86	10.33	12.48	10.43	13.12	11.02	10.93	12.54	10.57	19.02
Congested Data Count	67	97	16	5	69	5	18	27	75	83	29	77
80th %-ile TT	8.40	8.20	6.93	8.03	8.53	6.96	8.00	6.91	8.49	8.51	6.95	8.95
85th %-ile TT	8.26	8.84	8.01	8.31	8.36	8.01	8.12	6.98	8.68	8.62	8.78	8.74
90th %-ile TT	9.23	10.08	8.33	8.24	9.18	8.24	8.34	8.11	9.67	9.72	8.51	9.76

95th %-ile TT	10.87	13.11	8.48	8.96	10.47	8.04	8.17	8.26	10.77	11.37	9.75	18.47
Buffer Index (80th %-ile)	0.00	0.02	0.00	0.00	0.02	0.01	0.00	0.00	0.02	0.00	0.00	0.00
Buffer Index (85th %-ile)	0.12	0.10	0.01	0.04	0.13	0.02	0.02	0.01	0.18	0.15	0.10	0.06
Buffer Index (90th %-ile)	0.25	0.26	0.05	0.17	0.24	0.05	0.05	0.03	0.32	0.29	0.20	0.19
Buffer Index (95th %-ile)	0.47	0.63	0.22	0.28	0.42	0.17	0.17	0.19	0.47	0.51	0.38	1.12
Planning Time Index (80th %-ile)	0.95	1.06	0.89	0.91	0.97	0.90	0.90	0.89	0.97	0.97	0.90	1.03
Planning Time Index (85th %-ile)	1.07	1.14	0.91	0.94	1.08	0.91	0.92	0.90	1.12	1.11	1.00	1.13
Planning Time Index (90th %-ile)	1.19	1.30	0.95	1.06	1.19	0.94	0.95	0.92	1.25	1.25	1.10	1.26
Planning Time Index (95th %-ile)	1.40	1.69	1.09	1.16	1.35	1.04	1.05	1.07	1.39	1.47	1.26	2.25
Travel Time Index	1.59	2.16	1.53	1.33	1.61	1.35	1.69	1.42	1.41	1.62	1.36	2.46
Level of Travel Time Reliability	1.09	1.21	1.03	1.04	1.10	1.03	1.03	1.02	1.10	1.10	1.03	1.16
Semi-Variance	5.52	40.93	2.21	0.89	4.97	0.63	3.85	2.07	1.58	5.45	1.23	51.03
Semi-Variance Data Count	190	197	168	178	231	254	180	189	182	205	152	143
On-Time Arrival	0.93	0.90	0.98	1.00	0.94	1.00	0.98	0.98	0.92	0.92	0.97	0.91
On-Time Arrival Data Count	887	858	933	872	1030	872	859	997	807	940	848	730
Misery Index	1.54	2.97	1.18	1.22	1.60	1.11	1.20	1.30	1.43	1.68	1.33	3.08

Table 8.4.17 Monthly Reliability Measures in 2013 for Operating Condition Type "All" of T.H.100 NB

Year	**2013**											
Month	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Avg TT	8.38	8.26	8.39	8.53	8.25	8.57	6.91	6.87	8.45	8.35	8.21	9.81
Travel Time Rate (minute/mile)	1.04	1.17	1.05	1.07	1.03	1.07	0.98	0.97	1.05	1.04	1.02	1.39
Data Count	1021	876	876	949	1022	876	949	949	876	1095	803	803
Free-Flow TT using Speed Limit	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75	8.75

