TripLogic: A Demand-Response Dispatching System

A Manuscript

Submitted to

the Department of Computer Science

and the Faculty of the

University of Wisconsin-La Crosse

La Crosse, Wisconsin

by

Phillip James Molstad

in Partial Fulfillment of the

Requirements for the Degree of

Master of Software Engineering

December 2007

TripLogic: A Demand-Response Dispatching System

By Phillip James Molstad

We recommend acceptance of this manuscript in partial fulfillment of this candidate's requirements for the degree of Master of Software Engineering in Computer Science. The candidate has completed the oral examination requirement of the capstone project for the degree.

Dr. Kasi Periyasamy Examination Committee Chairperson	Date
Dr. Tom Gendreau Examination Committee Member	Date
Dr. Mark Headington Examination Committee Member	Date

ABSTRACT

MOLSTAD, PHILLIP, J., "TripLogic: A Demand-Response Dispatching System", Master of Software Engineering, December 2007, Advisors: Dr. Kasi Periyasamy and Dr. Tom Gendreau.

It is common in practice that many companies build complex and highly delicate business processes around their legacy (existing software) systems making it extremely difficult and costly to switch software systems or to change their business processes. Software re-engineering is an approach that takes legacy software that has become expensive to maintain or whose implementation is obsolete, and reconstructs it with current software technologies. The software re-engineering approach is important for recovering and reusing existing software assets, putting high software maintenance costs under control, and establishing a base for future software development. Top Hat, Inc. started developing its demand-response transportation software in 1994. Today, it is a vital part of their business. Unfortunately, it has become increasingly expensive to enhance its features and compete against their competitors. The design structure, code organization, and development platform of the legacy system make it difficult to debug, modify, and distribute to the end-user community. This report discusses the techniques and methodologies used to re-engineer Top Hat's demand-response transportation software. It also describes the benefits, challenges, and issues encountered while introducing good software development techniques, maintaining required functionality, and applying new technologies to the Top Hat's demand-response transportation software.

ACKNOWLEDGEMENTS

Many friends and colleagues have provided invaluable help, support and encouragement throughout the tenure of this project. The faculty and staff at the University of Wisconsin-La Crosse provided a thought-provoking environment and a source for many of the ideas and tactics discussed here.

Particular mention must be made of my project advisors Dr. Kasi Periyasamy and Dr. Tom Gendreau for their excellent advice and guidance, pleasant classroom environments, and commitment toward making the TripLogic project a success.

Special thanks are due to the project sponsor, Beverly Scott, President of Top Hat Inc., for initiating this project and providing the supporting materials and industry expertise needed for developing the TripLogic software. My deepest gratitude goes to Trane® for providing valuable resources, applicable software development experience, and tuition reimbursement. I would also like to thank Jill Bernstein, Manager of Global Logistics Information Systems, for her encouragement and support of this project.

To my wife Jenny and children Rachael and Abigail who endured several months of fatherless evenings while I worked on this project. Thank you Jenny, Rachael, and Abigail, for your patience, support and understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

API

An application program interface is the specific method prescribed by a computer operating system or by an application program by which a programmer writing an application program can make requests of the operating system or another application.

ActiveX

A Microsoft technology used for developing reusable object-oriented software components and sharing information among different applications.

Cross-Functional Flowchart

A cross-functional flowchart is a business process-mapping tool used to articulate the steps and stakeholders of a given process.

DBMS

A database management system (DBMS) is computer software designed for the purpose of managing databases. Typical examples of DBMSs include Oracle, DB2, Microsoft Access, Microsoft SQL Server, PostgreSQL, MySQL and FileMaker.

Demand-Response

The act of logging a request for service while the patron is on the phone and dispatching a vehicle within minutes of the request for service.

GPS

The Global Positioning System (GPS) is the only fully functional Global Navigation Satellite System (GNSS). Utilizing a constellation of at least 24 medium Earth orbit satellites that transmit precise microwave signals, the system enables a GPS receiver to determine its location, speed, direction, and time.

IEEE

The Institute of Electrical and Electronics Engineers is an international non-profit, professional organization for the advancement of technology related to electricity. It is a leading developer of industrial standards (having developed over 900 active industry standards) in a broad range of disciplines, including electric power and energy, biomedical technology and healthcare, information technology, information assurance, telecommunications, consumer electronics, transportation, aerospace, and nanotechnology.

Legacy System

A legacy system is a hardware and software system that uses technologies that are 'old' in comparison with today's technology. It is an older system that is a candidate for phase-out, upgrade, or replacement, generally because it is expensive to maintain, does not comply with data standards or other standards, but remains in operation because of some mission-critical application.

RDBMS

A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd (1970).

SQL

The Structured Query Language is a computer language designed for the retrieval and management of data in relational database management systems, database schema creation and modification, and database object access control management.

Top Hat, Inc.

A company that provides taxi and para-transit services in South Dakota, Missouri and Wisconsin counties and is the project sponsor.

1. Background Information

Top Hat Inc. has developed a paratransit servicing and demand-response dispatching software system that assists its coordinators with trip reservation, scheduling, billing, payroll and vehicle maintenance activities. The software is used to support all of Top Hat's contracts and businesses which include La Crosse County Human Services Children's and Rural contracts, Trempealeau County Unified Board, Green County Human Services contract, Chippewa Falls Shared Ride Taxi, River Falls Shared Ride Taxi, Joplin Missouri Public Transit, Access Medical Transit, Sioux Falls Transit, Access Mobility Products, CTS Taxi, and Hoods Up Service Center [2]. Each of these contracts and businesses is different in many ways but the software treats them as the same. The software is a legacy system, written in 1994, and is still being used to facilitate and manage the data for each of the contracts and businesses.

Top Hat Inc. wanted to re-engineer the whole system it has built over the course of thirteen years but such a re-engineering process would cause too much time and money. Instead, Top Hat Inc. decided to concentrate on its regular taxi service and re-engineer the demand-response dispatching system and to enhance its functionality. It is important to note that a re-engineering process is generally not done to enhance the functionality of an existing system, but to be used in preparation for enhancement. The re-engineered target system can then be built to easily facilitate enhancements [8].

Demand-response dispatching refers to logging and dispatching a taxi within minutes of the request for service. The model for Top Hat's implementation of a demand-response dispatching system was its CTS Taxi business that was opened in 1982 [2]. By taking a detailed look into the daily activities of the CTS Taxi business, it not only becomes evident that many of Top Hat’s business processes were built around their legacy system, but that it is vital to maintain its existing functionality.

At the start of each dispatcher’s shift, the dispatcher logs herself and each driver’s start and end times. The driver’s assigned vehicle, mileage, gas and oil usage are also recorded. To produce payroll for the dispatchers and drivers, the payroll clerk uses the start and end times recorded in the log. The preventative maintenance component of the software uses the vehicle information from the log to notify the service writer when vehicle maintenance is required.

Each driver that is beginning a shift will be given a schedule or driver’s log that either contains a list of trips already scheduled through the software based on client reservations, or is used to document all trips completed by the driver during his or her shift. The information from the driver’s log is later used to correct discrepancies between the information entered into the system by the dispatcher and the actual trip logged in by the driver.

When a dispatcher receives a call from a client, she begins a new call log. For purposes of gathering metrics, the call log is automatically time-stamped when the dispatcher enters the pickup and drop-off locations for the requested trip. While the client is on the phone, the dispatcher enters additional information such as the type of fare (cash or charge account), required equipment, and routing information. Next, a driver is assigned to the trip and the system time-stamps this event. Finally, the dispatcher will mark the trip with a status of ‘done’, ‘no show’, or ‘cancelled’ indicating that the trip has been completed. All completed trips are moved to the trip verification module.

In the verification stage, a verifier manually compares the dispatch records against the driver’s logs for all trips. If discrepancies are found, they are reviewed and corrected and then the trip is marked verified. Once the trips have been verified, they can be sent to either history (archived) or to billing. After the trips are moved to billing, invoices can be generated. Invoices can be printed anytime during the month and are separated by the system according to client's source of payment (where the invoice will be sent). As payments are received, accountants can use the system to apply the payment to any number of the client's unpaid trips. All client account balances are aged by the system providing accountants with a clear picture of not only payments made to the client's account but also of any overdue balances.

A variety of metrics can be printed from the software system. Some of these include vehicle mileage, average fare, fuel usage, cancellations, no-shows, time between taking a call and dispatching a driver, and riders per hour. The information extracted from metrics can be sorted using any of the attributes such as client code, client name, and location.

As described above, the demand-response dispatching system is an integral part of Top Hat's business processes. Since it is vital to maintain the same level of functionality as in the old system and it is critical to the business to expand its feature set, software re-engineering techniques were used to capture and reuse the legacy system's assets, reduce the high maintenance costs, and provide a new base for future development.

2. Getting Started

Initially the sponsor provided a list of informal requirements and source code of the existing software system. To successfully re-engineer this legacy software system, the developer had to first determine what database platform and development tools would be best suited for the project. After learning what features were implemented in the current software system and reviewing the informal requirements, it was determined that the software be re-engineered to take advantage of features provided by Microsoft SQL Server and the C# (C-Sharp) programming language. An equally important component to a successful software project is the choice of a software process model, better known as the software life-cycle.

Combining the information learned in the C-S 446/546 - Object-Oriented Software Development course and personal research done on the software re-engineering process, it became evident that the stages outlined in the software development process tightly mapped to the various levels of abstraction used in the re-engineering process. Further studies led the developer to one of the original models of the software development process called the waterfall model (Royce, 1970). Today, it remains widely used and was the developer’s choice for this project due to its similarity to the refinement principle in the re-engineering process. Figure 2.1 describes a general model for software re-engineering, and also shows the similarities between the forward engineering (refinement) phase of re-engineering and the core stages in the waterfall model: requirements analysis and definition, system and software design, implementation and unit testing, and system testing.

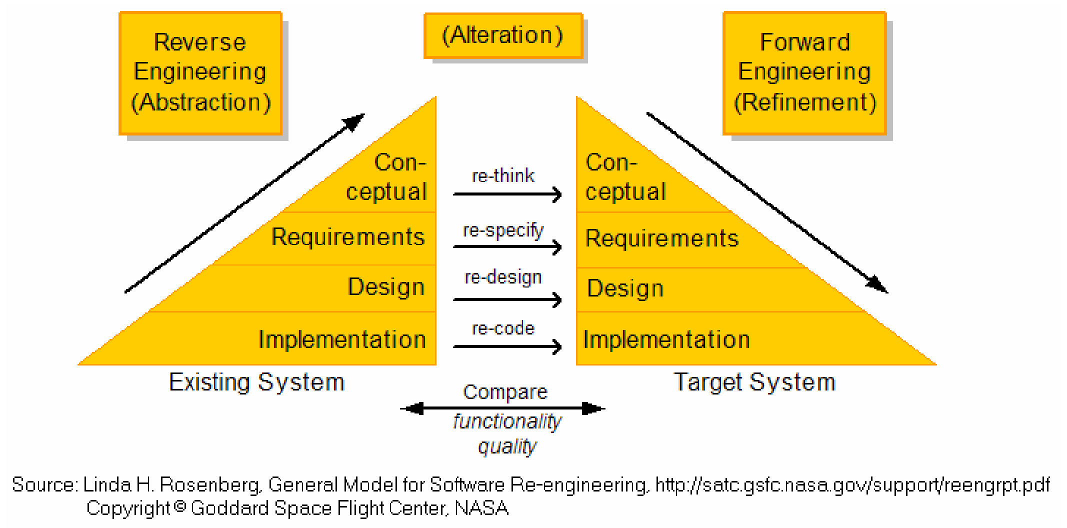

Reverse Engineering (Abstraction)
(Alteration)
Forward Engineering (Refinement)
Con-ceptual
Requirements
Design
Implementation
re-think
re-specify
re-design
re-code
Con-ceptual
Requirements
Design
Implementation
Existing System
Target System
Compare
functionality
quality
Source: Linda H. Rosenberg, General Model for Software Re-engineering, http://satc.gsfc.nasa.gov/support/reengrpt.pdf
Copyright © Goddard Space Flight Center, NASA

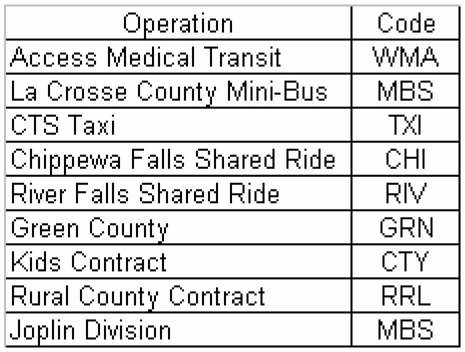
Figure 2.1. General model for software re-engineering

The model in Figure 2.1 applies three phases of re-engineering: abstraction, alteration, and refinement [8]. During the abstraction process, existing information, especially from the source code or detailed design, is elevated to an abstract level, extracting the most important aspects of the application leaving behind low level details such as algorithms and data types. The outcome of the abstraction process is the requirements specifications for the new system. The alteration process changes the system representation without changing the functionality of the existing system. By applying new software techniques and technologies to better represent the existing functionality of the system, one could alter the system representation without losing functionality. Refinement is the opposite of abstraction. Here, the abstract information is replaced with more detailed information resembling software development of new code.

2.1 Existing Software Design

Through reading existing documentation and gathering various facts about the existing software from the project sponsor, it was possible to learn the software's history and the overall system architecture and software design. The original software was written in the Visual FoxPro programming language with data stored in FoxPro free tables - tables not in a database container (DBC). Although the specific dates are not known, the software has gone through a couple of transformations. Originally, the whole system was written as separate applications, and each application was used to manage specific areas of the business. Later, the modules were brought together into an integrated system. However, as the company expanded and provided additional services, the integrated design was not flexible enough to support the changing demands of the business.

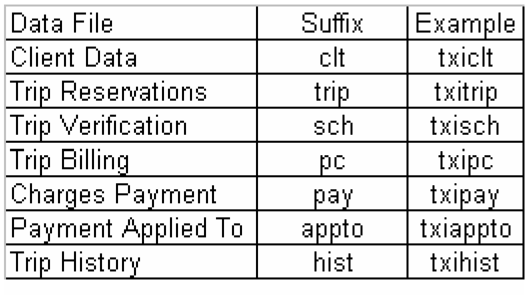
Today, the TripSoft system is comprised of four applications that are used to manage data needed for the taxi and paratransit businesses owned and operated by Top Hat, Inc. Each of Top Hat's businesses is represented as an operation within the system. Each operation is assigned a three-character operation code as listed in Table 1. The operation code is utilized in the client number and data file names.



Operation	Code
Access Medical Transit	WMA
La Crosse County Mini-Bus	MBS
CTS Taxi	TXI
Chippewa Falls Shared Ride	CHI
River Falls Shared Ride	RIV
Green County	GRN
Kids Contract	CTY
Rural County Contract	RRL
Joplin Division	MBS

Table 1. Operation codes

All data files for a particular business are referred to as a system. Each data file name begins with the operation code and ends with a file type suffix. Table 2 illustrates the core data files and file types used for the CTS Taxi system.



