SURTS

System for Uniform Route-based Transportation Simulation

Design Document

Authors:
Connor Boehm

Group: A41

This page intentionally left blank.

Document Revision History

Date	Version	Description	Author
10/25/2012	0.0	Initial draft	
11/23/2021	1.0	Used template to make individual design documentation for SURTS	Connor Boehm

This page intentionally left blank.

Contents

1 Introduction

This section addresses the purpose of this document including the intended audience, an introduction to the problem and a detailed view of the project's design. In the discussion, the design of the final system including several detailed diagrams will be described in detail.

1.1 Purpose

Provide a brief introduction to the document. What is the purpose of the document? What is its layout?

This document serves as a guide and resource for the implementation of the SURTS system, in particular the Business Layer Module.

1.2 System Overview

Provide a brief description of the purpose of the system, its target users and the environment in which the system will be used. This is basically from the initial part of the requirements document with revisions if any.

Develop a system that will simulate a uniform route-based transportation system. Because transportation systems handle many diverse populations in motion, developing transit routes that meet the needs of the users while keeping costs low is a challenge. This simulation will be used as a testing environment to determine optimal routes, fleet usage, and profitability.

1.3 Design Objectives

In this section discuss the overall system design goals. Broadly mention what functionality the design attempts to cover and what it doesn't. Also, refer to non-functional requirements like performance and usability that the design addresses. The requirements specification document could be useful in writing this section. Take a look at the functional and non-functional requirements definitions in your requirements specification.

This section should provide the overall scope and context for the design. After reading this section the reader should be able to understand *what* features and functions are being addressed. In the later sections, you will be explaining *how* this is achieved in your design.

The system hope to

1.4 References

Provide references to other documents.

SURTS Software Requirements Specification v2.0, *Boehm, et al*

1.5 Definitions, Acronyms, and Abbreviations

Definition of various terms, acronyms, or abbreviations used in the document.

Note: If your TAs ask, "what do you mean by this…?", may be you should consider adding a brief definition here.

If some of the terms have been defined in the requirements document already, you may just refer the reader to the glossary of the requirements document.

2 Design Overview

2.1 Introduction

This section should highlight the design approach that you adopt (e.g. object-oriented design or structured design), the proposed architecture of the system (e.g. client-server) and the relevant techniques and tools used (e.g., Microsoft Visio, Dia, Argo8, etc.).

I followed the approach of primarily using the observer, abstract factory, and façade design patterns. The façade was necessary for providing an interface, the factories will allow future expansion of any other transportation mode or feature, and the observers allow the log to collect data efficiently.

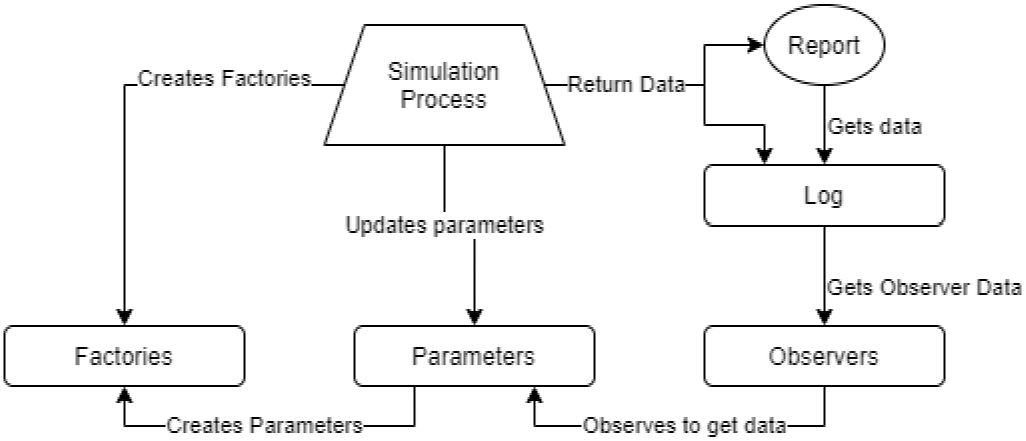
2.2 Environment Overview

This section should describe the environment in which the system will be run. This should include diagrams and descriptions of the environment. For example, if we are describing a web app, this will include a network diagram or topology. If we are developing a library, we should describe what might be calling us. In order to do this effectively, you should have previously stated your system's scope.

Be sure to address where this application will reside and how it will be executed.

2.3 System Architecture

Provide a high-level description of the system architecture. Use a few block diagrams to show the major components and their interaction. Remember to describe the conventions and notations used in your diagrams.



This a broad view of the system structure in the BML. As seen, the simulation process is the interface/controller of the system. It will create factories to make the parameters. Simulation Process will have a virtual update process that will update the parameters during the simulation, and a configure option to change the parameters. The logs and reports will give data to the interface for other subsystems to use (gui,etc) and gets that data from what is relayed through the observers in parameters.