Congested Avg. TT	12.78	15.15	13.85	18.01	11.27	12.25	11.09	10.94	12.67	12.58	11.68	18.07
Congested Data Count	59	121	46	45	77	79	5	16	65	67	50	197
80th %-ile TT	8.35	8.62	8.23	8.35	8.12	8.14	6.96	6.91	8.82	8.31	6.98	11.51
85th %-ile TT	8.85	9.78	8.06	8.93	8.99	9.00	8.03	6.96	8.71	8.26	8.88	13.42
90th %-ile TT	8.81	11.19	9.06	8.83	8.89	9.91	8.18	8.03	9.59	9.06	9.02	16.95
95th %-ile TT	10.43	16.25	10.17	10.02	10.71	11.91	8.49	8.91	10.86	10.44	10.61	23.56
Buffer Index (80th %-ile)	0.00	0.04	0.00	0.00	0.00	0.08	0.01	0.01	0.05	0.00	0.00	0.17
Buffer Index (85th %-ile)	0.06	0.18	0.09	0.05	0.10	0.19	0.02	0.01	0.17	0.12	0.09	0.37
Buffer Index (90th %-ile)	0.19	0.35	0.23	0.17	0.23	0.31	0.04	0.02	0.29	0.23	0.25	0.73
Buffer Index (95th %-ile)	0.41	0.97	0.38	0.33	0.48	0.57	0.23	0.15	0.46	0.42	0.47	1.40
Planning Time Index (80th %-ile)	0.95	1.11	0.93	0.95	0.92	1.05	0.90	0.89	1.01	0.94	0.90	1.49
Planning Time Index (85th %-ile)	1.01	1.26	1.04	1.02	1.03	1.16	0.91	0.90	1.12	1.07	1.02	1.73
Planning Time Index (90th %-ile)	1.14	1.44	1.17	1.14	1.15	1.28	0.93	0.91	1.24	1.17	1.16	2.19
Planning Time Index (95th %-ile)	1.35	2.10	1.31	1.29	1.38	1.54	1.10	1.02	1.40	1.35	1.37	3.04
Travel Time Index	1.65	1.96	1.79	2.32	1.45	1.58	1.43	1.41	1.64	1.62	1.51	2.20
Level of Travel Time Reliability	1.07	1.25	1.06	1.09	1.05	1.18	1.03	1.03	1.15	1.08	1.04	1.52
Semi-Variance	8.26	20.38	9.07	26.70	2.64	4.10	0.86	1.28	5.27	6.07	3.12	34.66
Semi-Variance Data Count	202	212	167	166	189	220	263	254	188	215	146	208
On-Time Arrival	0.95	0.87	0.95	0.95	0.93	0.92	0.99	0.98	0.93	0.94	0.94	0.79
On-Time Arrival Data Count	967	764	832	904	950	807	944	934	815	1031	753	637
Misery Index	1.54	2.51	1.68	2.17	1.49	1.71	1.16	1.22	1.81	1.59	1.50	3.25