Data File	Suffix	Example
Client Data	clt	txiclt
Trip Reservations	trip	txitrip
Trip Verification	sch	txisch
Trip Billing	pc	txipc
Charges Payment	pay	txipay
Payment Applied To	appto	txiappto
Trip History	hist	txihist

Table 2. File types

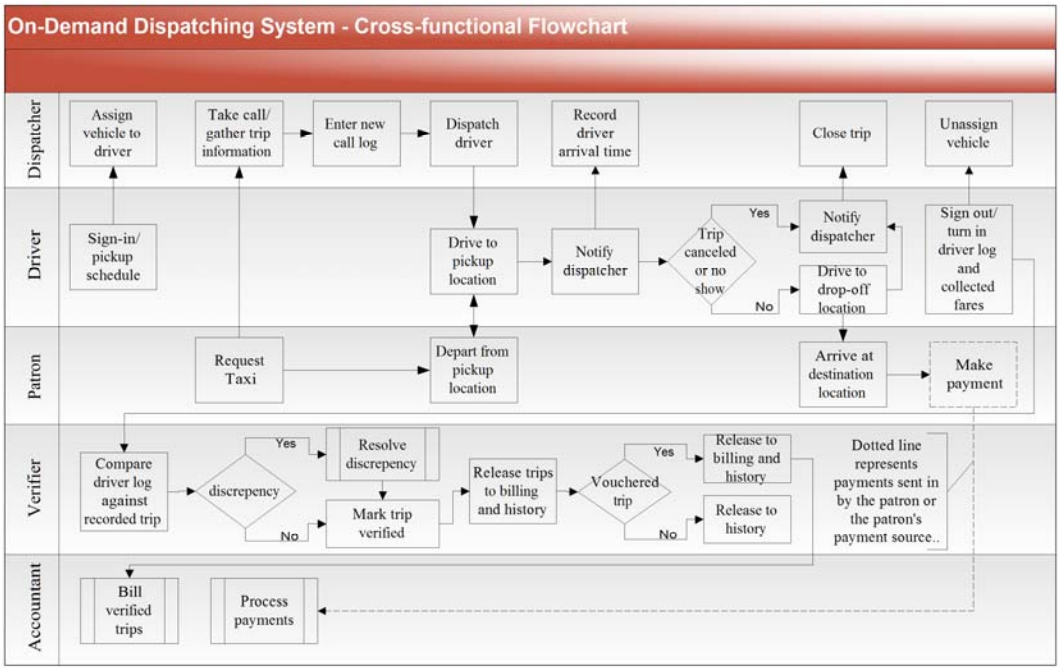
In addition to the data files used by the various operations, there are a number of shared data files, which are common to all operations. Many of these files serve as lookup values (data in dropdown lists) during data entry. The remaining data files contain detailed information such as accident history, driver's hours, driver certifications, and information used for tracking and reporting a variety of metrics.

The free table design of the TripSoft software has been challenging for users of the system and presents many disadvantages. Items such as validation rules, formatting rules, input masks, default values, and triggers have been implemented in code. If any of these items needs to be changed, the application code must be updated. This significantly raises the cost of maintaining the system. Other disadvantages include lack of support for transaction processing, referential integrity having to be implemented in code, and limitation of field names to ten characters in length. In contrast, the benefits of using such a design are that free tables are inherently simple, compatible with other development tools and applications, and easy to deploy.

2.2 Roles and Responsibilities

It is important to understand how different organizational or functional units, such as departments, interact with each other and with the software system. This understanding can play a major role in identifying not only the design aspects of the software, but also will make a developer familiar with the business processes surrounding the software system. In this context, a cross-functional flowchart is particularly useful in showing the relationships between a business process and the organizational or functional unit who is responsible for that process.

A cross-functional flowchart is divided into different lanes describing the control of different organizational or functional units. Shapes within a given lane represent steps in the business process for which the organizational or functional unit is responsible. The cross-functional flowchart in Figure 2.2 shows the responsibilities of each functional unit involved in the on-demand dispatching process.



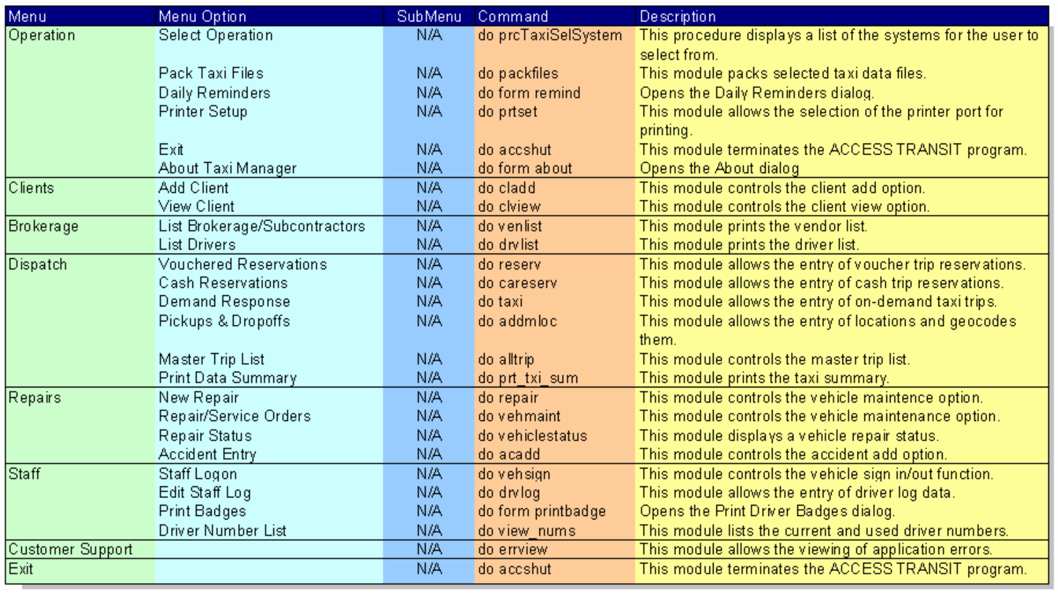
On-Demand Dispatching System - Cross-functional Flowchart
Dispatcher
Driver
Patron
Verifier
Accountant
Assign vehicle to driver
Take call/ gather trip information
Enter new call log
Dispatch driver
Record driver arrival time
Close trip
Unassign vehicle
Sign-in/ pickup schedule
Drive to pickup location
Notify dispatcher
Trip canceled or no show
Yes
No
Notify dispatcher
Drive to drop-off location
Sign out/ turn in driver log and collected fares
Request Taxi
Depart from pickup location
Arrive at destination location
Make payment
Compare driver log against recorded trip
discrepency
Yes
No
Resolve discrepency
Mark trip verified
Release trips to billing and history
Vouchered trip
Yes
No
Release to billing and history
Release to history
Dotted line represents payments sent in by the patron or the patron's payment source..
Bill verified trips
Process payments

Figure 2.2. Cross-functional flowchart of On-Demand Dispatching System

This kind of flowchart is also useful in other ways. For instance, when a question arises, this flowchart can help locate who or which functional unit is responsible for a particular process or module in concern. It can also help identify issues within the current process so they can be addressed before requirements of the target system are finalized.

2.3 Reverse Engineering

The initial challenge to any software re-engineering project is to capture implicit requirements through source code review of the existing system. Since the existing legacy system is menu driven, the developer first compiled a list of menu items from the various modules. Table 3 depicts the menu listing for the Taxi Manager module. The information collected from menu items provides a good reference to the core interface components and their interrelationships as accessed by users of the system.



Menu	Menu Option	SubMenu	Command	Description
Operation	Select Operation	N/A	do prcTaxiSelSystem	This procedure displays a list of the systems for the user to select from.
	Pack Taxi Files	N/A	do packfiles	This module packs selected taxi data files.
	Daily Reminders	N/A	do form remind	Opens the Daily Reminders dialog.
	Printer Setup	N/A	do prtset	This module allows the selection of the printer port for printing.
	Exit	N/A	do accshut	This module terminates the ACCESS TRANSIT program.
	About Taxi Manager	N/A	do form about	Opens the About dialog
Clients	Add Client	N/A	do cladd	This module controls the client add option.
	View Client	N/A	do clview	This module controls the client view option.
Brokerage	List Brokerage/Subcontractors	N/A	do venlist	This module prints the vendor list.
	List Drivers	N/A	do drvlist	This module prints the driver list.
Dispatch	Vouchered Reservations	N/A	do reserv	This module allows the entry of voucher trip reservations.
	Cash Reservations	N/A	do careserv	This module allows the entry of cash trip reservations.
	Demand Response	N/A	do taxi	This module allows the entry of on-demand taxi trips.
	Pickups & Dropoffs	N/A	do addmloc	This module allows the entry of locations and geocodes them.
	Master Trip List	N/A	do alltrip	This module controls the master trip list.
	Print Data Summary	N/A	do prt_txi_sum	This module prints the taxi summary.
Repairs	New Repair	N/A	do repair	This module controls the vehicle maintence option.
	Repair/Service Orders	N/A	do vehmaint	This module controls the vehicle maintenance option.
	Repair Status	N/A	do vehiclestatus	This module displays a vehicle repair status.
	Accident Entry	N/A	do acadd	This module controls the accident add option.
Staff	Staff Logon	N/A	do vehsign	This module controls the vehicle sign in/out function.
	Edit Staff Log	N/A	do drvlog	This module allows the entry of driver log data.
	Print Badges	N/A	do form printbadge	Opens the Print Driver Badges dialog.
	Driver Number List	N/A	do view_nums	This module lists the current and used driver numbers.
Customer Support		N/A	do errview	This module allows the viewing of application errors.
Exit		N/A	do accshut	This module terminates the ACCESS TRANSIT program.

Figure 2.3. Menu listing for Taxi Manager

Next, the developer looked behind the scenes to recapture the essential design, structure and content of each interface component. It is important to capture not only the technical relationships and interactions, but also the information and business rules that have been used to run the business. Since the TripSoft system was designed to use free tables, business rules were also implemented in the code. This made it difficult to distinguish between functional requirements that govern the system’s behavior, and business rules that enforce policies of the organization. The important distinction to remember about business rules is that they are more susceptible to change than that of functional requirements. Identifying the business rules and planning to move them outside of the application code will help reduce maintenance costs of the new target system.

As part of the abstraction process, the developer referred to the command column of the menu listing and began to extract the detailed information. In this way, it was decided to convert the menu listings into software requirements not already defined by the sponsor. Included in this menu listing were various reports accessed by users of the system. Reports are another great resource for extracting system requirements.

With reports, organizations display information in a logical and concise form used to communicate information to clients and throughout the business. This makes reports a powerful resource for gathering system requirements. Reports often contain the most important attributes and metrics the business needs to capture. For this project, the attributes used in reports helped identify the key requirements for data storage. After all, having a good database design is not helpful if you do not store the correct information.

3. Data Storage

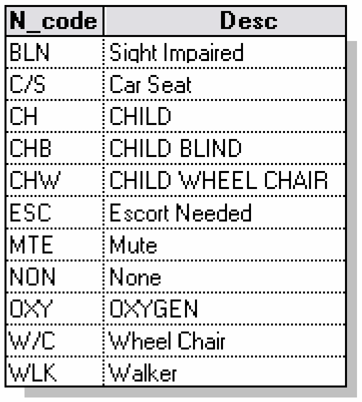
The TripLogic project required more than re-engineering the software system. It also required a new database design. The developer had several goals for the new design. However, the primary concerns were to reduce data redundancy, increase data integrity, provide concurrency control, and to make data retrieval convenient and efficient. One of the main reasons for choosing a database management system (DBMS) as the data store is that it provides the platform that addresses all the concerns raised previously. It provides central control of both data and the programs accessing that data.

By using a DBMS, the developer was able to separate the data access layer from the presentation and business layers of the new target system's architecture. As an example, business rules were implemented as objects in the database rather than code in the application. With the flexibility, scalability, and reliability of a DBMS, the new TripLogic software system will be able to accommodate change in business needs and processes at a fraction of the cost currently incurred with the existing system.

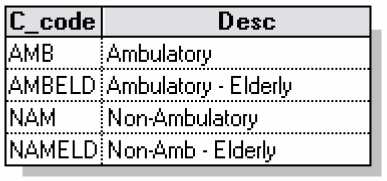
3.1 Database Architecture

During the abstraction phase of the re-engineering process, the developer came across several blocks of application code used for writing to and retrieving data from the FoxPro free tables. It became apparent quite early in the process that there were many opportunities to compress similar free tables into a single database table in order to reduce redundancy.

As mentioned earlier in Chapter 2, many of the free tables contain lookup values. Each of the tables contains two attributes named Code and Description. In addition to the free tables, there are several places in the application code where lookup values are hard-coded. Figure 3.1 shows how the existing system stores the client needs and classification lookup values in separate free tables.



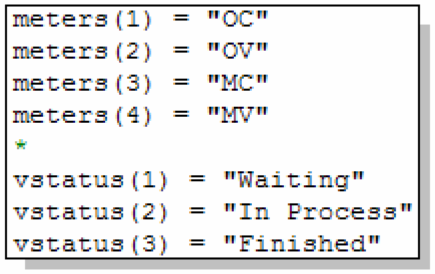
N_code	Desc
BLN	Sight Impaired
C/S	Car Seat
CH	CHILD
CHB	CHILD BLIND
CHW	CHILD WHEEL CHAIR
ESC	Escort Needed
MTE	Mute
NON	None
OXY	OXYGEN
W/C	Wheel Chair
WLK	Walker



C_code	Desc
AMB	Ambulatory
AMBELD	Ambulatory - Elderly
NAM	Non-Ambulatory
NAMELD	Non-Amb - Elderly

Figure 3.1. Listings of client needs and classifications

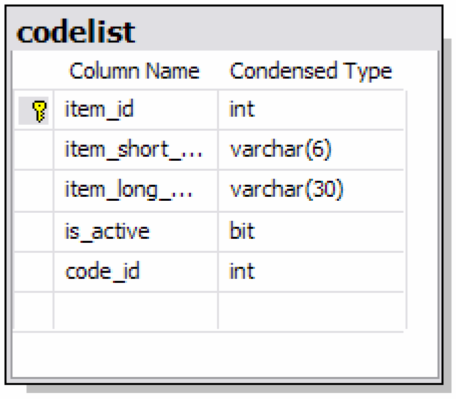
The challenge with hard-coded values is that they are often implemented differently as shown in Figure 3.2. In this example, there are two arrays called ‘meters’ and ‘vstatus’ respectively. The ‘meters’ array is populated with code values that have no descriptions while the ‘vstatus’ array is populated with descriptions that have no code values. Generating selection lists in this manner introduces inconsistencies in the user interface and should be avoided.



```
meters(1)  = "OC"
meters(2)  = "OV"
meters(3)  = "MC"
meters(4)  = "MV"
*
vstatus(1)  = "Waiting"
vstatus(2)  = "In Process"
vstatus(3)  = "Finished"
```

Figure 3.2. Hard-coded values in application code

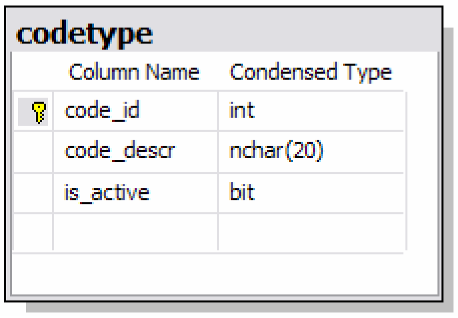
Since these lookup values all had similar attributes, the developer compressed them into a single database table and added additional attributes to make data retrieval more efficient and flexible. As presented in Figure 3.3, the developer structured the database table with five attributes. The ‘item_id’ attribute is an identity column used as the primary key of the table. The ‘item_short_descr’ and ‘item_long_descr’ are the columns that represent the code and description attributes of the existing free tables. An attribute called ‘is_active’ was added to allow users to check and to deactivate codes that may have been used in the past but are no longer needed. If the developer were to allow users to delete codes that are referenced in other tables, data integrity issues would arise. Finally, the developer added a ‘code_id’ field that represents the type of item being defined.


codelist

Column Name	Condensed Type
item_id	int
item_short_...	varchar(6)
item_long_...	varchar(30)
is_active	bit
code_id	int

Figure 3.3. Codelist database table structure

The attribute ‘code_id’ is an integer value that is not very meaningful by itself. To give ‘code_id’ some meaning, a table called ‘codetype’ was created to store the descriptions of each ‘code_id’. The complete structure of the codetype database table is presented in Figure 3.4. By creating these two database tables, it was possible to combine the contents of four existing free tables and avoid hard-coding values in the application code. The final step was to provide a convenient and efficient way to retrieve data from these two tables.

