Annex B: Prototype Draft Model Code

April 30th, 2015
Siamak Khaledi, Ankit Shah, Matthew Shoaf

1. Code Set: Prototype FTLADS Draft Model

1.1. Notes on Building the Wheel Draft Algorithm

- Based on the data analysis of the DC jurisdiction, there were 3 popular streets identified.
- The lowest capacity on the above 3 streets was 12.
- Draft pick numbers 1 to 12 were considered equally valued prime numbers for the draft.
- 252 truck numbers were divided into 21 groups of 12.
- The groups were arranged based on the constraints of the problem.
    - No consecutive working days.
    - Difference between a truck's number of working days in a week compared to any other truck should be no more than 1.
    - Prime pick numbers to be distributed across all 5 days of the week for each truck vendor.
    - Groups (1-12) and (13-24) are placed at 10 number of groups interval so that a truck vendor has either of the 1st or 2nd pick every 10 days.
- Numbers in each of the groups are randomly ranked from highest to lowest.
- Numbers are picked in the order of highest rank from each group to the lowest without repetition. For example, if Group (1-12) had 1 as the highest rank number, followed by Group (145-156) with 145 as the highest and so on, the algorithm will pick "1-145 - … " and after picking from the 21st group, it will pick the next highest rank number from all these groups. This will go on until all numbers from each of the groups are exhausted. The sequence of numbers that would be formed constitutes a "Wheel". A cycle of 252 numbers as days is now ready to be used.
- Save the sequence of numbers in the database.

1.2 Executable for Constructing the Wheel Draft Order

Below is the executable code from ASP.NET that is used to create the "Wheel" draft pick order:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class SequenceGenerator
{

    public static string GenerateSequence()
    {
        string result;
        result = "";

        Random rnd = new Random();
        List<int> group1 = new List<int>(new int[] { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 });
```

```
List<int> group2 = new List<int>(new int[] { 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156 });
List<int> group3 = new List<int>(new int[] { 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108 });
List<int> group4 = new List<int>(new int[] { 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36 });
List<int> group5 = new List<int>(new int[] { 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192 });
List<int> group6 = new List<int>(new int[] { 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48 });
List<int> group7 = new List<int>(new int[] { 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144 });
List<int> group8 = new List<int>(new int[] { 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204 });
List<int> group9 = new List<int>(new int[] { 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60 });
List<int> group10 = new List<int>(new int[] { 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228 });
List<int> group11 = new List<int>(new int[] { 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240 });
List<int> group12 = new List<int>(new int[] { 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24 });
List<int> group13 = new List<int>(new int[] { 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120 });
List<int> group14 = new List<int>(new int[] { 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252 });
List<int> group15 = new List<int>(new int[] { 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96 });
List<int> group16 = new List<int>(new int[] { 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168 });
List<int> group17 = new List<int>(new int[] { 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84 });
List<int> group18= new List<int>(new int[] { 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132 });
List<int> group19 = new List<int>(new int[] { 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180 });
List<int> group20 = new List<int>(new int[] { 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72 });
List<int> group21 = new List<int>(new int[] { 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216 });

int i = 1;
while (i < 13)
{
   var index = rnd.Next(0, group1.Count);
   if (i == 1)
   {
      result = group1[index].ToString();
   }
   else
   {
      result = result + "," + group1[index].ToString();
   }
   group1.RemoveAt(index);

   index = rnd.Next(0, group2.Count);
   result = result + "," + group2[index].ToString();
   group2.RemoveAt(index);

   index = rnd.Next(0, group3.Count);
   result = result + "," + group3[index].ToString();
   group3.RemoveAt(index);

   index = rnd.Next(0, group4.Count);
   result = result + "," + group4[index].ToString();
   group4.RemoveAt(index);


   index = rnd.Next(0, group5.Count);
   result = result + "," + group5[index].ToString();
   group5.RemoveAt(index);

   index = rnd.Next(0, group6.Count);
   result = result + "," + group6[index].ToString();
   group6.RemoveAt(index);

   index = rnd.Next(0, group7.Count);
   result = result + "," + group7[index].ToString();
   group7.RemoveAt(index);

   index = rnd.Next(0, group8.Count);
   result = result + "," + group8[index].ToString();
   group8.RemoveAt(index);

   index = rnd.Next(0, group9.Count);
   result = result + "," + group9[index].ToString();
   group9.RemoveAt(index);

   index = rnd.Next(0, group10.Count);
   result = result + "," + group10[index].ToString();
```