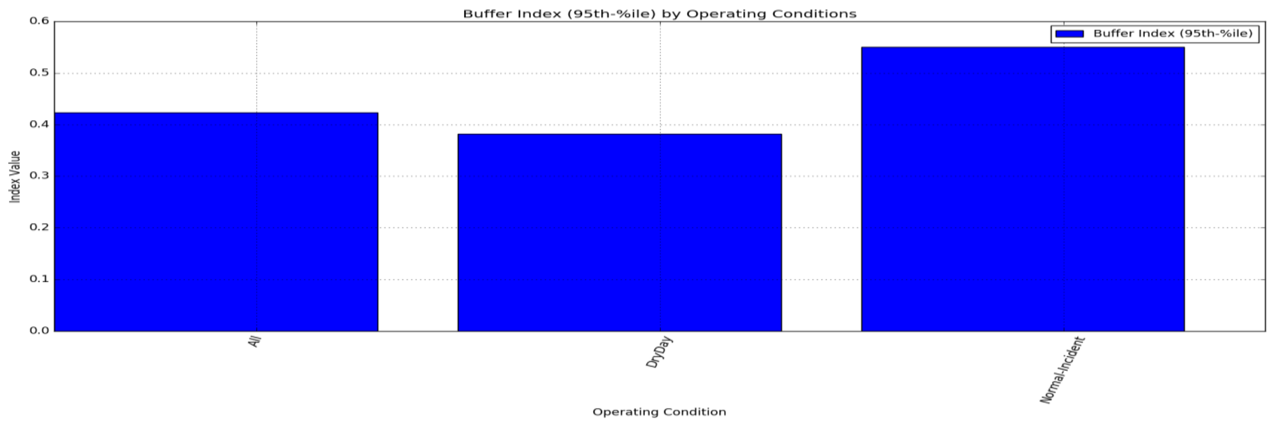


Figure 8.4.23 Buffer Index (95th-ile) by Operating Condition of T.H.100 NB

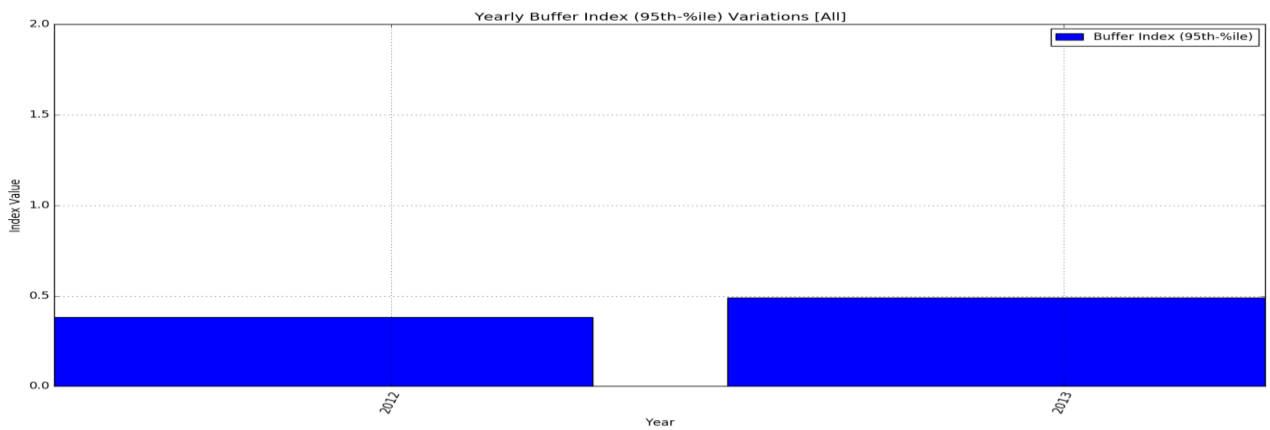


Figure 8.4.24 Yearly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of T.H.100 NB

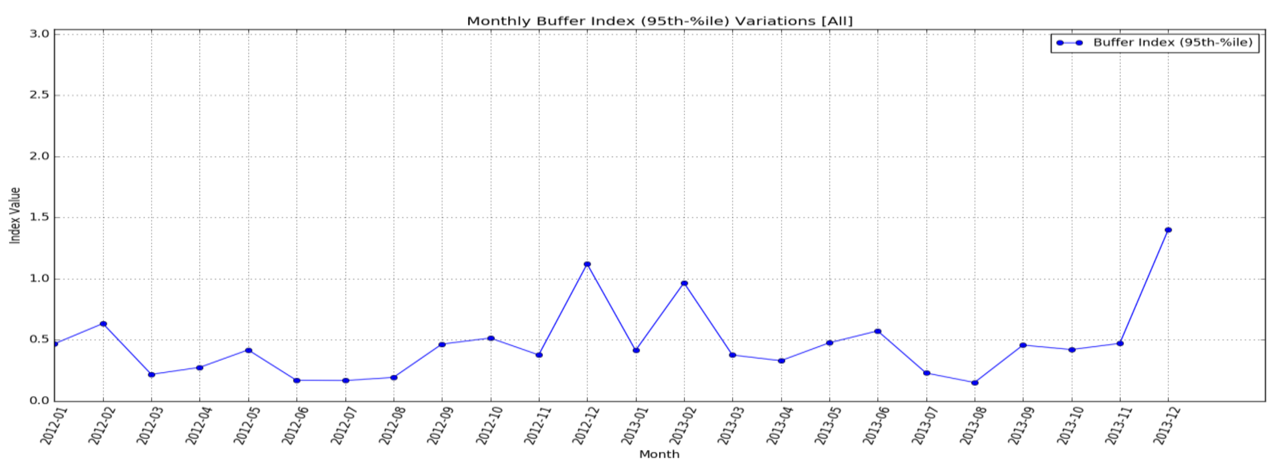


Figure 8.4.25 Monthly Buffer Index (95th-ile) Variations for Operating Condition Type "All" of T.H.100 NB