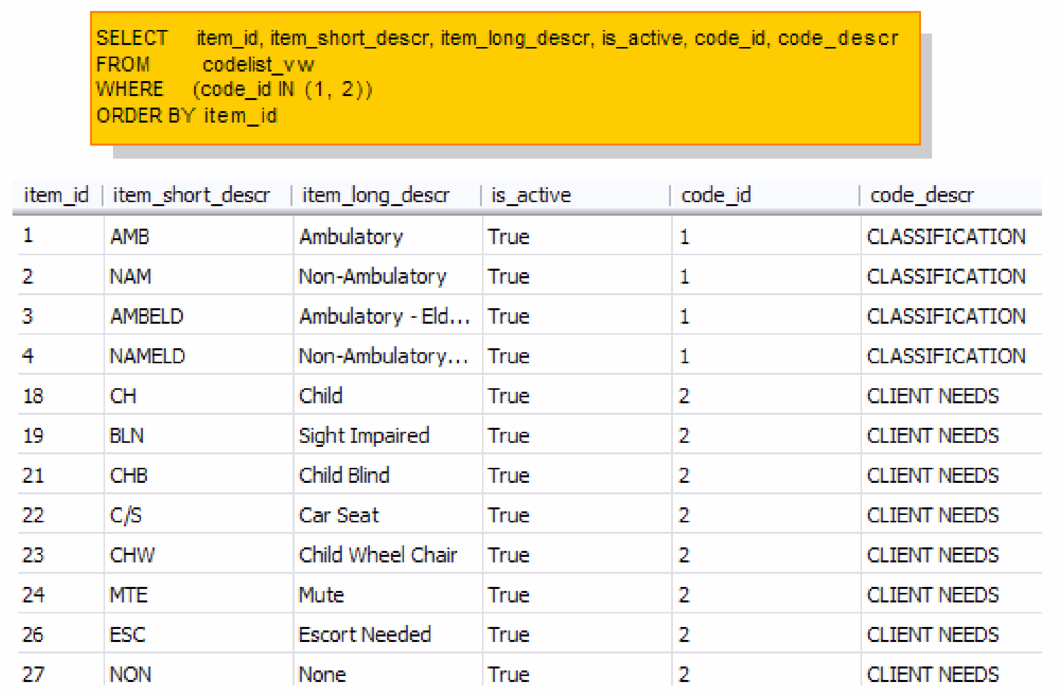


codetype

	Column Name	Condensed Type
🔑	code_id	int
	code_descr	nchar(20)
	is_active	bit

Figure 3.4. Codetype database table structure

Using a database view, the ‘codetype’ and ‘codelist’ tables were joined by their common attribute ‘code_id’. From this, a complete listing of active and inactive codes in the new system was obtained. A subset of data from the ‘codelist_vw’ view is shown in Figure 3.5.



```
SELECT    item_id, item_short_descr, item_long_descr, is_active, code_id, code_descr
FROM       codelist_vw
WHERE    (code_id IN (1, 2))
ORDER BY item_id
```

item_id	item_short_descr	item_long_descr	is_active	code_id	code_descr
1	AMB	Ambulatory	True	1	CLASSIFICATION
2	NAM	Non-Ambulatory	True	1	CLASSIFICATION
3	AMBELD	Ambulatory - Eld...	True	1	CLASSIFICATION
4	NAMELD	Non-Ambulatory...	True	1	CLASSIFICATION
18	CH	Child	True	2	CLIENT NEEDS
19	BLN	Sight Impaired	True	2	CLIENT NEEDS
21	CHB	Child Blind	True	2	CLIENT NEEDS
22	C/S	Car Seat	True	2	CLIENT NEEDS
23	CHW	Child Wheel Chair	True	2	CLIENT NEEDS
24	MTE	Mute	True	2	CLIENT NEEDS
26	ESC	Escort Needed	True	2	CLIENT NEEDS
27	NON	None	True	2	CLIENT NEEDS

Figure 3.5. Listing from the codelist_vw database view

When data is in the form of a view, it can be filtered, sorted, and grouped as required by an application. Views simplify interactions with the database system by hiding the low-level data structures and table relations from users.

Another way to provide a consistent and efficient interface between applications and the data is through the use of stored procedures. A stored procedure is similar to a procedure in other programming languages, but it is compiled in the database and stored as a collection of SQL statements. It can have both input and output parameters, contain programming statements that perform operations in the database including calling other procedures, and return a status value to indicate success or failure.

The design approach was to use stored procedures as the primary interface between the TripLogic software and the database, and use views to simplify the stored procedure code and provide data to reports. With this approach, the developer not only created a consistent interface between the application and database but also was able to move the business rules that were extracted during the re-engineering process into the appropriate stored procedures. This is a good place to implement business rules since they can be applied before or after data in the database has been manipulated by the application, and can be changed at any time without affecting the application.

3.2 Referential Integrity

One of the primary issues with the free table structure used in the existing software is that referential integrity was implemented in code. With a RDBMS, relationship constraints among the tables are established to enforce those relations. This type of constraint is called a *referential integrity constraint*, often referred to as a *foreign key constraint*. For example, there exists a foreign key constraint between the ‘codetype’ and ‘codelist’ database tables discussed earlier. In the TripLogic SQL Server database, this foreign key relationship is represented diagrammatically as illustrated in Figure 3.6.

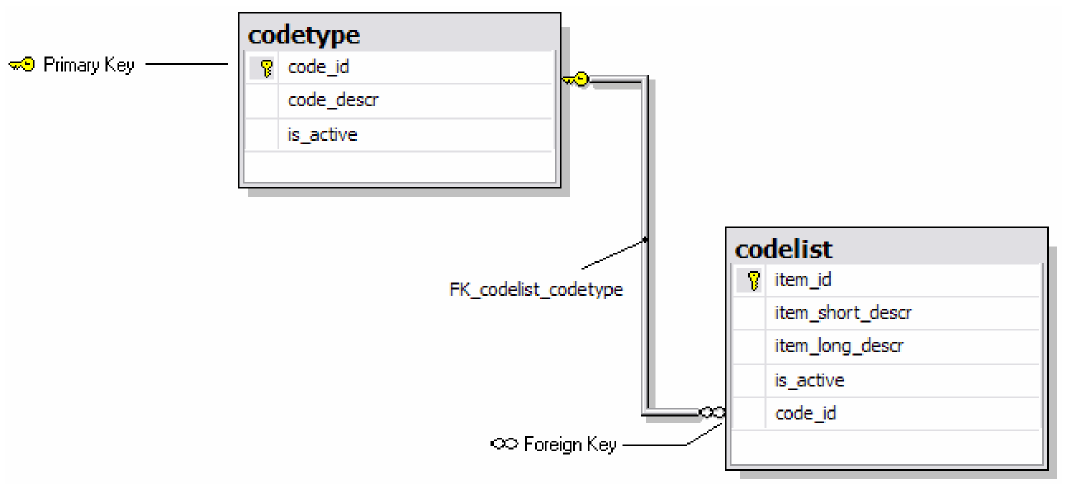


Figure 3.6. Diagrammatic representation of a foreign key constraint

This constraint guarantees that a matching 'code_id' value exists in the 'codetype' table for every 'code_id' value in the 'codelist' table. In addition to controlling the data that can be stored in the 'codelist' table, the constraint also controls changes to data in the 'codetype' table. If a user deletes a code type from the 'codetype' table, and the code type's ID is used for items in the 'codelist' table, the relational integrity between the two tables is broken; the deleted code type's items are orphaned in the 'codelist' table without a link to the data in the 'codetype' table. A foreign key constraint prevents this situation. The constraint enforces referential integrity by ensuring that changes cannot be made to data in the 'codetype' table if those changes invalidate the link to data in the 'codelist' table.

4. The TripLogic Application

This chapter explains the core requirements of the TripLogic application as defined by the sponsor and extracted from the existing software's source code. It also presents an overview of the challenges encountered during the design, development and implementation phases of the waterfall software life-cycle, and the application architecture in further detail

4.1 Requirements Specifications

After compiling a list of functional and non-functional requirements from both the sponsor and those captured in the abstraction process, the developer performed a requirements analysis. This analysis was done to thoroughly understand the project without worrying about the implementation details and to establish the services the system should provide, along with the constraints under which it must operate. Performing an in-depth analysis of the requirements helps reduce software defects from being carried into later phases of the software life-cycle, and significantly reduces the overall project development costs. Statistically, finding and fixing software defects after deployment is often 100 times more expensive than detecting and fixing them during the requirements and design phase [4].

One aspect of the requirements analysis process involves performing a feasibility study. This ensures that the identified user needs can be satisfied by using current software technologies, that the proposed system is cost-effective from a business perspective, and that the project can adhere to budgetary constraints [4].

Once the analysis was complete, the developer created a specification document conforming to the IEEE Standards on Requirements Specification [3]. This document served as the requirements for this project and was analyzed and updated to correct any inconsistencies and ambiguities found as the project progressed.

4.2 Project Challenges

The most challenging aspect to any software engineering project is satisfying all of the customer’s needs; yet, today’s rapid advancement in development tools and technologies often compounds this challenge.

As an example, the existing software referenced several reports that had to be recreated using a new reporting tool. Early in the design phase, the choice to use Crystal Reports as the reporting tool seemed logical because it was integrated into the development software planned for developing the TripLogic product. Later, during the design phase, the developer learned that Microsoft announced a new reporting product called SQL Server Reporting Services (SSRS) bundled with SQL Server. The developer was now challenged with the decision to stay with the original design and use Crystal Reports .NET included with the development software or use SSRS included with SQL Server.

After careful consideration that Microsoft would no longer include Crystal Reports with its software development tools, and analyzing the capabilities of SSRS, the developer determined SSRS would be sufficient for the reporting needs of this project. However, making the decision to use SSRS as the reporting tool introduced risks into the project plan. The developer had to learn a new reporting tool risking the project’s schedule, and limitations in the new reporting tool could have a negative effect on the flexibility of the TripLogic software. This demonstrates how changes in development tools and technologies can affect a software project even when it is well into the design phase of the software life-cycle.

One major requirement for the TripLogic software was to add a Global Positioning System and Automated Vehicle Location (GPS/AVL) feature into the software. By adding this feature, the sponsor plans to enable the dispatcher to capture the current geographical location, speed, and heading of each dispatched driver. During the feasibility study, the developer realized the challenges of adding such a feature to the application.

Today, when someone mentions GPS, one can immediately visualize a device that displays a road map containing his/her current geographical location. It is easy to overlook the fact that a GPS receiver's data is first parsed and interpreted before it is plotted on a map. The developer had to first study the National Marine Electronics Association (NMEA) 0183 standard for interfacing marine electronics devices (used by most GPS receivers) before designing an algorithm to read and interpret GPS data.

After reading several articles on NMEA 0183, it was evident that an algorithm could be developed, but would require thorough testing. Since the sponsor did not have GPS hardware to test the algorithm thoroughly, the developer was forced to look for alternative solutions. One logical alternative was to integrate existing GPS capable software with the TripLogic product. Due to time and budgetary constraints, the product search was limited to mapping software. The key components required were GPS/AVL capabilities and a rich Application Programming Interface (API) providing for a seamless integration. The selected product was Microsoft MapPoint 2004. After the product was selected, Microsoft announced the release of its MapPoint web service but the developer chose not to implement this technology to avoid adding further risk to the project plan. The TripLogic design would allow for this enhancement to be implemented later.

4.3 Application Architecture

The TripLogic software is written in C# using Microsoft Visual Studio 2005 Express Edition. It has been integrated with Microsoft MapPoint 2004 to provide GPS/AVL capabilities, to display routing information, and to estimate trip time and costs. The database is run on the next generation Microsoft SQL Server Desktop Engine (MSDE) called SQL Server 2005 Express Edition.

The TripLogic application communicates with Microsoft MapPoint 2004 through an ActiveX control. With the ActiveX control, the developer has access to the Microsoft base-level North America map and can customize the routing and data mapping functions. Using the MapPoint application installed on the dispatcher’s machine, a GPS receiver can be configured, provided its input/output format is set to support the NMEA 0183 version 2.0 or later format. Once a GPS receiver is configured, MapPoint checks for the driver’s location every second and displays it on the map. In later versions of MapPoint, the developer has direct access to a GPS task pane allowing a dispatcher to set GPS options and calculate routes from the driver’s current position, directly from the host application.

Using the Report Designer installed with SQL Server 2005 Express, the developer was able to create report models (used for ad-hoc reporting) and report definition files. These report models and definition files can be deployed to a report server (remote mode) or copied to a computer where SQL Server Reporting Services has not been installed (local mode). To render the reports, the TripLogic application uses a WinForms ReportViewer control. The developer is provided with a full-featured API to control report functionality at run time.

The TripLogic architecture requires two databases. The first database is installed with SQL Server Reporting Services and stores all of the data required by Reporting Services. The TripLogic software uses the second database to store application and business data. Both databases can be installed in the same instance of SQL Server; however, a separate report server is recommended for improved performance and scalability. As illustrated in Figure 4.1, the current architecture has both databases installed on the same instance of SQL Server.