```
            group10.RemoveAt(index);

            index = rnd.Next(0, group11.Count);
            result = result + "," + group11[index].ToString();
            group11.RemoveAt(index);

            index = rnd.Next(0, group12.Count);
            result = result + "," + group12[index].ToString();
            group12.RemoveAt(index);

            index = rnd.Next(0, group13.Count);
            result = result + "," + group13[index].ToString();
            group13.RemoveAt(index);

            index = rnd.Next(0, group14.Count);
            result = result + "," + group14[index].ToString();
            group14.RemoveAt(index);

            index = rnd.Next(0, group15.Count);
            result = result + "," + group15[index].ToString();
            group15.RemoveAt(index);

            index = rnd.Next(0, group16.Count);
            result = result + "," + group16[index].ToString();
            group16.RemoveAt(index);

            index = rnd.Next(0, group17.Count);
            result = result + "," + group17[index].ToString();
            group17.RemoveAt(index);

            index = rnd.Next(0, group18.Count);
            result = result + "," + group18[index].ToString();
            group18.RemoveAt(index);

            index = rnd.Next(0, group19.Count);
            result = result + "," + group19[index].ToString();
            group19.RemoveAt(index);

            index = rnd.Next(0, group20.Count);
            result = result + "," + group20[index].ToString();
            group20.RemoveAt(index);

            index = rnd.Next(0, group21.Count);
            result = result + "," + group21[index].ToString();
            group21.RemoveAt(index);


            i++;

        }
        return result;

    }

}
```

1.3 Creation of the First Day of the Wheel Algorithm

- Request the sequence of numbers from the database.
- Randomly rank all truck vendor IDs in the system.
- Assign the corresponding wheel draft pick numbers to the truck vendor IDs to create the first day of the wheel.
- Trucks rotate clockwise on the wheel every day, meaning they will receive the draft number belonging to the truck immediately after them on the following day.
    - For example, if Truck 1 had a draft pick number 1, Truck 2 on its immediate right on the wheel had a draft pick number 145, Truck 3 to the immediate right of Truck 2 had a draft pick number 97, then on the following day, Truck 1 will have a draft pick number 145, Truck 2 will have a draft pick number 97 and Truck 3 will have a draft pick number of the truck that was on its immediate right the day before.

1.4 Executable for Creating the First Day of the Wheel Draft Order

Below is the executable code from ASP.NET that is used to create the first day of the Wheel draft pick order:

```
public static List<PickSet> CreateFirstDayofWheel()
  {

    List<PickSet> pickDayOne;

    //Step 1 Check if there is already an active day one
    DataLayer objDataLayer = new DataLayer();

    int hasDayOne = objDataLayer.hasDayOne();

    if (hasDayOne == 0)
    {
      //Step 1- get sequence for generating draft picks
      DataSet PickRange = objDataLayer.getPickRange();

      //Step 2- pass sequence to random number generator that returns list of pickRange for Day 1
      pickDayOne = genRandomPick(PickRange);

      // Step 3 - Save day one pick data to PickOrder Table
      objDataLayer.SaveDayOne(pickDayOne);

      objDataLayer = null;

      return pickDayOne;