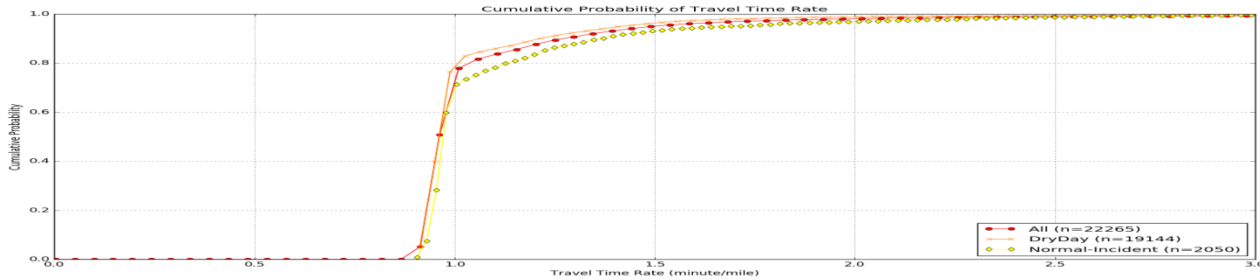


Figure 8.4.26 Cumulative Probability of Travel Time Rate of T.H.100 NB

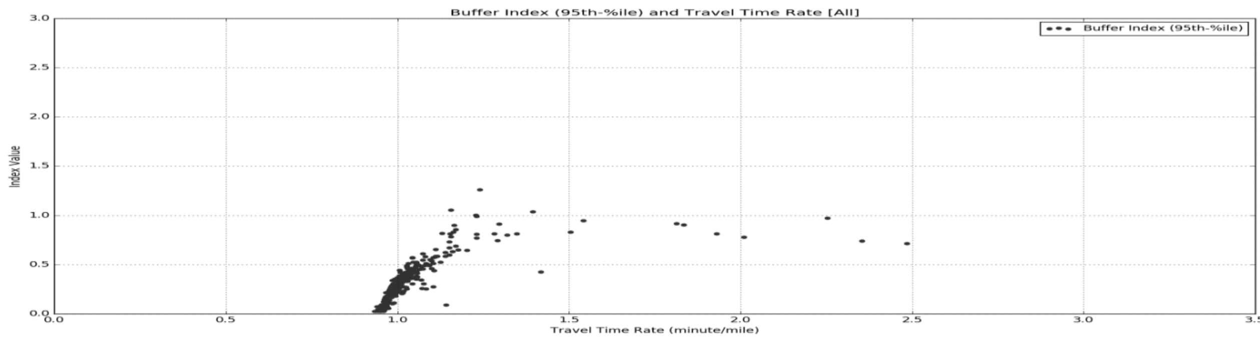


Figure 8.4.27 Daily Buffer Index (95th-ile) vs. Travel-Time Rate for "All operating condition" of T.H.100 NB