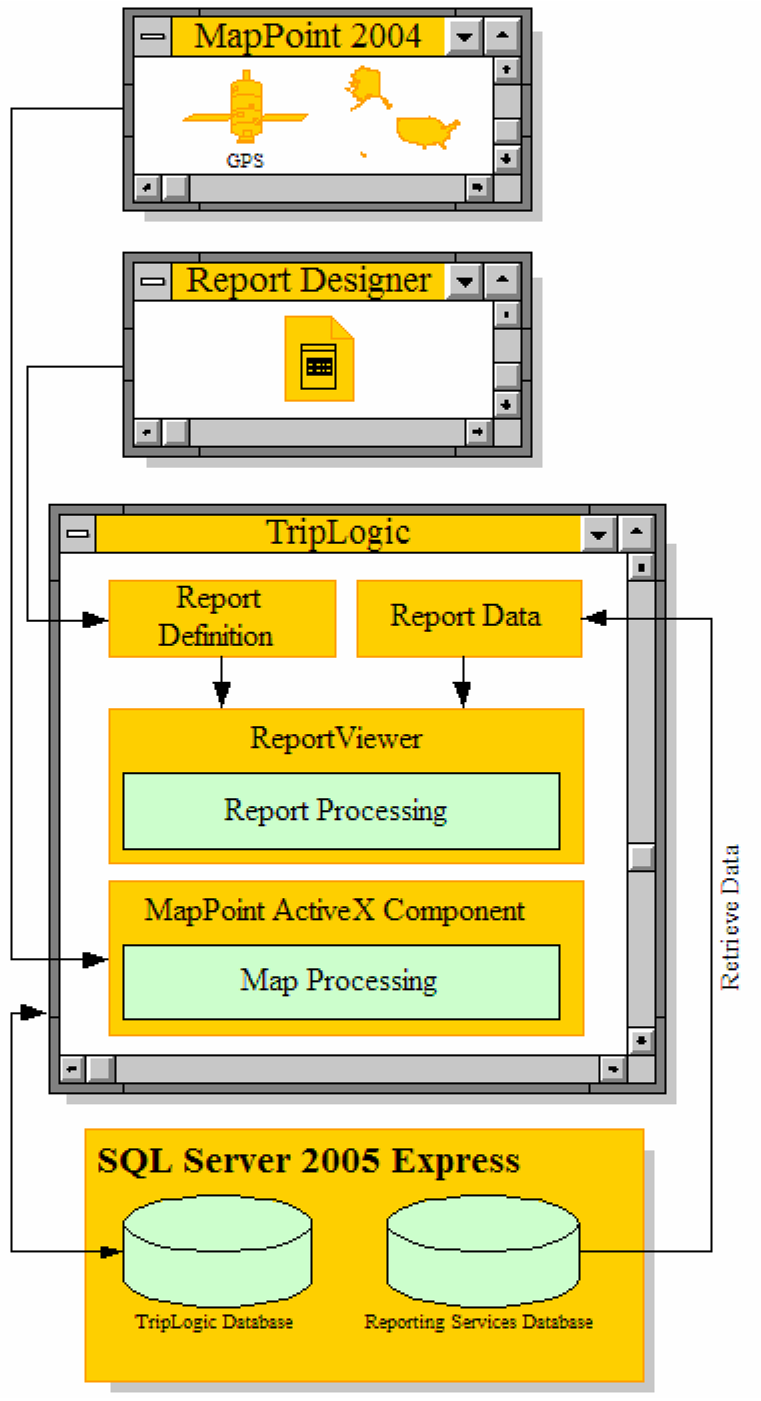
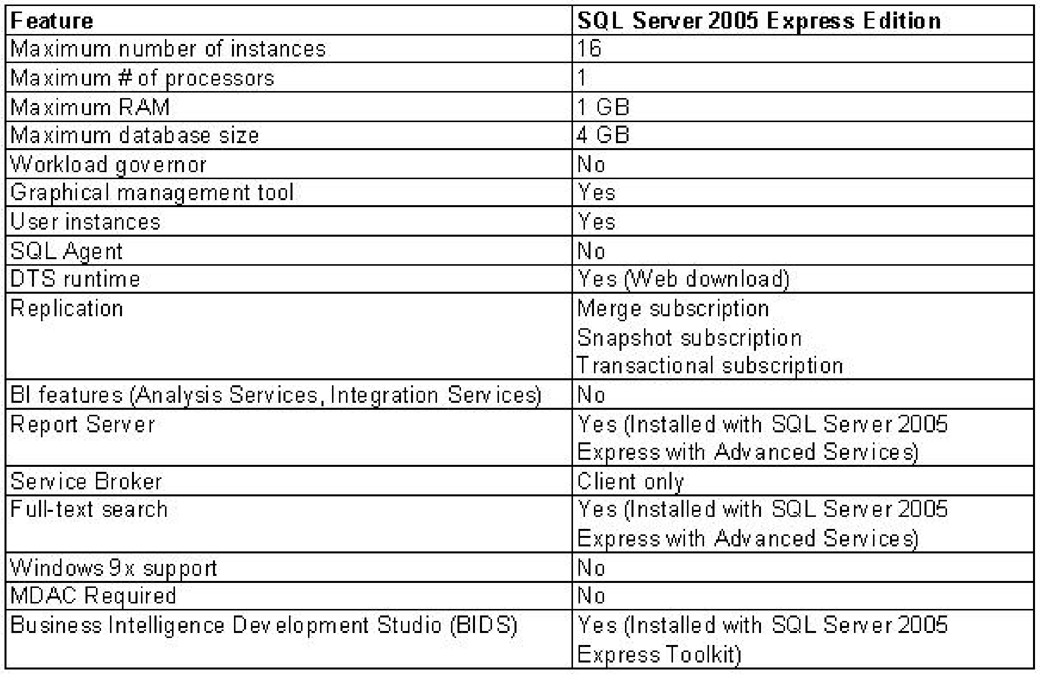


Figure 4.1. TripLogic Architecture

It is important to mention the limitations of SQL Server 2005 Express. As shown in Table 3, the primary limitations pertain to memory, number of supported processors, and database size. However, given the sponsor's budgetary constraints, small number of users, and existing hardware, this platform provides the best overall features and meets all of the sponsor's current needs.



Feature	**SQL Server 2005 Express Edition**
Maximum number of instances	16
Maximum # of processors	1
Maximum RAM	1 GB
Maximum database size	4 GB
Workload governor	No
Graphical management tool	Yes
User instances	Yes
SQL Agent	No
DTS runtime	Yes (Web download)
Replication	Merge subscription Snapshot subscription Transactional subscription
BI features (Analysis Services, Integration Services)	No
Report Server	Yes (Installed with SQL Server 2005 Express with Advanced Services)
Service Broker	Client only
Full-text search	Yes (Installed with SQL Server 2005 Express with Advanced Services)
Windows 9x support	No
MDAC Required	No
Business Intelligence Development Studio (BIDS)	Yes (Installed with SQL Server 2005 Express Toolkit)

Table 3. SQL Server 2005 Express Features

Since SQL Server 2005 Express can be installed and redistributed without licensing fees, it provides the sponsor with very low startup costs and the ability to upsize in the future.

5. User Interface Design

The TripLogic graphical user interface enables the users to work with multiple forms at the same time through its multiple document interface (MDI) design. When the application is opened, the user is presented with the application's main window consisting of a Menu Bar, Navigation Menu, and Workspace. To interact with the application, a user can select menu options from the Menu Bar or Navigation Menu. If the selected menu option is attached to one of the data entry screens, the screen is opened in the main window's Workspace and becomes the active screen.

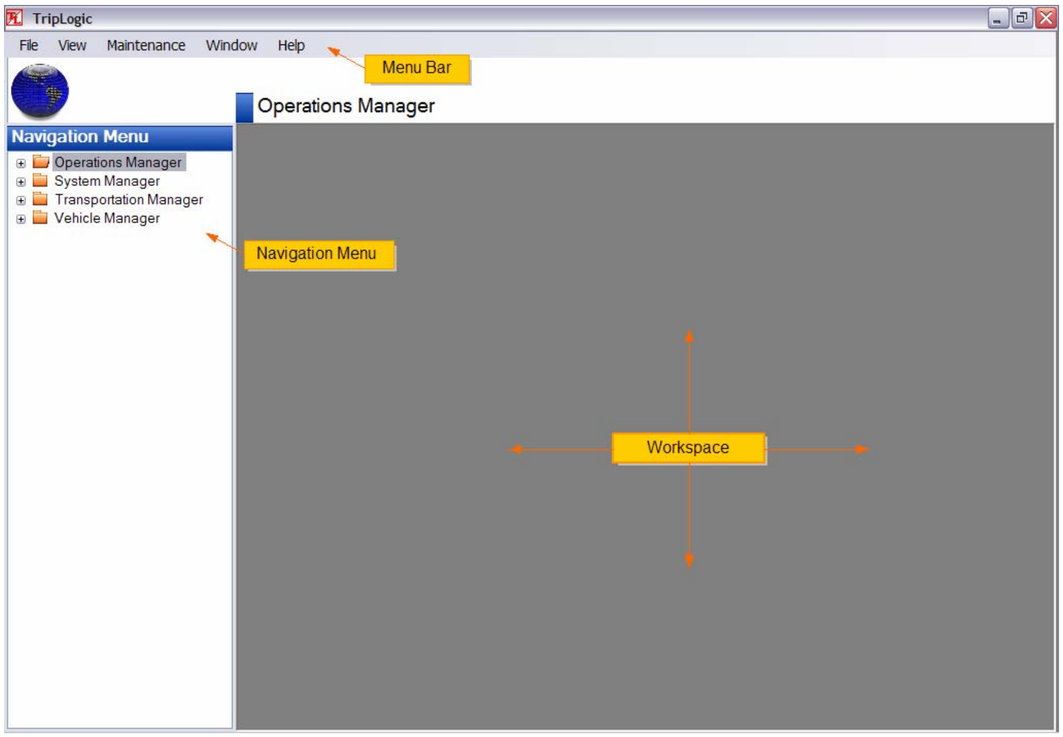
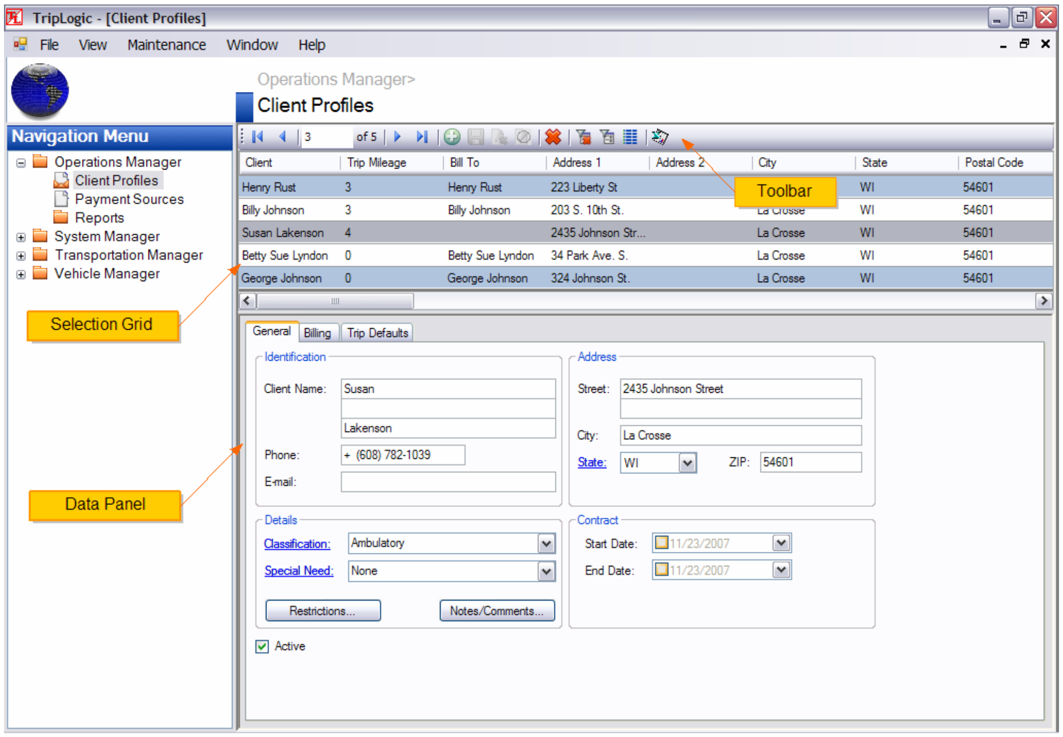


Figure 5.1. Main Application Window

A user can open as many data entry windows as necessary; however, the application only allows a single instance of each data entry screen to be opened. This prevents a user from starting work in two or more instances of the same screen, a source of frustration for users working in MDI applications.

The Navigation Menu has been grouped into categories representing the four applications that make up the current system. This helps users learn the new application more easily because they can expect similar features to exist under categories with the same names as their legacy applications. To allow for additional workspace, the user can hide the Navigation Menu and use the Maintenance menu on the Menu Bar to navigate to the various data entry screens.

Each data entry screen is composed of three parts, a Toolbar, Selection Grid, and Data Panel. The Toolbar contains buttons for navigating through current entries, adding new entries, saving changes, canceling changes, deleting entries, filtering the Selection Grid entries, showing or hiding the Selection Grid, and exporting data to Excel. When a data entry screen is opened in the Workspace, the Selection Grid is populated with all of the existing entries. For example, when the Client Profile screen is opened, the information for all clients in the system is loaded into the Selection Grid. The Selection Grid can then be used to filter (search) the existing entries and navigate between entries. When an entry in the Selection Grid is activated, it becomes editable in the Data Panel. When a user simply drags a column in the Selection Grid to move it to a new position, the column positions will be saved automatically when the screen is closed. By hiding both the Navigation Menu and Selection Grid, the user is given the maximum amount of workspace. When a data entry screen is closed, the state of its Selection Grid is automatically saved. Similarly, when the application is closed, the Navigation Menu's state is saved. Therefore, the next time the application is opened or a data entry screen is opened, it will look exactly as it did when it was last closed. The Data Panel is where the user enters new data or updates existing information. Common attributes displayed in the Data Panel are grouped together and labeled to assist the user with identifying the contents of the data entry fields. For example, the client name, phone, and e-mail attributes are grouped together and labeled ‘Identification’ to inform the user that these attributes identify a specific client.



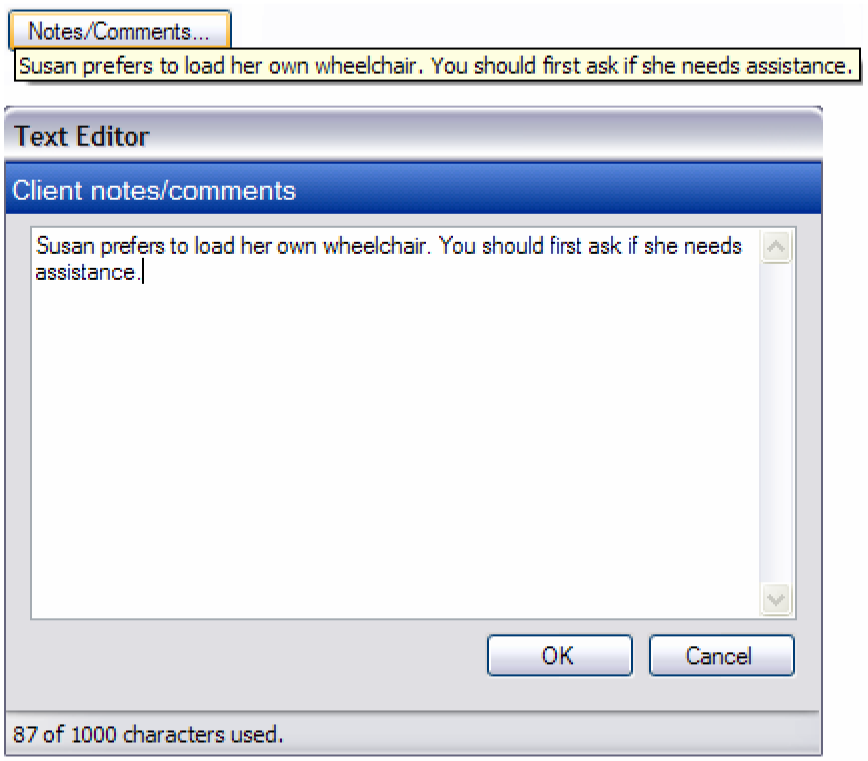
Client	Trip Mileage	Bill To	Address 1	Address 2	City	State	Postal Code
Henry Rust	3	Henry Rust	223 Liberty St			WI	54601
Billy Johnson	3	Billy Johnson	203 S. 10th St.		La Crosse	WI	54601
Susan Lakenson	4		2435 Johnson Str...		La Crosse	WI	54601
Betty Sue Lyndon	0	Betty Sue Lyndon	34 Park Ave. S.		La Crosse	WI	54601
George Johnson	0	George Johnson	324 Johnson St.		La Crosse	WI	54601

Client Name: Susan
Lakenson
Phone: + (608) 782-1039
E-mail:
Street: 2435 Johnson Street
City: La Crosse
State: WI
ZIP: 54601
Classification: Ambulatory
Special Need: None
Start Date: 11/23/2007
End Date: 11/23/2007
✓ Active

Figure 5.2. Data Entry Screen for Client Profiles

Figure 5.2 shows the Client Profiles data entry screen opened in the main window’s Workspace. The client, Susan Lakenson, is selected in the Selection Grid and editable in the Data Panel. As shown in the grouped attributes labeled ‘Details’, all data entry fields that allow the user to select a value from a list will be labeled with a special control called a LinkLabel that behaves similarly to a link on a web page. When pressed, it will open the corresponding data entry screen as a modal dialog. This provides users with the ability to directly add or modify entries in the selection lists while they are in the process of entering information in the active data entry screen. This design keeps the users from having to remember which data entry screen contains the source for values in the selection lists and significantly reduces data entry time.