2.4 Constraints and Assumptions

Mention the major design constraints here. These may have been imposed by the customer, which can be found in the requirements document. Explain how your design accommodates these constraints.

There may also be constraints imposed because of your system interacting with other external systems or being dependent on some external systems to provide part of the functionality. In such cases clearly mention the type of software your system interacts with (e.g. XYZ database software, ABC email software) and the constraints this imposes (e.g. only text based email messages are allowed).

Implementation languages and platforms may impose certain constraints. Mention them here, including the languages or technologies you will use in your implementation.

For constraints imposed by your design choices, briefly mention the options that you had, the tradeoffs involved and the reason why you made the choice.

None as of 11/23/2020 (v1.0)

3 Interfaces and Data Stores

This section describes the interfaces into and out of the system as well as the data stores you will be including in your system.

3.1 System Interfaces

The various interfaces provided to users and/or other external systems should be defined here. If you had included user interface descriptions in your requirements document you may refer to them here. If you provide interfaces to other systems, say export and import data to a different software, you should mention them here.

Do not include JavaDocs here. You should describe the interfaces using prose and graphics if your interface includes a GUI.

MainInterface class is the façade interface that other subsystems such as the GUI will use.

3.2 Data Stores

Describe the data stores you will create AS PART OF YOUR SYSTEM. This should NOT include databases or other data stores that are external to your system. This information should be included in a previous section.

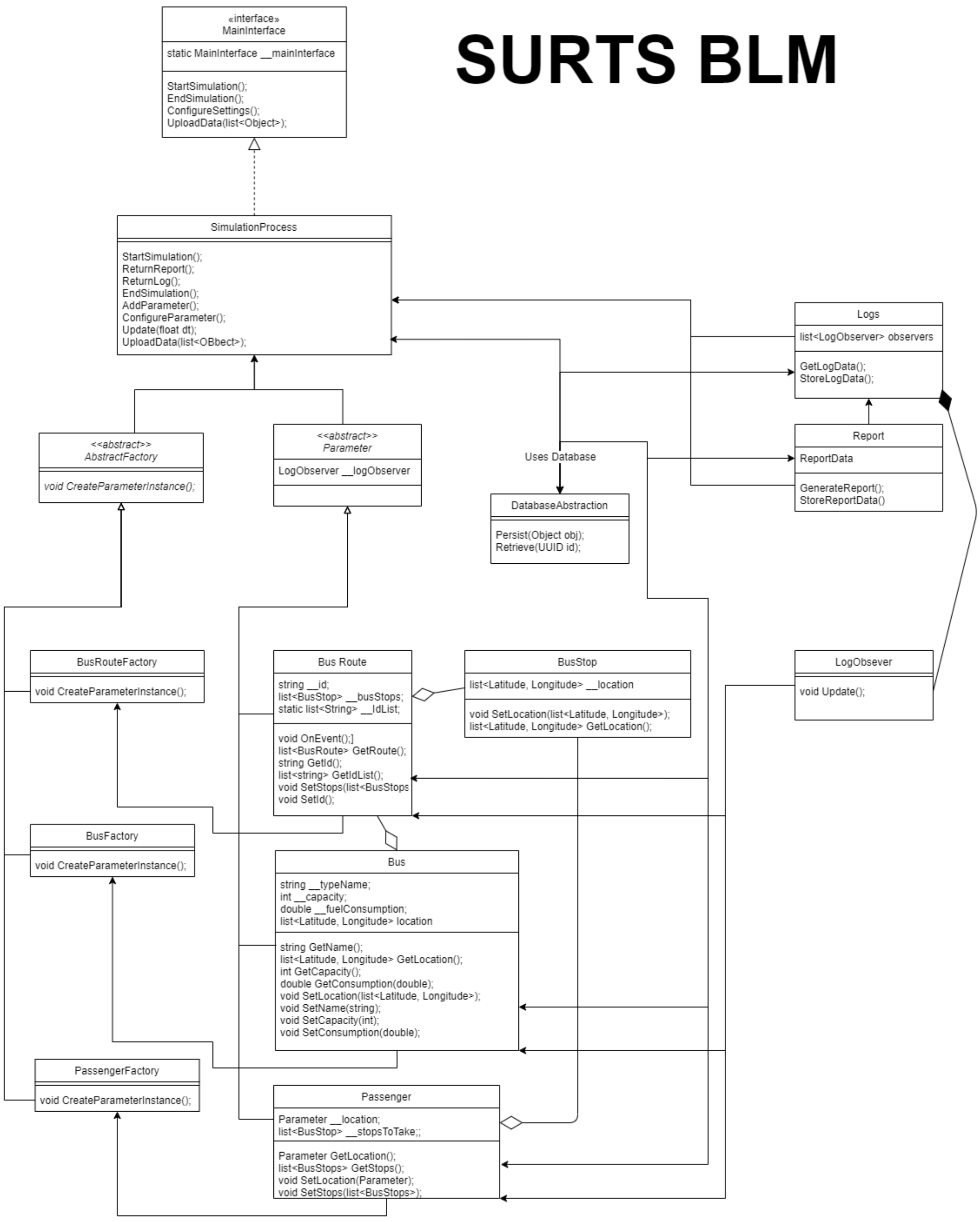
4 Structural Design

4.1 Class Diagram

Provide the class diagram for the system. If the class diagram is too big, partition the diagram using some reasonable criteria. For example, you may provide the client-side and the server-side object models as separate diagrams.