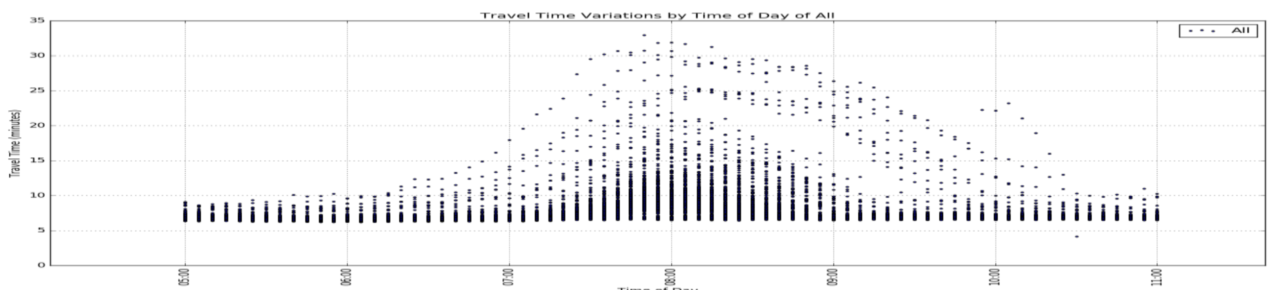


Figure 8.4.28 Time-of-Day Travel Time Distribution of T.H.100 NB

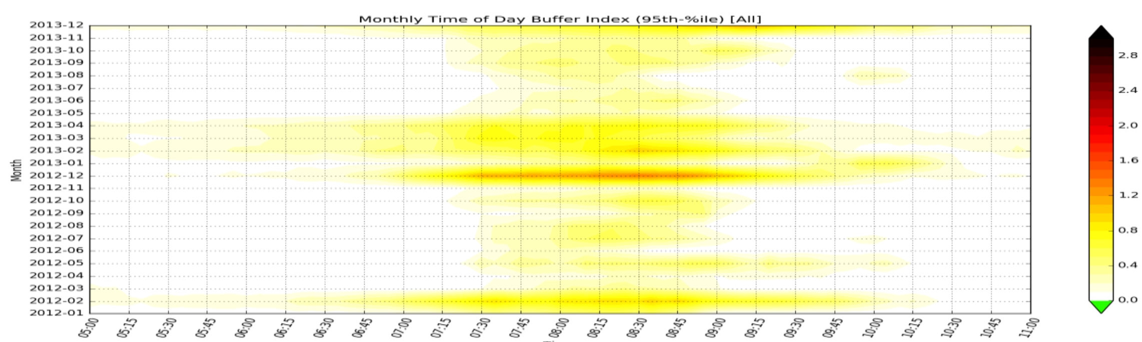


Figure 8.4.29 Monthly TOD Buffer Index (95%-ile) of T.H.100 NB

CHAPTER 9: CONCLUSIONS

Travel-time reliability has been emerging as one of the major measures in quantifying the operational effectiveness of transportation networks. While the importance of travel time reliability in measuring the performance of transportation systems has been well recognized by transportation professionals, the current state of the practice has not reached the point where various types of reliability measures under different operating conditions can be automatically generated using data from multiple sources.

This research has developed a computerized Travel-Time Reliability Measurement System (TTRMS), which can automate the time-consuming process of gathering and managing a large amount of data from multiple sources and calculating a set of reliability indices for the predefined corridors in the metro freeway network. The TTRMS developed in this study employed a top-down approach, where the detailed system architecture was first designed. Further, a comprehensive data-management system was also developed before the development of individual modules. In particular, a set of data-import functions developed and incorporated into the data-management system can automatically download both traffic and non-traffic data from external sources, such as MnDOT's traffic data and NOAA's weather data archives. A travel-time calculation function to determine the travel times at work zones with various lane-configurations was then developed by enhancing the existing travel-time function in TICAS (Traffic Information and Condition Analysis System), developed in the previous research. Next, the reliability estimation module, the main computational engine of TTRMS, was developed to calculate a set of predetermined reliability indices following user-specified operating conditions for predefined corridors and time periods. A reliability-based, time-of-day travel-time estimation module was also developed in a webpage format, whose connectivity to the existing MnDOT's driver information system has shown promise. The development of the user-interfaces for both system administrator and general users was followed. Finally, all the individual modules were integrated, and the resulting TTRMS was tested by applying it for estimating the reliability measures for the selected corridors in the metro freeway network.

The test results using real data indicate that the TTRMS developed in this study can substantially reduce the time and effort in estimating the various types of reliability measures under different operating conditions for the predefined corridors. Further, the map-based graphical user-interfaces of the TTRMS provide both the system administrator and the general users with a flexible environment in defining corridors and specifying the operating conditions for reliability estimation. The modular approach adopted in developing TTRMS allows the addition of new reliability measures and data sources without any major modification of its structure.

The enhancement needs of TTRMS include the automation of the input process for the work-zone lane-configuration data, such as lane-closure and shifting locations and time periods. In the current version of TTRMS, the lane-configuration data for each work-zone are entered manually by users. The availability of the electronic version of work zone data can substantially improve the efficiency and accuracy of the data input process for the work-zone routes.

The future research possibilities with TTRMS can include the identification and prioritization of bottlenecks in the metro freeway network. The extension of reliability to new measures, which can quantify the vulnerability and resilience levels of the existing corridors in dealing with large-scale incidents and natural events, is also recommended. Such measures can be directly applicable for effectively allocating the operational resources to the priority routes and also for developing short- and long-term plans for freeway-network improvements.

REFERENCES

1. Texas Transportation Institute and Cambridge Systems, Inc. (2005), *Travel Time Reliability: Making It There on Time, All the time*, FHWA-HOP-06-070, Retrieved from http://ops.fhwa.dot.gov/publications/tt_reliability, Federal Highway Administration, Washington, D.C.

2. Transportation Research Board (2015), Retrieved from http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Pages/Reliability_Projects_302.aspx.

3. Kuhn, B., Higgins, L., Nelson, A., Finley, M., Ullman, G., Chrysler, S., Wunderlich, C., Shah, V., Dudek, C. (2014), *A Lexicon for Conveying Travel Time Reliability Information,* SHRP2 Report, S2-L14-RW-2, Transportation Research Board, Washington, D.C.

4. List, G., Williams, B. and Rouphail, N. (2014), *Guide to Establishing Monitoring Programs for Travel Time Reliability*, SHRT2 Report, S2-LO2-RR-1, Transportation Research Board, Washington, D.C.

5. Kwon, E. and Park, C. (2012), *Development of Freeway Operational Strategies with IRIS-In-Loop Simulation*, Final Report, 2012-04, Minnesota Department of Transportation, St. Paul, Minnesota.