The TripLogic graphical user interface also avoids taking up valuable screen space by replacing large, multi-line text fields with buttons. As illustrated in Figure 5.3, when the mouse is moved over a button, the first 100 characters of underlying text will be displayed to the user. If the user needs to see the complete text or make changes, she would press the button to open up a text editor that shows the user how many characters can be typed into the field as well as how many characters have already been entered.



Susan prefers to load her own wheelchair. You should first ask if she needs assistance.

Susan prefers to load her own wheelchair. You should first ask if she needs assistance.

87 of 1000 characters used.

Figure 5.3. Multi-line Text Fields Replaced With Text Editor

Another feature the TripLogic interface provides is the ability for a user to drag a trip from the Trip Profiles screen and drop it onto the Maps screen. When a trip is dropped onto the map, the route and driving directions are automatically generated from the trip information. If a driver has been assigned to the trip before it is dragged to the Maps screen, the route will be added to that driver's pushpin set. This allows a dispatcher to easily view the trips a particular driver has been assigned to during the scheduled shift. Since MapPoint only allows one active route per map, the dispatcher will need to drag a trip onto the Maps screen to recalculate the route and driving directions. However, the application will not add the trip to the driver's pushpin set if it already exists.

Based on requirements from the sponsor, the application had to provide a mechanism to auto-fill values whenever possible. For example, when an employee profile is created, the employee is associated with a company. If that employee is chosen from a selection list on a data entry screen, the application will check to see what company the employee belongs to and automatically fill in the company field, if applicable. Similarly, when a client is selected from a list, any client information the data entry screen requires will be populated by the system. This feature reduces typing errors and saves the user's valuable time.

6. Continuing Work

As stated earlier, this project was limited to re-engineering the demand-response dispatching portion of Top Hat's existing software product. Even after the sponsor agreed to reduce the scope of this project, some requirements were not implemented or fully tested. The list below denotes those requirements that were considered during the design of the current project, but have not yet been implemented or fully tested.

- Ability to record driver accidents
    - Injuries
    - Driving conditions
    - Insurance information
- Ability to record incidents
    - Incidents between employees
    - Incidents between driver and patron(s)
    - Complaints about service
- Capture current weather conditions
    - Metrics to help enhance trip estimating based on weather conditions
- Daily Reminders
    - Dispatcher communications
    - High importance trip notifications
- GPS/AVL
    - No GPS receiver was available for testing

- Security
    - Application level security
    - Database level security

Later phases of the software will need to address the paratransit businesses owned by Top Hat as well as its vehicle repair business. These businesses introduce significant design challenges such as handling county and government regulations, trip reservations, different trip types such as round trips and multi-leg trips, vehicle work orders, vehicle maintenance scheduling, and vehicle parts inventory.

Other goals for future releases of the software are to add payroll and accounts payable functions to the software. An alternative to these features is to provide the ability to export data to existing financial software such as Intuit’s QuickBooks and BusinessWorks by Sage Software, Inc.

The sponsor’s long-term goal for this product is to add mobile data terminals so that calls are sent directly to the taxis from the application and to allow credit card transactions to be securely processed. Based on the requirements of these features, it would be advantageous to convert TripLogic into a web-based application. This would not only allow for the application to subscribe to public services for credit card transaction processing and provide support for mobile terminals, but also allow integration with location-based web services such as maps, driving directions and proximity searches.

7. Conclusion

In conclusion, the TripLogic software was re-engineered from existing legacy software which was expensive to maintain and unable to support the changing needs of Top Hat’s diverse businesses. The TripLogic software product has automated many of the data entry tasks resulting in significant time-savings for users. It provides a consistent interface to the application data that is enforced by referential integrity constraints specified in the database. TripLogic’s architecture provides for low development and deployment costs through its selection of development tools and platforms while ensuring the product remains reliable and scalable.

The developer’s cost effective solution has introduced limitations into the software but, in contrast, has given the sponsor insight into several technologies that will enable her to make knowledgeable decisions regarding the future of TripLogic.

8. Bibliography

[1] http://msdn2.microsoft.com/en-us/library/wxt2cwcc(VS.80).aspx, *Connecting to Data in Visual Studio Overview*, Microsoft Corporation, 2007.

[2] www.tophatinc.com, *The History of TopHat Inc.*, TopHat Inc., 2007.

[3] Institute of Electrical and Electronics Engineers, Inc., New York, NY, USA. *IEEE Recommended Practice for Software Requirements Specifications*, 1998. IEEE Standard 830-1998.

[4] Barry Boehm and Victor R. Basili, "Software Defect Reduction Top 10 List", *Computer*, vol. 34, no. 1, pp. 135-137, Jan., 2001.

[5] Andrew Flick and Jason Beres, “Spice Up Your Windows Forms”, Visual Studio Magazine, vol. 14, no. 13, pp. 18-25, Nov., 2004.

[6] Jesse Liberty, *Programming C#* (Second Edition), O’Reilly & Associates, Inc., 2002.

[7] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems* (Second Edition), The Benjamin/Cummings Publishing Company, Inc., 1994.

[8] Linda H. Rosenberg, *Software Re-engineering*, Goddard Space Flight Center, NASA, 1996.

[9] Henry F. Korth and Abraham Silberschatz, *Database System Concepts* (Second Edition), McGraw-Hill, Inc., 1991

[10] Ian Sommerville, *Software Engineering* (Third Edition), Addison-Wesley Publishing Company, 1989.

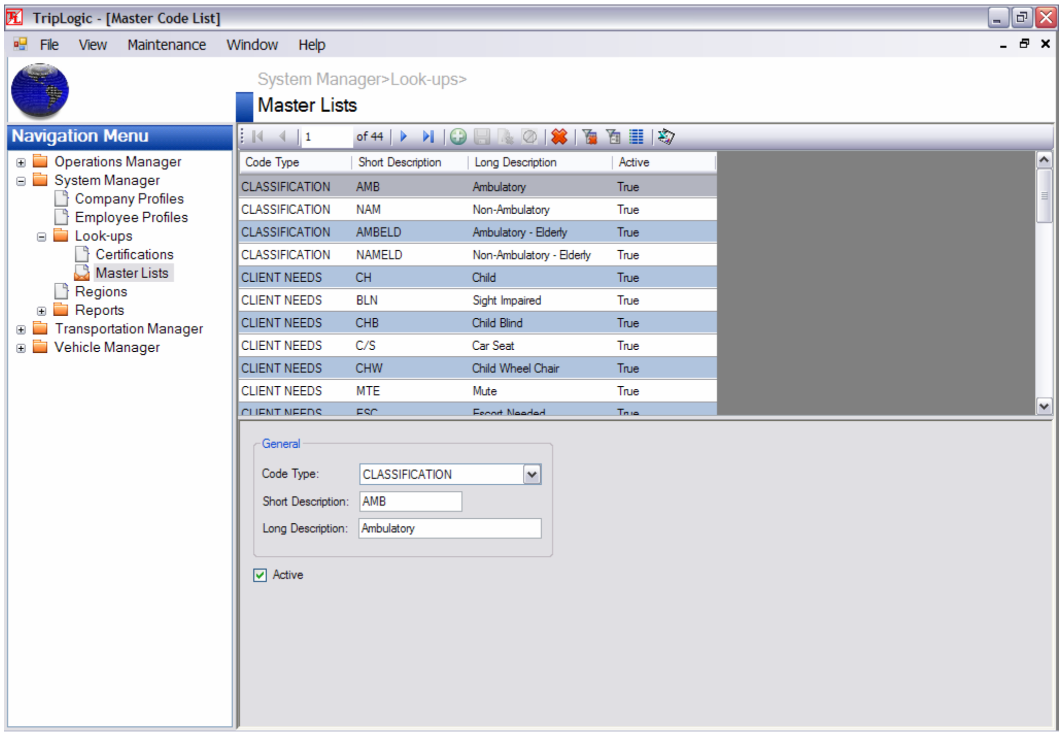
[11] Joshua Trupin, “Pocket PC Migrating a GPS App from the Desktop to eMbedded Visual Basic 3.0”, MSDN Magazine, vol. 16, no. 1, pp. 107-114, Jan., 2001.

[12] Mickey Williams, *Microsoft Visual C#* (core reference), Microsoft Press, 2002.

9. Appendices

Appendix A. Sample TripLogic Data Entry Screen Shots

TripLogic Master Code List Window



Code Type	Short Description	Long Description	Active
CLASSIFICATION	AMB	Ambulatory	True
CLASSIFICATION	NAM	Non-Ambulatory	True
CLASSIFICATION	AMBELD	Ambulatory - Elderly	True
CLASSIFICATION	NAMELD	Non-Ambulatory - Elderly	True
CLIENT NEEDS	CH	Child	True
CLIENT NEEDS	BLN	Sight Impaired	True
CLIENT NEEDS	CHB	Child Blind	True
CLIENT NEEDS	C/S	Car Seat	True
CLIENT NEEDS	CHW	Child Wheel Chair	True
CLIENT NEEDS	MTE	Mute	True
CLIENT NEEDS	ESC	Escort Needed	True

General

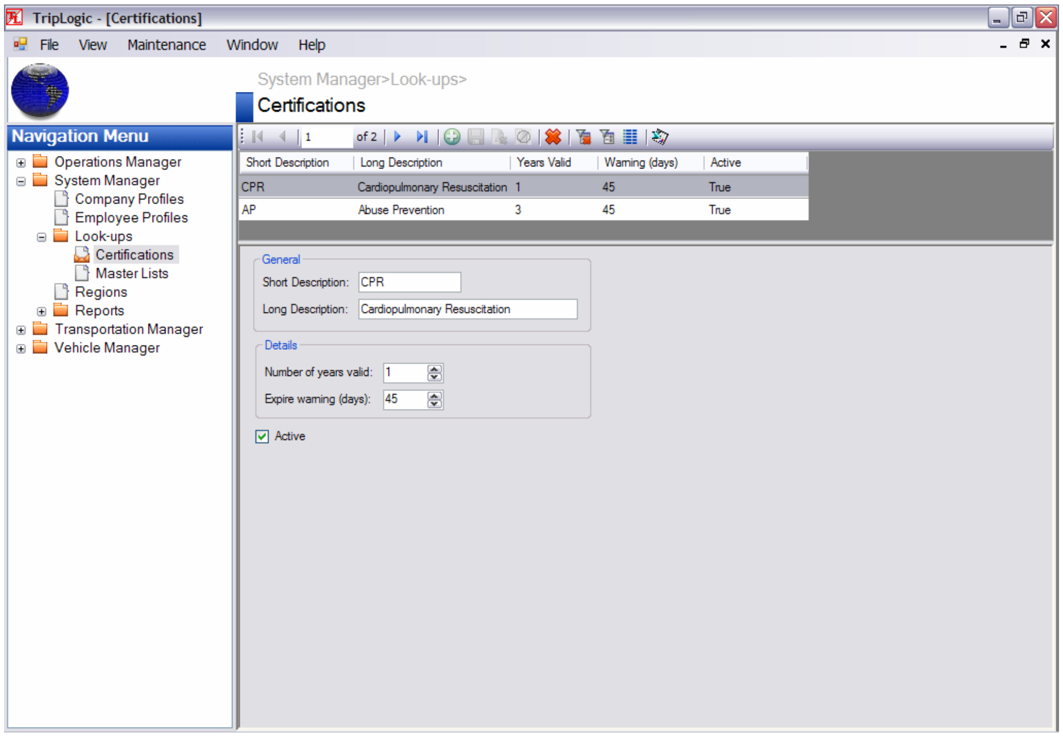
Code Type: CLASSIFICATION

Short Description: AMB

Long Description: Ambulatory

[✓] Active

TripLogic Driver Certifications Window



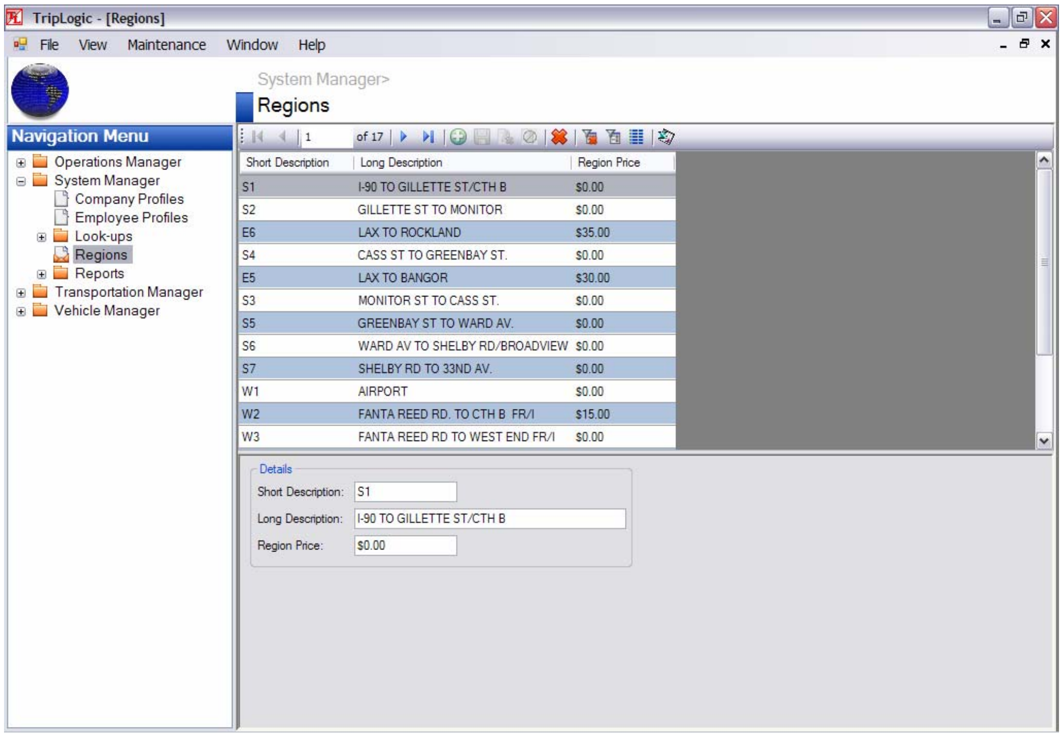
Short Description	Long Description	Years Valid	Warning (days)	Active
CPR	Cardiopulmonary Resuscitation	1	45	True
AP	Abuse Prevention	3	45	True