What should go into a class diagram?

All the classes should find a place here. All associations between classes should be identified and associations should be decorated with the right cardinality. Aggregation and inheritance relationships should be identified. ***A brief explanation should accompany each diagram including your justification for your design.*** *Make sure you include figure numbers to help you reference figures in the document.*


SURTS BLM
«interface»
MainInterface
static MainInterface __mainInterface
StartSimulation();
EndSimulation();
ConfigureSettings();
UploadData(list<Object>);
SimulationProcess
StartSimulation();
ReturnReport();
ReturnLog();
EndSimulation();
AddParameter();
ConfigureParameter();
Update(float dt);
UploadData(list<OBbect>);
Logs
list<LogObserver> observers
GetLogData();
StoreLogData();
Report
ReportData
GenerateReport();
StoreReportData()
<<abstract>>
AbstractFactory
void CreateParameterInstance();
<<abstract>>
Parameter
LogObserver __logObserver
Uses Database
DatabaseAbstraction
Persist(Object obj);
Retrieve(UUID id);
BusRouteFactory
void CreateParameterInstance();
Bus Route
string __id;
list<BusStop> __busStops;
static list<String> __IdList;
void OnEvent();]
list<BusRoute> GetRoute();
string GetId();
list<string> GetIdList();
void SetStops(list<BusStops
void SetId();
BusStop
list<Latitude, Longitude> __location
void SetLocation(list<Latitude, Longitude>);
list<Latitude, Longitude> GetLocation();
LogObsever
void Update();
BusFactory
void CreateParameterInstance();
Bus
string __typeName;
int __capacity;
double __fuelConsumption;
list<Latitude, Longitude> location
string GetName();
list<Latitude, Longitude> GetLocation();
int GetCapacity();
double GetConsumption(double);
void SetLocation(list<Latitude, Longitude>);
void SetName(string);
void SetCapacity(int);
void SetConsumption(double);
PassengerFactory
void CreateParameterInstance();
Passenger
Parameter __location;
list<BusStop> __stopsToTake;;
Parameter GetLocation();
list<BusStops> GetStops();
void SetLocation(Parameter);
void SetStops(list<BusStops>);

4.2 Class Descriptions

In this section you would describe in detail each class, its attributes, and its methods. You should logically group classes together. For example, you may use your architecture diagrams to group classes within a sub-system together.

Provide a subsection for each class. For each class, briefly describe its purpose, any constraints, (e.g., only single instance) and list the attributes and the methods of each class in the class diagram.

For each class, describe each of its attributes with the following details: name, type, a one line description of the attribute if its meaning is not intuitive, and constraints on the attribute (e.g., attribute must have unique value for each object or value range is restricted to positive integers).

Each method should be described with the following details: method name, return type and value, parameters, purpose and a brief description of the algorithm used (if it is non-trivial). Pre-conditions and post-conditions should be mentioned here if there are any assumptions about the arguments or the return values. List the attributes read and modified by this method and other methods invoked by this method.

The following is provided as an example:

4.3 Classes in the Fuel Subsystem

TODO: Team Assignment. My UML has around 60-ish class functions

5 Dynamic Model

The purpose of this section is to model how the system responds to various events, i.e., model the system's behavior. We do this using UML sequence diagrams.

The first step is to identify different scenarios (e.g. Fuel Level Overshoots), making sure you address each use case in your requirements document. Do not invent scenarios, rather a general guideline is to include scenarios that would make sense to the customer. For example, for the course enrollment system, logging in is a valid scenario.

TODO: Team Assignment

5.1 Scenarios

For each scenario you will have a subsection with the following information:

- *Scenario Name: Provide some meaningful name for the scenario (it is a good idea to include this in the subsection name.*
- *Scenario Description: A brief description of what the scenario is about and the sequence of actions that take place*
- *Sequence Diagram: A sequence diagram showing various events and their relative time ordering (as discussed in class).*

6 Non-functional requirements

In this section you have to explain how you expect to meet the non-functional requirements specified in the requirements document. As far as possible, assess your design objectively for each non-functional requirement specified in the requirement document and explain how you meet the requirement. If some non-functional requirements are not totally under the control of the system you are designing, that should be mentioned here. You should also mention how you expect the system to evolve in future and how your design would accommodate those anticipated changes.

7 Supplementary Documentation

Provide any other relevant documentation that may help understanding the design.