    }
    else
    {
        objDataLayer = null;
        return null;
    }

  }
```

1.5 Assignment Algorithm

- Pre-Condition: The truck vendors have set their preferences for the streets.
- After the cut-off date to have truck vendors input / change their preferences, the administrator can run this algorithm.
- The algorithm is going to request the street information (identifier, name, capacity) for each of the streets from the database.
- The algorithm is going to request the street preferences per truck vendor from the database.
- The algorithm is then going to get the truck picking order for each of the days in the duration of the iteration (5 days of the week) from the database.
- The algorithm then starts assigning the parking spots based on the truck vendor's preferences in the order of the draft pick numbers for that day.

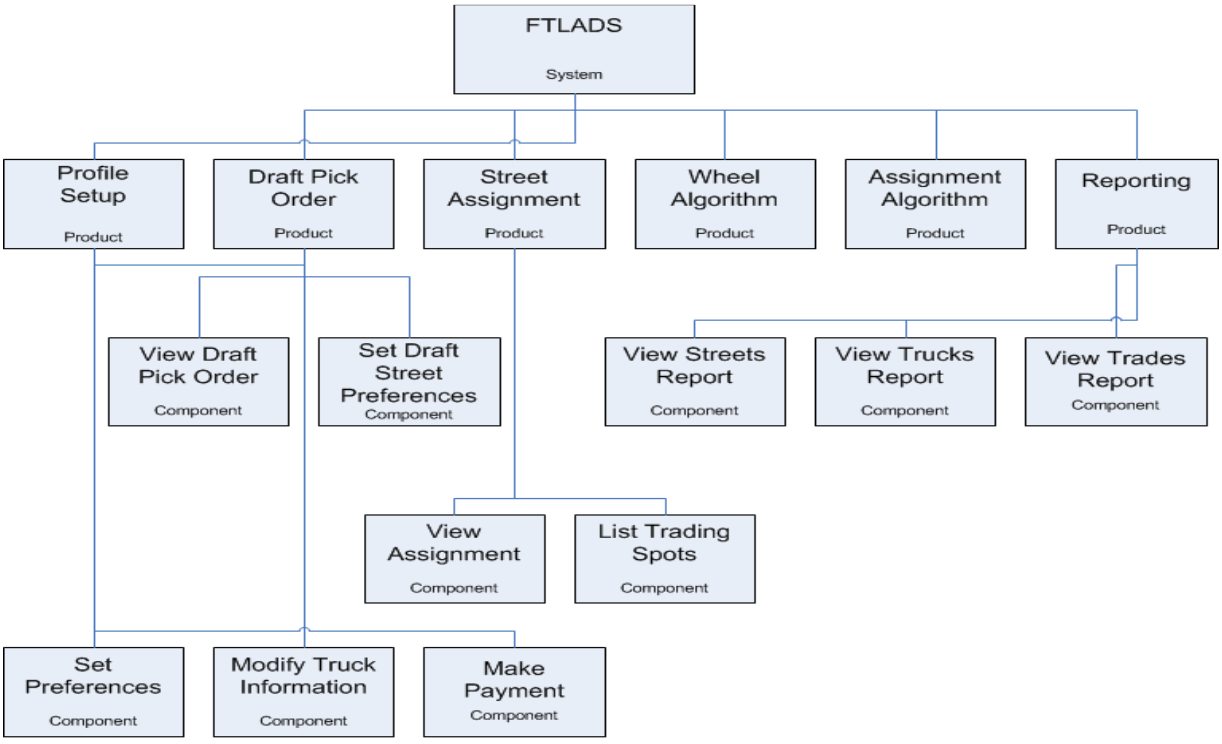
1.6 Executable for the Assignment Algorithm

Below is the executable code from ASP.NET that is used to create the assignment algorithm:

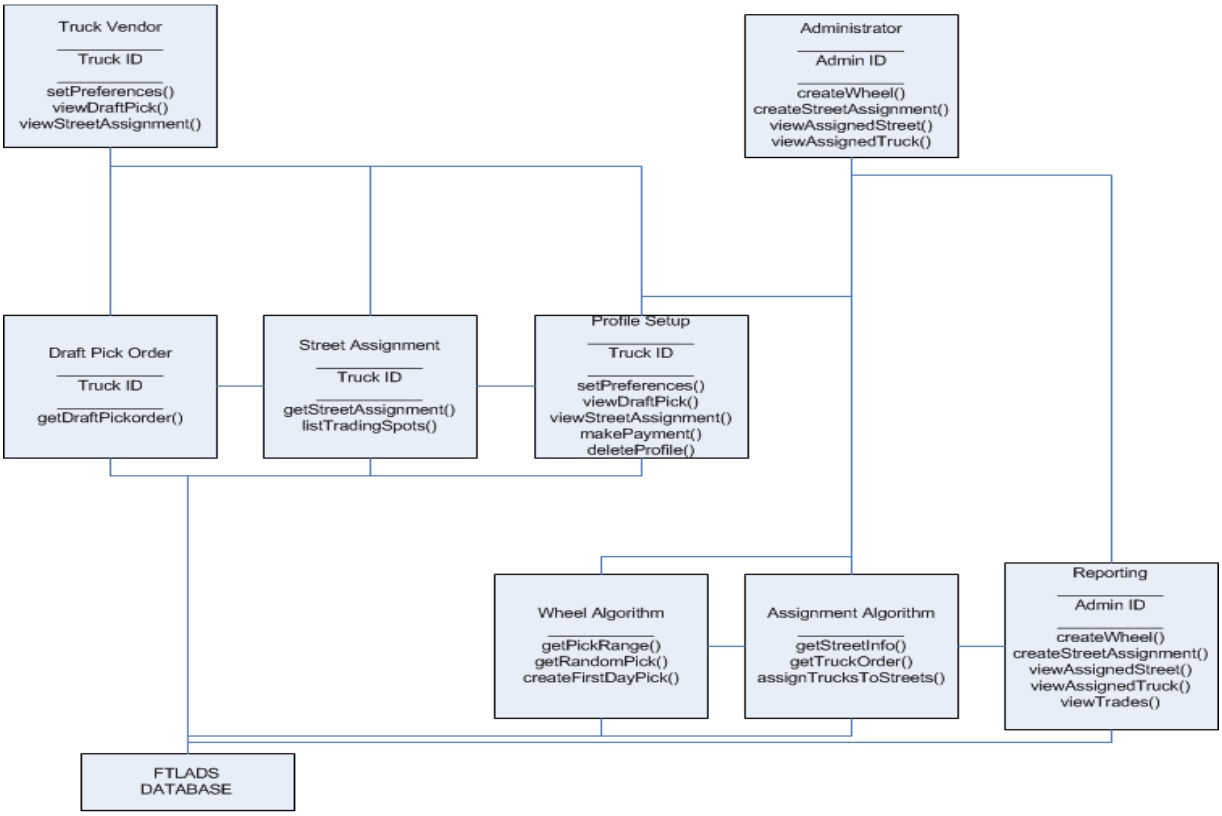
```
// Assigning streets
   private void assign(List<Street> listStreet)
   {
      DataSet PickOrderForDay;
      DataSet TruckStreetPref;
      DataLayer objDataLayer = new DataLayer();