General
Short Description: CPR
Long Description: Cardiopulmonary Resuscitation

Details
Number of years valid: 1
Expire warning (days): 45

☑ Active

TripLogic Regions Window



Short Description	Long Description	Region Price
S1	I-90 TO GILLETTE ST/CTH B	$0.00
S2	GILLETTE ST TO MONITOR	$0.00
E6	LAX TO ROCKLAND	$35.00
S4	CASS ST TO GREENBAY ST.	$0.00
E5	LAX TO BANGOR	$30.00
S3	MONITOR ST TO CASS ST.	$0.00
S5	GREENBAY ST TO WARD AV.	$0.00
S6	WARD AV TO SHELBY RD/BROADVIEW	$0.00
S7	SHELBY RD TO 33ND AV.	$0.00
W1	AIRPORT	$0.00
W2	FANTA REED RD. TO CTH B FR/I	$15.00
W3	FANTA REED RD TO WEST END FR/I	$0.00

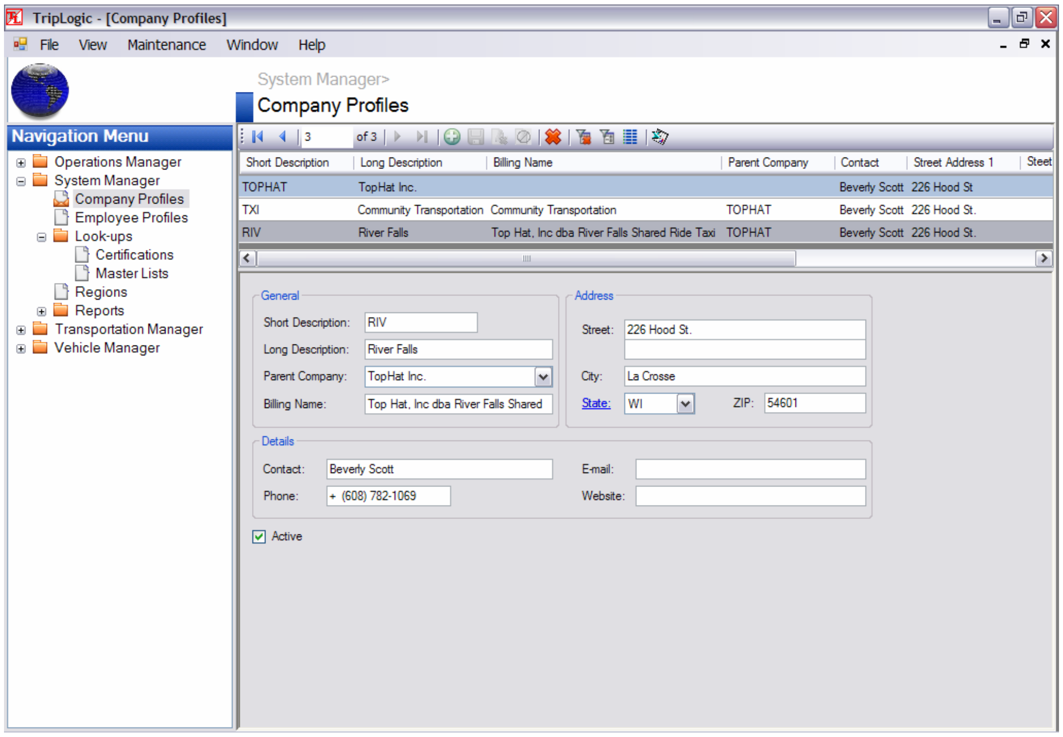
Details

Short Description: S1

Long Description: I-90 TO GILLETTE ST/CTH B

Region Price: $0.00

TripLogic Company Profiles Window



Short Description	Long Description	Billing Name	Parent Company	Contact	Street Address 1	Steet
TOPHAT	TopHat Inc.			Beverly Scott	226 Hood St	
TXI	Community Transportation	Community Transportation	TOPHAT	Beverly Scott	226 Hood St.	
RIV	River Falls	Top Hat, Inc dba River Falls Shared Ride Taxi	TOPHAT	Beverly Scott	226 Hood St.	

General

Short Description: RIV

Long Description: River Falls

Parent Company: TopHat Inc.

Billing Name: Top Hat, Inc dba River Falls Shared

Address

Street: 226 Hood St.

City: La Crosse

State: WI

ZIP: 54601

Details

Contact: Beverly Scott

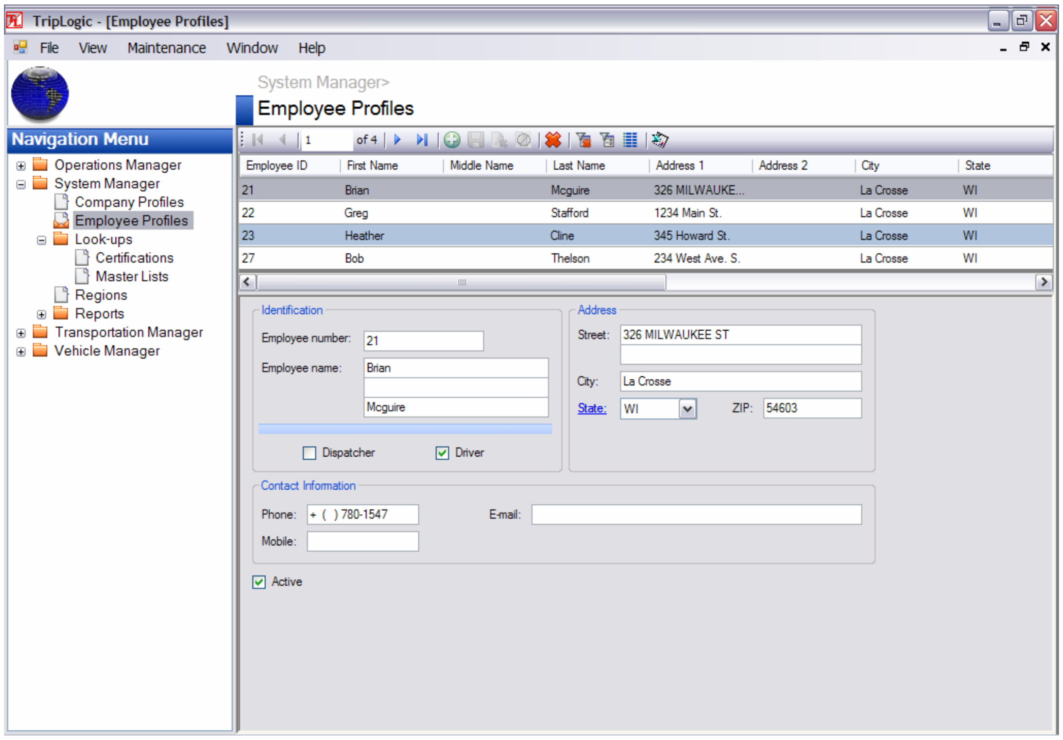
Phone: + (608) 782-1069

E-mail:

Website:

✓ Active

TripLogic Employee Profiles Window



Employee ID	First Name	Middle Name	Last Name	Address 1	Address 2	City	State
21	Brian		Mcguire	326 MILWAUKE...		La Crosse	WI
22	Greg		Stafford	1234 Main St.		La Crosse	WI
23	Heather		Cline	345 Howard St.		La Crosse	WI
27	Bob		Thelson	234 West Ave. S.		La Crosse	WI

Identification

Employee number: 21

Employee name: Brian

Mcguire

[ ] Dispatcher [✓] Driver

Address

Street: 326 MILWAUKEE ST

City: La Crosse

State: WI ZIP: 54603

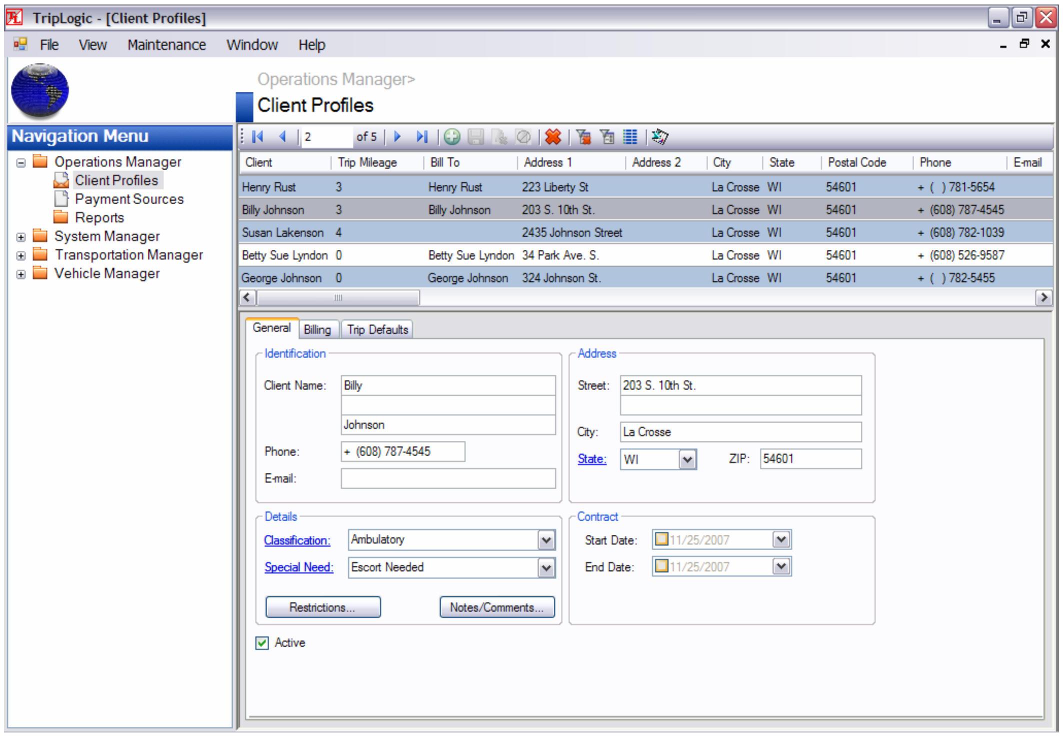
Contact Information

Phone: + ( ) 780-1547 E-mail:

Mobile:

[✓] Active

TripLogic Client Profile Window



Client	Trip Mileage	Bill To	Address 1	Address 2	City	State	Postal Code	Phone	E-mail
Henry Rust	3	Henry Rust	223 Liberty St		La Crosse	WI	54601	+ ( ) 781-5654	
Billy Johnson	3	Billy Johnson	203 S. 10th St.		La Crosse	WI	54601	+ (608) 787-4545	
Susan Lakenson	4		2435 Johnson Street		La Crosse	WI	54601	+ (608) 782-1039	
Betty Sue Lyndon	0	Betty Sue Lyndon	34 Park Ave. S.		La Crosse	WI	54601	+ (608) 526-9587	
George Johnson	0	George Johnson	324 Johnson St.		La Crosse	WI	54601	+ ( ) 782-5455	

Identification

Client Name: Billy

Johnson

Phone: + (608) 787-4545

E-mail:

Address

Street: 203 S. 10th St.

City: La Crosse

State: WI ZIP: 54601

Details

Classification: Ambulatory

Special Need: Escort Needed

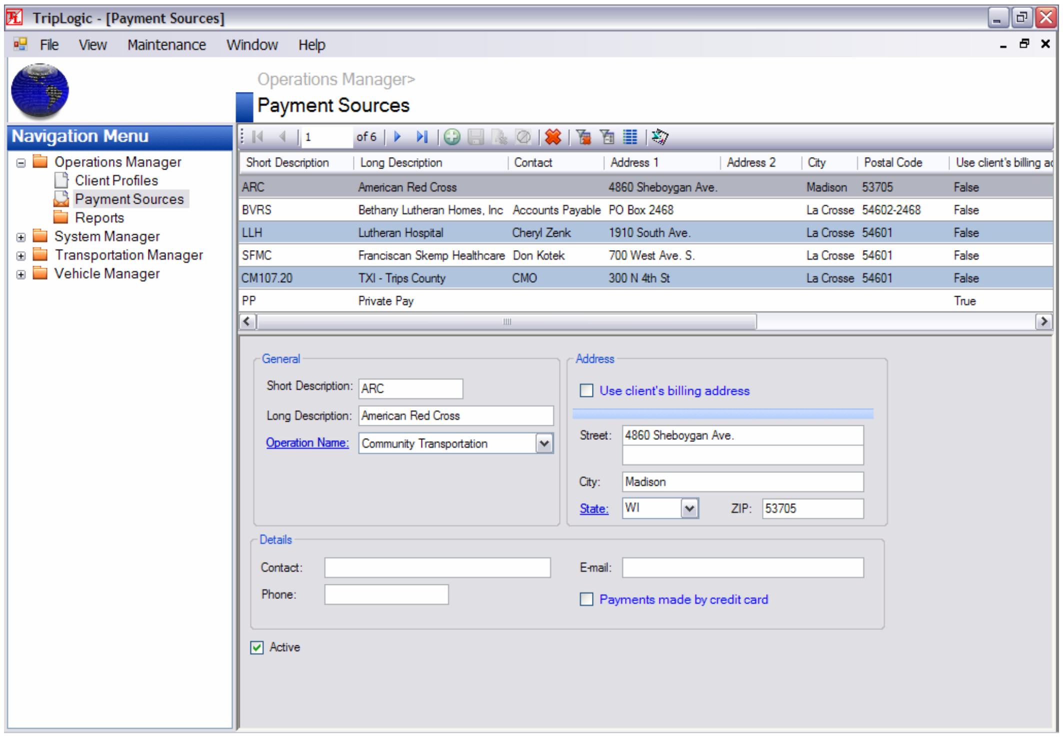
Contract

Start Date: 11/25/2007

End Date: 11/25/2007

✓ Active

TripLogic Payment Sources Window



Short Description	Long Description	Contact	Address 1	Address 2	City	Postal Code	Use client's billing ad
ARC	American Red Cross		4860 Sheboygan Ave.		Madison	53705	False
BVRS	Bethany Lutheran Homes, Inc	Accounts Payable	PO Box 2468		La Crosse	54602-2468	False
LLH	Lutheran Hospital	Cheryl Zenk	1910 South Ave.		La Crosse	54601	False
SFMC	Franciscan Skemp Healthcare	Don Kotek	700 West Ave. S.		La Crosse	54601	False
CM107.20	TXI - Trips County	CMO	300 N 4th St		La Crosse	54601	False
PP	Private Pay						True

General

Short Description: ARC

Long Description: American Red Cross

Operation Name: Community Transportation

Address

☐ Use client's billing address

Street: 4860 Sheboygan Ave.

