Metro System Software Design

# Introduction

This document seeks to give an overview of the software design considered for the Metro System application. The purpose of the application is to help a city plan and design a metro system from the ground up. It will also provide tools for the user to track the current state of the system, real time tracking of all equipment in use, and statistical analysis tools for how the system is being utilized.

## Purpose

The document describes the software architecture chosen for the application. It will allow users, developers and product owners to get a high level overview of the current state of the project. It will lay out important design decisions, class diagrams, use case diagrams and process interactions. It will also help the development to plan out the details of the project ahead of time and have this reviewed with the entire staff responsible for the project. This document will adhere to the 4+1 view model as the approach to make it easy for each stakeholder to find the information they desire.

## Scope

The scope of the architecture document will provide high level and as well as lower level details of the system being used to drive the application. This will allow all levels of individuals involved with the project to get as detailed as they want in terms of what is their domain involvement in the project. The 4+1 view model will allow the system to be very accurately and detailed as a whole. This will also allow each section to grow and be maintained on its own.

## Definition, Acronyms, and Abbreviations

TBD

## Overview

The report will provide a detailed analysis of the software architecture of the Metro System application. It will begin by providing a very high overview of the product and get more detailed in later sections. This will allow each individual to go as deep as they want to in researching the details of the product.

# Architectural Representation

The section describes the views used to represent the architectural document

## Scenarios

This section describes all the scenarios of the system that the end user will encounter.

## Logical View

This section holds the requirements, object hierarchy and system layers. Describes the design of the object model and the relationships between the various subsystems.

## Process View

Non-functional requirements, describes the design’s concurrency and synchronization aspects. Elaborates the run time behavior.

## Development View

Software components, describes the modules and subsystem divisions of the system.

## Physical View

Persistence, describes how the system achieves persistence of the data model. Describes mapping of software to hardware and shows system’s aspects as well as the object relationships and database schemas used.

# Architecture Goals and Constraints

## Server Side

The server side will host the connection that will real time data to be fed to the application about the system’s current state. The server will store and host information such as real time tracking of metro locations, metro usage statistics, and more. The information will be stored in SqlLite and will be available to the application over WCF protocol.

## Client Side

The application will be hosted as a standalone executable on the client’s computer. The application will connect with a server to retrieve current information about the configured metro system. The client can modify the metro system with the information pushed to the server and saved on the server end. Metro system information can also be pulled from the server to display any information the user wants about a particular metro system.

## Security

User administration shall be used to allow only certain individuals to perform actions on the metro system’s configuration. The user roles will be defined to have specific privileges of access. The first user configured will be the admin which alone has the privileges to do everything.

## Persistence

Persistent storage will be done on the server end and will be stored as an SqlLite database.

## Reliability/Availability

The system shall be tested through unit testing, integration testing, automated tested and system testing.

## Performance

The system responds to any request under standard database and WCF service requests. It can also depend on hardware, network, and internet connection speeds. This connection speed can affect the real time tracking data feature the most. The statistical data generation may take some time to request.

## Portability and Reuse

The processes will all use reusable code and common framework code to make good use of code reuse.

## Development Tools

|  |  |
| --- | --- |
| **Programming** | **Visual Studio 2019** |
| **Database** | **SQLite** |
| **Diagrams** | **Draw.IO** |
| **Documentation** | **Microsoft Word** |
| **Tasking / Planning** | **Trello** |

## Software

|  |  |
| --- | --- |
| **UI** | **WPF / MVVM Light / Caliburn Micro** |
| **Communication** | **WCF** |
| **Database** | **SQLite, EF6, Dapper** |
| **Language** |  |
| **Application** | **C#** |
| **Server** | **C#** |
| **Persistence** |  |
| **Application Configuration** | **Database Storage** |
| **Application Settings** | **JSON Storage (NewtonSoft)** |
| **Configuration Management** | **Autofac** |

# Use Case View

# Logical View

# Process View

The system will be made up of several processes.

There will be two main applications, the metro system editor, which will allow for configuring a metro system, and the metro system observer, which will allow for observing a created metro system.

The servers will be made up of a ***configuration*** server to store the metro’s that are currently configured, a ***real-time tracking***server which is receiving real time information on the metro system’s state, and a ***statistical*** server which is storing usage information as it comes in from various sources.

The ***metro system editor*** will primarily connect to and communicate with the ***configuration*** server while the ***metro system observer*** application will connect to all servers.

# Implementation View

The overall engineering direction of the software is inspired by the **aspect-oriented programming** model, which seeks to keep cross-cutting concerns from interfering with and being a part of the flow of the main software.

The following web article explains the approach and why it is a powerful programming model to follow: <https://dzone.com/articles/aspect-oriented-programming-in-c-using-dispatchpro>. The main technique for achieving AOP in this application is using the ***decorator design pattern***.

Data and behavior will be separated for most design choices. This follows some good patterns found here: <https://blogs.cuttingedge.it/steven/posts/2011/meanwhile-on-the-command-side-of-my-architecture/> and here: <https://blogs.cuttingedge.it/steven/posts/2011/meanwhile-on-the-query-side-of-my-architecture/>

# Data View