      // Get preference data per day
      for (int i = 1; i <= 252; i++)
      {
                        // Get data for day i preferences
                         PickOrderForDay = objDataLayer.getPickOrderForDay(i);

         // Initialize used Count to 0 for each iteration of loop
         for (int t = 0; t < listStreet.Count; t++)
         {
            listStreet[t].UsedSpots = 0;
         }
         // loop over trucks in order of pick order numbers
         if (PickOrderForDay != null)
         {
            foreach (DataRow row in PickOrderForDay.Tables[0].Rows)
            {
               // get street preference for truck
               TruckStreetPref = objDataLayer.getTruckStreetPref(truckid);

               foreach (DataRow rowPref in TruckStreetPref.Tables[0].Rows)
               {
                  int streetPref = Convert.ToInt32(rowPref["TruckStreetPref_Street_fk"].ToString());

                  // find index in list of street object for this street
```

```
int lstIndex = listStreet.FindIndex(r => r.StreetId == streetPref);

if (!listStreet[lstIndex].IsFull)
{
  // add this street to preference list
    objDataLayer.addTruckStreetAssign(truckid,streetPref,dayNum,weekNum);

  // increment counter for used spots
    listStreet[lstIndex].UsedSpots = listStreet[lstIndex].UsedSpots + 1;