City: Madison

State: WI ZIP: 53705

Details

Contact:

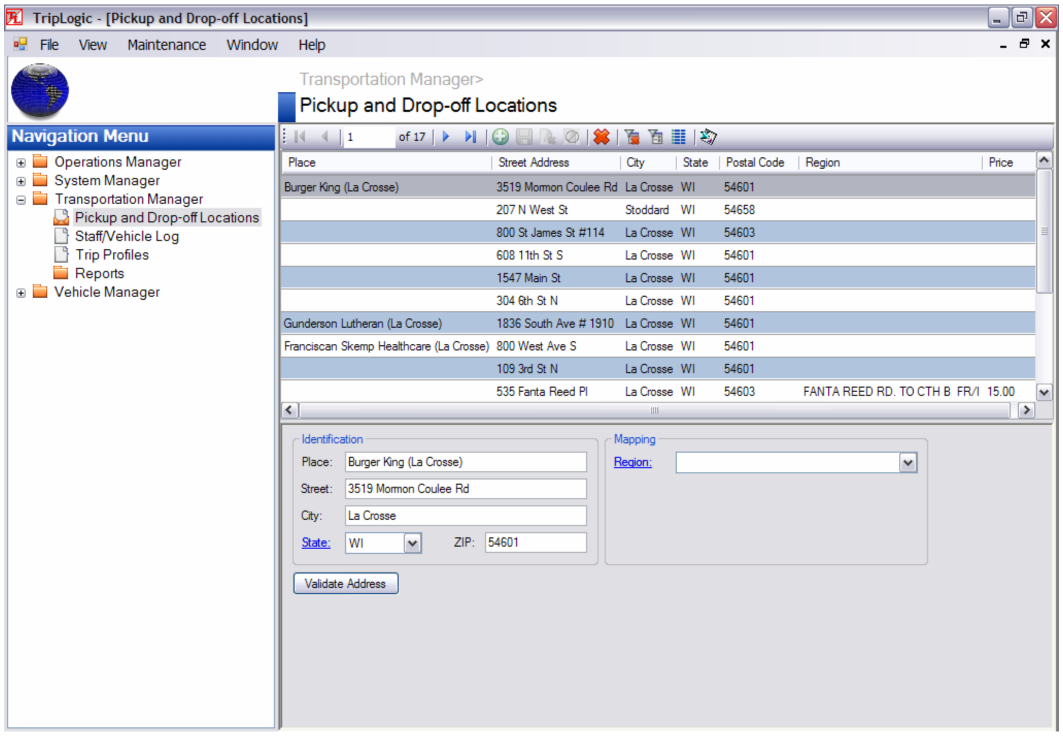
Phone:

E-mail:

☐ Payments made by credit card

☑ Active

TripLogic Pickup and Drop-off Locations Window



Place	Street Address	City	State	Postal Code	Region	Price
Burger King (La Crosse)	3519 Mormon Coulee Rd	La Crosse	WI	54601		
	207 N West St	Stoddard	WI	54658		
	800 St James St #114	La Crosse	WI	54603		
	608 11th St S	La Crosse	WI	54601		
	1547 Main St	La Crosse	WI	54601		
	304 6th St N	La Crosse	WI	54601		
Gunderson Lutheran (La Crosse)	1836 South Ave # 1910	La Crosse	WI	54601		
Franciscan Skemp Healthcare (La Crosse)	800 West Ave S	La Crosse	WI	54601		
	109 3rd St N	La Crosse	WI	54601		
	535 Fanta Reed Pl	La Crosse	WI	54603	FANTA REED RD. TO CTH B FR/I	15.00

Identification

Place: Burger King (La Crosse)

Street: 3519 Mormon Coulee Rd

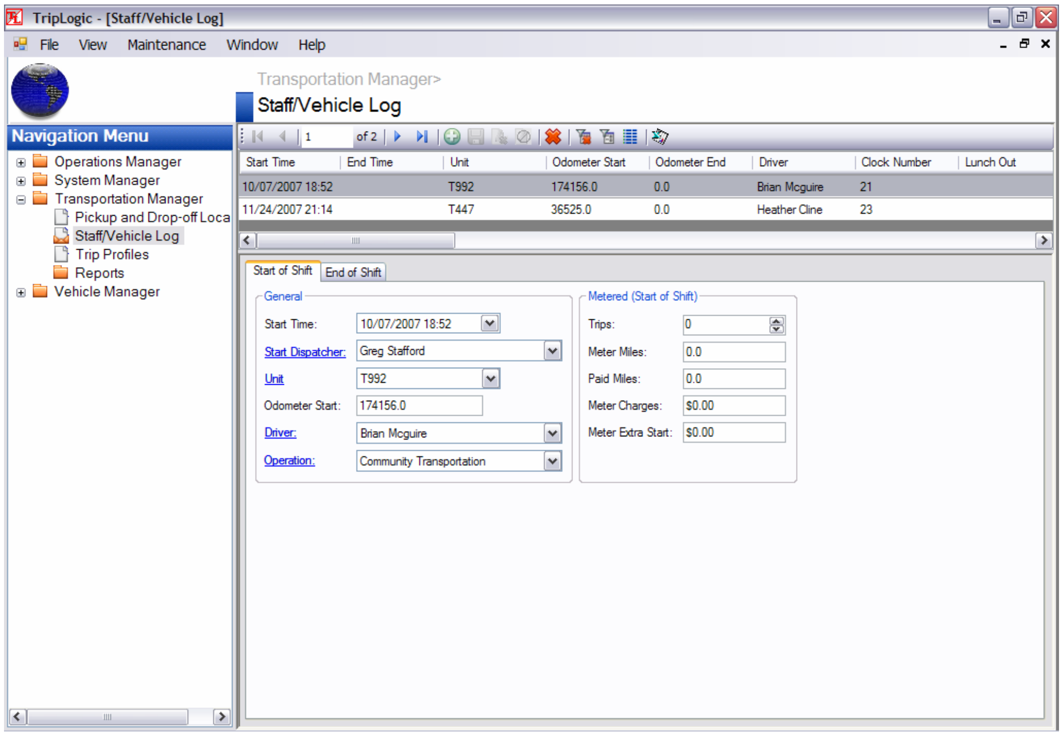
City: La Crosse

State: WI ZIP: 54601

Mapping

Region:

TripLogic Staff/Vehicle Log Window



Start Time	End Time	Unit	Odometer Start	Odometer End	Driver	Clock Number	Lunch Out
10/07/2007 18:52		T992	174156.0	0.0	Brian Mcguire	21	
11/24/2007 21:14		T447	36525.0	0.0	Heather Cline	23	

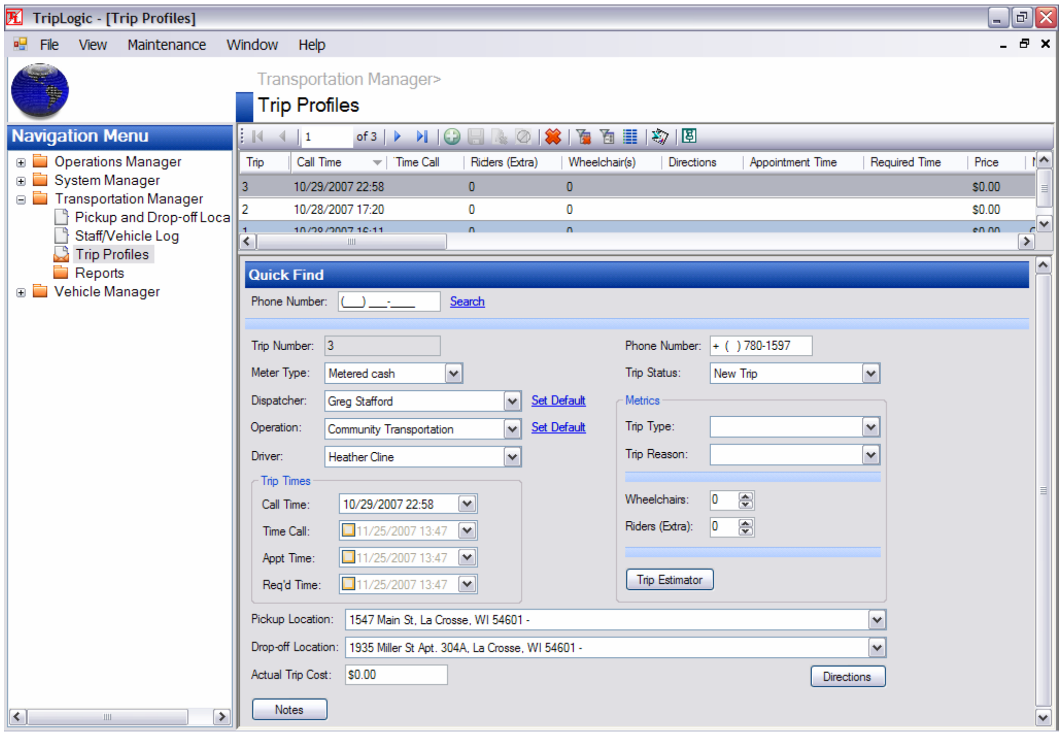
General

Start Time:	10/07/2007 18:52
Start Dispatcher:	Greg Stafford
Unit	T992
Odometer Start:	174156.0
Driver:	Brian Mcguire
Operation:	Community Transportation

Metered (Start of Shift)

Trips:	0
Meter Miles:	0.0
Paid Miles:	0.0
Meter Charges:	$0.00
Meter Extra Start:	$0.00

TripLogic Trip Profiles Window



Trip	Call Time	Time Call	Riders (Extra)	Wheelchair(s)	Directions	Appointment Time	Required Time	Price
3	10/29/2007 22:58		0	0				$0.00
2	10/28/2007 17:20		0	0				$0.00
1	10/28/2007 16:11		0	0				$0.00

Quick Find

Phone Number: (___) ___-____ Search

Trip Number:	3	Phone Number:	+ ( ) 780-1597
Meter Type:	Metered cash	Trip Status:	New Trip
Dispatcher:	Greg Stafford (Set Default)	Metrics	
Operation:	Community Transportation (Set Default)	Trip Type:	
Driver:	Heather Cline	Trip Reason:	

Trip Times

Call Time:	10/29/2007 22:58
Time Call:	11/25/2007 13:47
Appt Time:	11/25/2007 13:47
Req'd Time:	11/25/2007 13:47

Wheelchairs:	0
Riders (Extra):	0

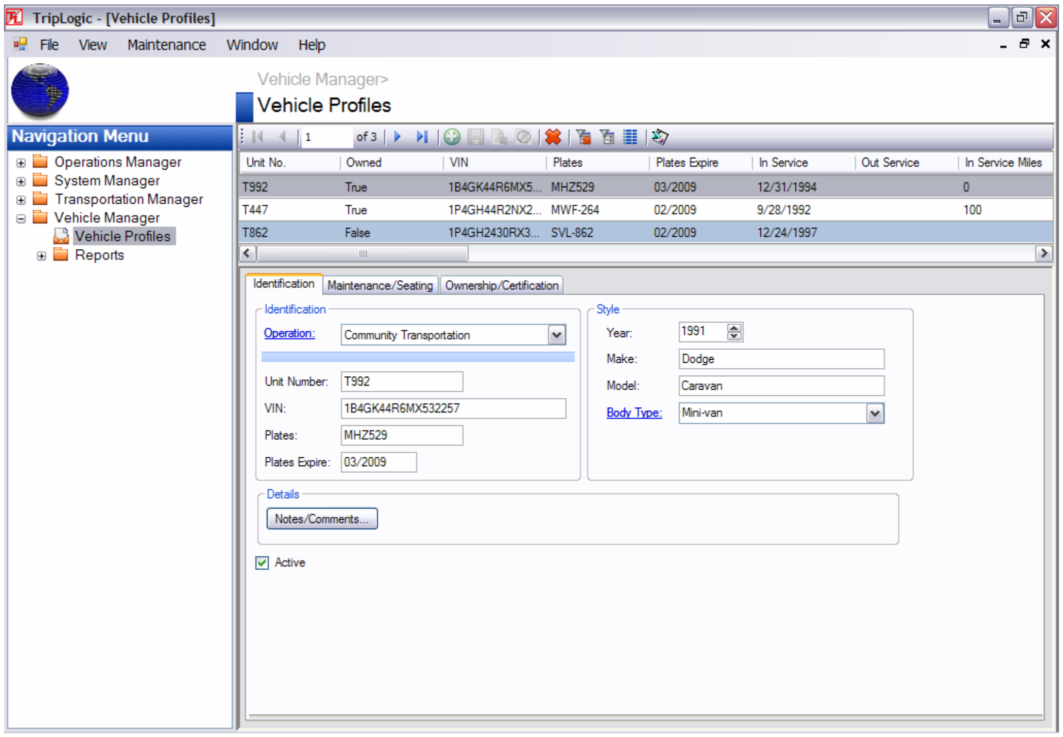
Trip Estimator

Pickup Location:	1547 Main St, La Crosse, WI 54601 -
Drop-off Location:	1935 Miller St Apt. 304A, La Crosse, WI 54601 -
Actual Trip Cost:	$0.00

Directions

Notes

TripLogic Vehicle Profiles Window



Unit No.	Owned	VIN	Plates	Plates Expire	In Service	Out Service	In Service Miles
T992	True	1B4GK44R6MX5...	MHZ529	03/2009	12/31/1994		0
T447	True	1P4GH44R2NX2...	MWF-264	02/2009	9/28/1992		100
T862	False	1P4GH2430RX3...	SVL-862	02/2009	12/24/1997		

Identification

Operation: Community Transportation

Unit Number: T992

VIN: 1B4GK44R6MX532257

Plates: MHZ529

Plates Expire: 03/2009

Style

Year: 1991

Make: Dodge

Model: Caravan

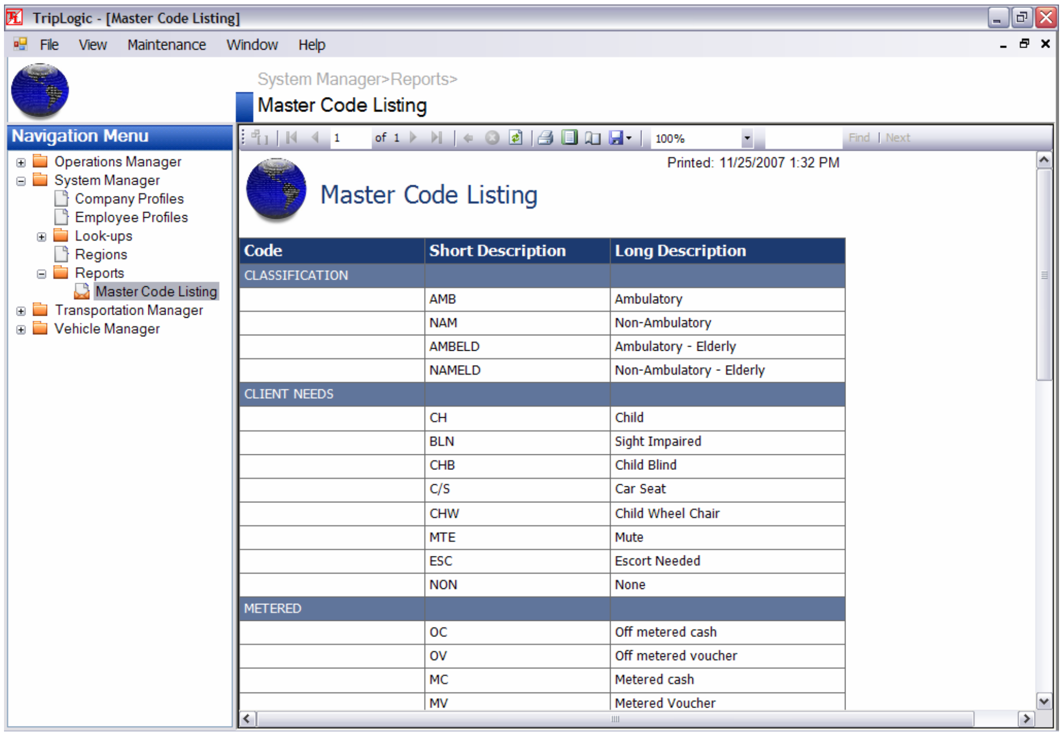
Body Type: Mini-van

Details

[✓] Active

Appendix B. Sample TripLogic Reports

TripLogic Master Code Listing Report

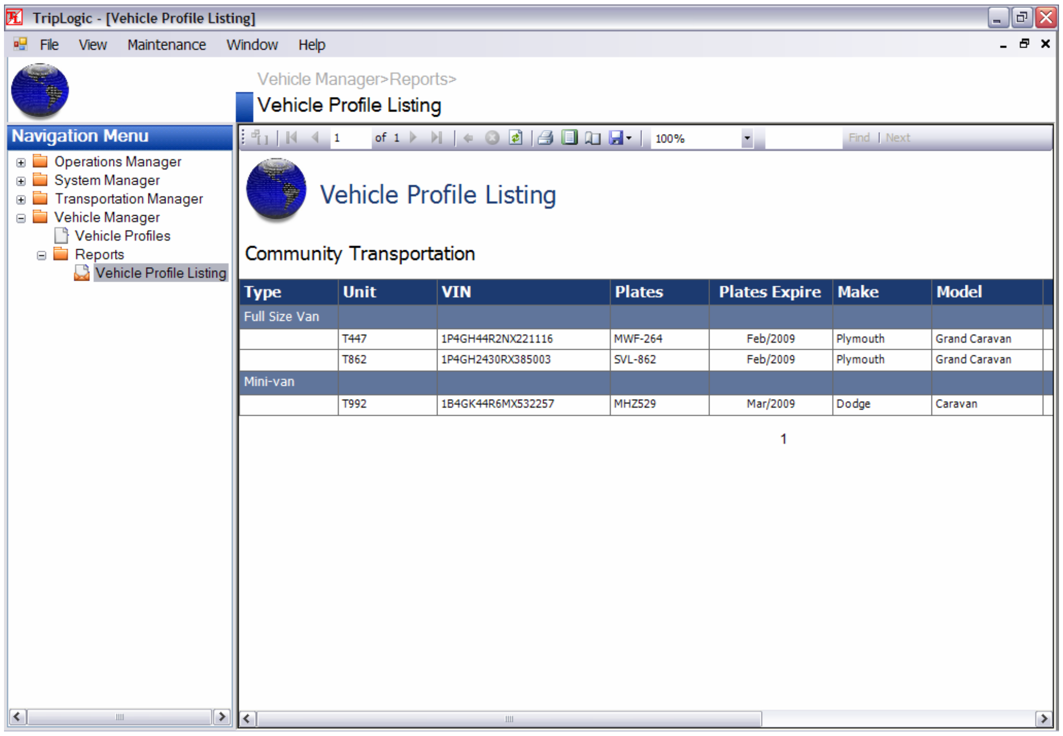


Printed: 11/25/2007 1:32 PM

Master Code Listing

Code	Short Description	Long Description
CLASSIFICATION		
	AMB	Ambulatory
	NAM	Non-Ambulatory
	AMBELD	Ambulatory - Elderly
	NAMELD	Non-Ambulatory - Elderly
CLIENT NEEDS		
	CH	Child
	BLN	Sight Impaired
	CHB	Child Blind
	C/S	Car Seat
	CHW	Child Wheel Chair
	MTE	Mute
	ESC	Escort Needed
	NON	None
METERED		
	OC	Off metered cash
	OV	Off metered voucher
	MC	Metered cash
	MV	Metered Voucher

TripLogic Vehicle Profile Report



Vehicle Profile Listing

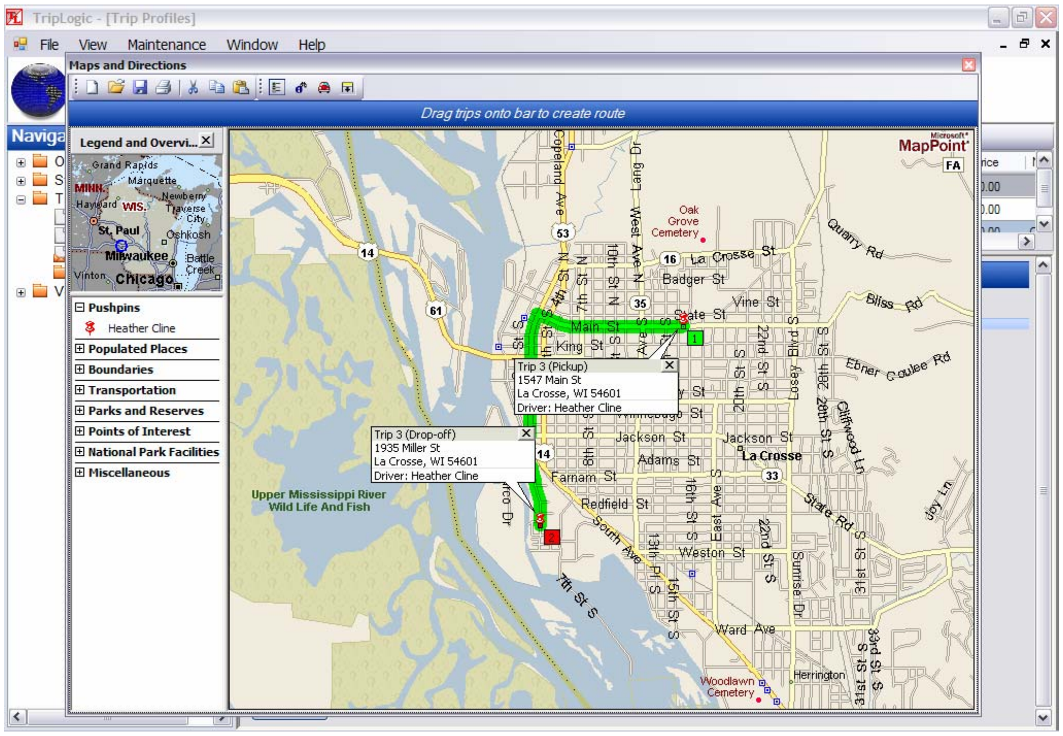
Community Transportation

Type	Unit	VIN	Plates	Plates Expire	Make	Model
Full Size Van						
	T447	1P4GH44R2NX221116	MWF-264	Feb/2009	Plymouth	Grand Caravan
	T862	1P4GH2430RX385003	SVL-862	Feb/2009	Plymouth	Grand Caravan
Mini-van						
	T992	1B4GK44R6MX532257	MHZ529	Mar/2009	Dodge	Caravan

1

Appendix C. Sample TripLogic Mapping Screen Shots

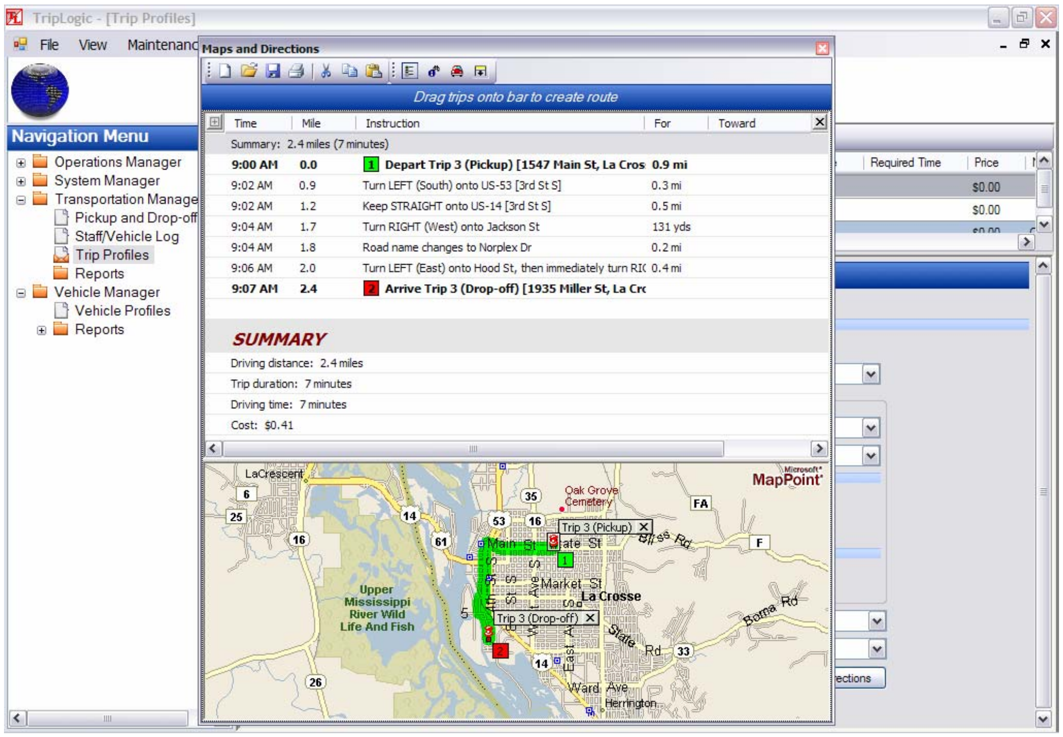
TripLogic Mapping and Directions With Legend Pane Window



Trip 3 (Pickup)
1547 Main St
La Crosse, WI 54601
Driver: Heather Cline

Trip 3 (Drop-off)
1935 Miller St
La Crosse, WI 54601
Driver: Heather Cline

TripLogic Mapping and Directions With Driving Directions Pane Window



Maps and Directions

Drag trips onto bar to create route

Time	Mile	Instruction	For	Toward
Summary: 2.4 miles (7 minutes)				
9:00 AM	**0.0**	🟩 **Depart Trip 3 (Pickup) [1547 Main St, La Cros**	**0.9 mi**	
9:02 AM	0.9	Turn LEFT (South) onto US-53 [3rd St S]	0.3 mi	
9:02 AM	1.2	Keep STRAIGHT onto US-14 [3rd St S]	0.5 mi	
9:04 AM	1.7	Turn RIGHT (West) onto Jackson St	131 yds	
9:04 AM	1.8	Road name changes to Norplex Dr	0.2 mi	
9:06 AM	2.0	Turn LEFT (East) onto Hood St, then immediately turn RI(	0.4 mi	
9:07 AM	**2.4**	🟥 **Arrive Trip 3 (Drop-off) [1935 Miller St, La Cr**		

SUMMARY

Driving distance: 2.4 miles

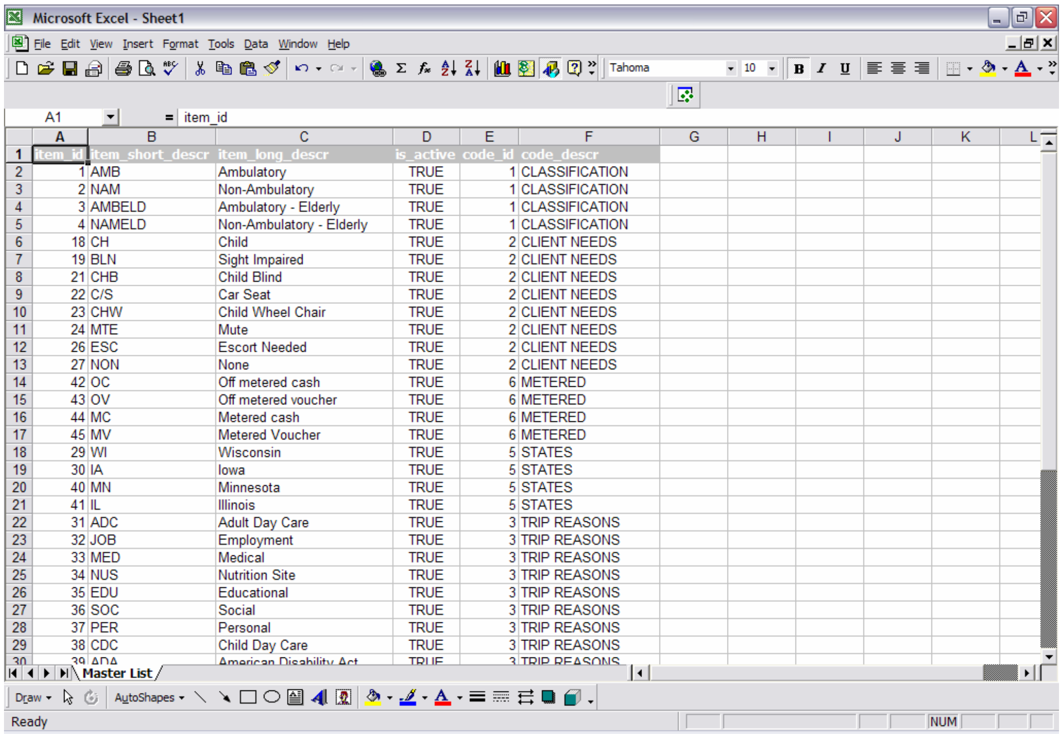
Trip duration: 7 minutes

Driving time: 7 minutes

Cost: $0.41

Appendix D. Sample TripLogic Data Export Screen Shot

TripLogic Master Code List Data Export to Microsoft Excel



item_id	item_short_descr	item_long_descr	is_active	code_id	code_descr
1	AMB	Ambulatory	TRUE	1	CLASSIFICATION
2	NAM	Non-Ambulatory	TRUE	1	CLASSIFICATION
3	AMBELD	Ambulatory - Elderly	TRUE	1	CLASSIFICATION
4	NAMELD	Non-Ambulatory - Elderly	TRUE	1	CLASSIFICATION
18	CH	Child	TRUE	2	CLIENT NEEDS
19	BLN	Sight Impaired	TRUE	2	CLIENT NEEDS
21	CHB	Child Blind	TRUE	2	CLIENT NEEDS
22	C/S	Car Seat	TRUE	2	CLIENT NEEDS
23	CHW	Child Wheel Chair	TRUE	2	CLIENT NEEDS
24	MTE	Mute	TRUE	2	CLIENT NEEDS
26	ESC	Escort Needed	TRUE	2	CLIENT NEEDS
27	NON	None	TRUE	2	CLIENT NEEDS
42	OC	Off metered cash	TRUE	6	METERED
43	OV	Off metered voucher	TRUE	6	METERED
44	MC	Metered cash	TRUE	6	METERED
45	MV	Metered Voucher	TRUE	6	METERED
29	WI	Wisconsin	TRUE	5	STATES
30	IA	Iowa	TRUE	5	STATES
40	MN	Minnesota	TRUE	5	STATES
41	IL	Illinois	TRUE	5	STATES
31	ADC	Adult Day Care	TRUE	3	TRIP REASONS
32	JOB	Employment	TRUE	3	TRIP REASONS
33	MED	Medical	TRUE	3	TRIP REASONS
34	NUS	Nutrition Site	TRUE	3	TRIP REASONS
35	EDU	Educational	TRUE	3	TRIP REASONS
36	SOC	Social	TRUE	3	TRIP REASONS
37	PER	Personal	TRUE	3	TRIP REASONS
38	CDC	Child Day Care	TRUE	3	TRIP REASONS
39	ADA	American Disability Act	TRUE	3	TRIP REASONS