  // exit from loop
    break;
}}}}}
```

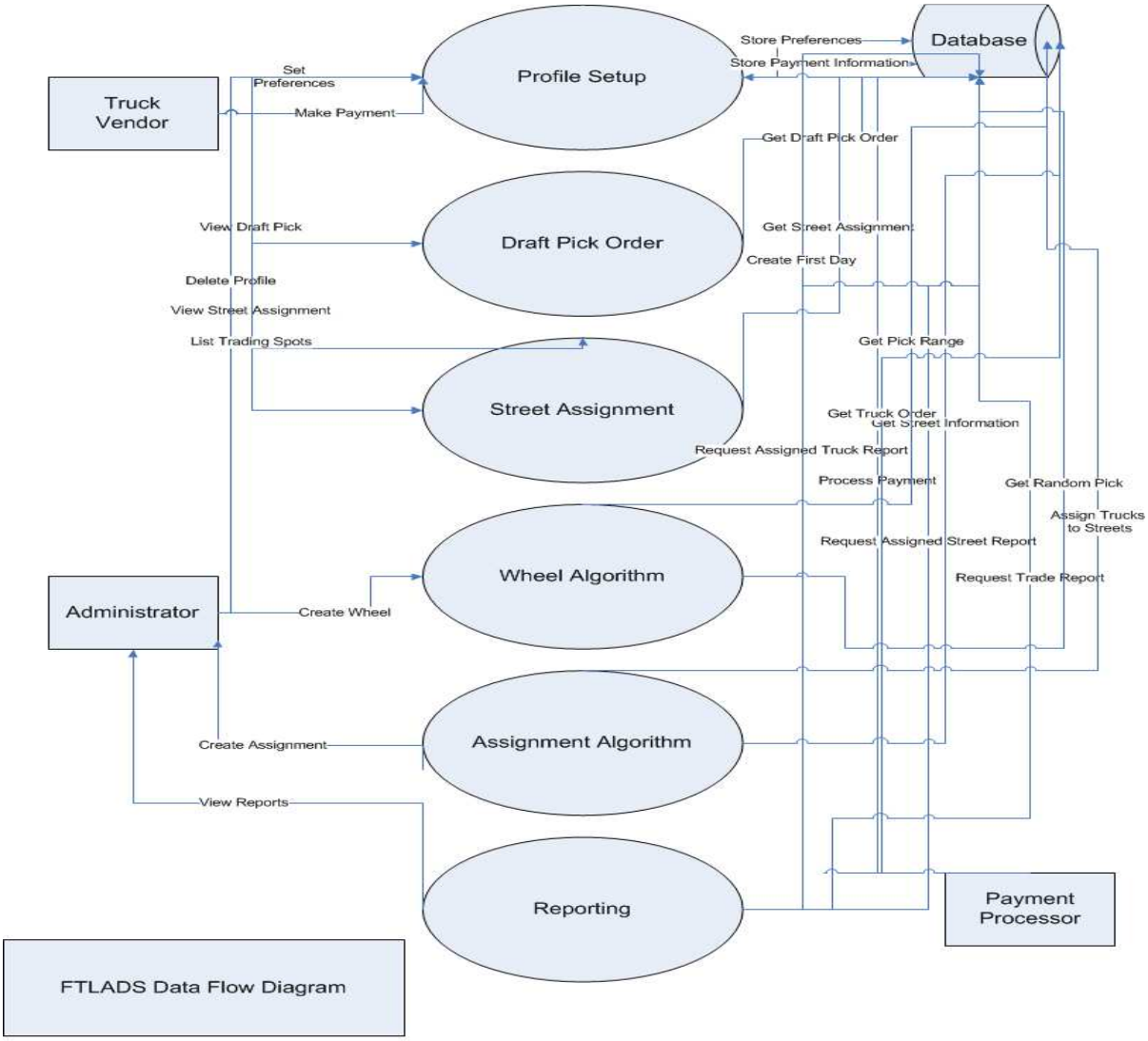
1.7 Diagrams



FTLADS System Architecture



FTLADS Object Model



FTLADS Data Flow Diagram