ECSE 321: Intro to Software Engineering

Project Deliverable#2

Architecture and Design Specification with Prototype

Group 15:

Lea Akkari – 260674169
Mahdis Asaadi – 260663335
Mohammed Edris Jebran – 260742799
Reem Madkour – 260726480
Abed Al Rahman Atassi - 260720213

McGill University, Department of Electrical and Computer Engineering

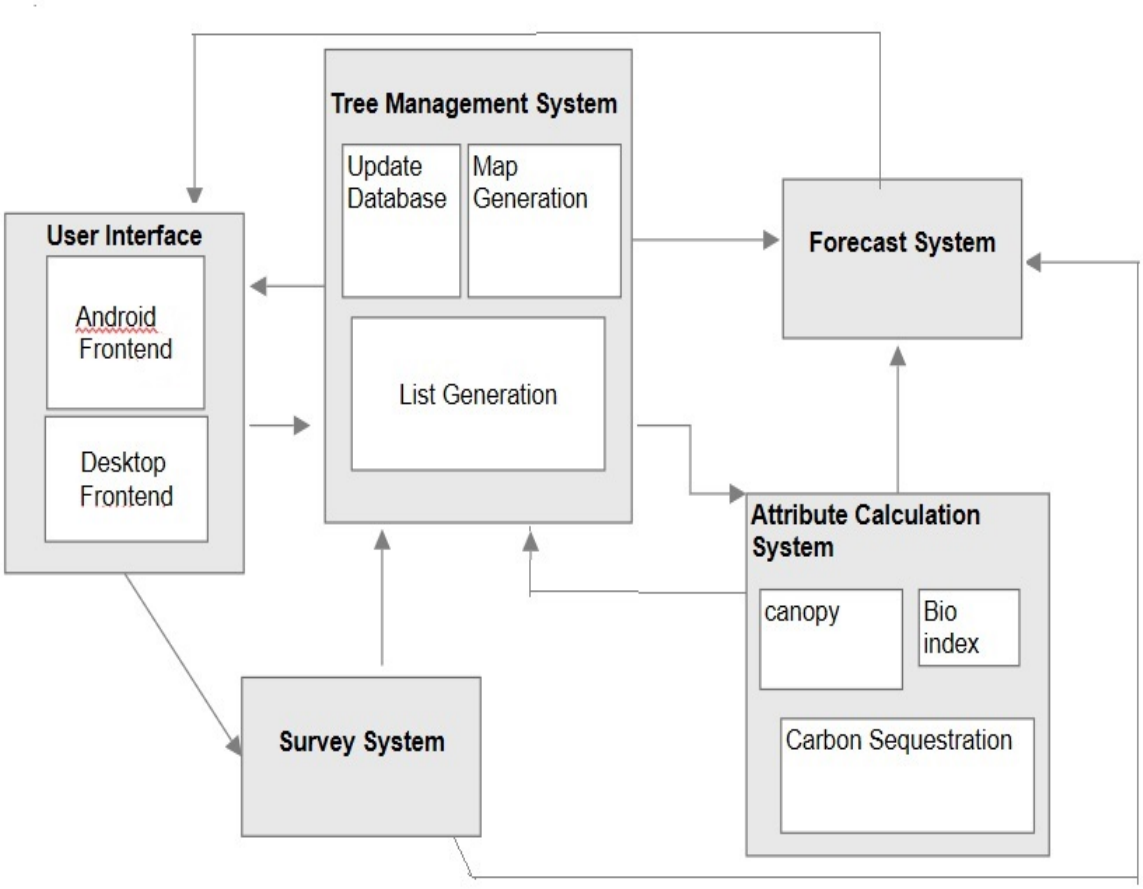
Github Link
https://github.mcgill.ca/ECSE321-2018-Winter/Project-15/commit/32bde47de689e9211e9351733c9debc0f80195e5

Table of Contents

1. Description of Architecture of Proposed Solution with a Block Diagram

Block Diagram

(Part a and b)



Sub-System Description

Overview:

The user Interface system consists of the Android Frontend and the Web Frontend. It is where the system receives the user requests and displays the responses. The requests either are sent to the tree Management System or the Survey system. The Tree Management System handles all requests that are not surveys(all the different versions of listing trees and updating information), but the Survey System handles all the status changes for trees(planted/cutdown/diseased/to be cut down). The survey System then handles these changes through communication with the Tree System and makes the necessary updates.
The Forecast System handles the user requests to generate forecasts based on what if scenarios. It receives its scenarios from the Survey system where the user inputs them, then it sends the data to the attribute calculation system which returns back the corresponding sustainability attributes for the Forecast system to generate a report and return it to the user.
The attribute calculation system can also send attributes directly to the Tree Management System when it is requested to calculate the current tree attributes.

Data:

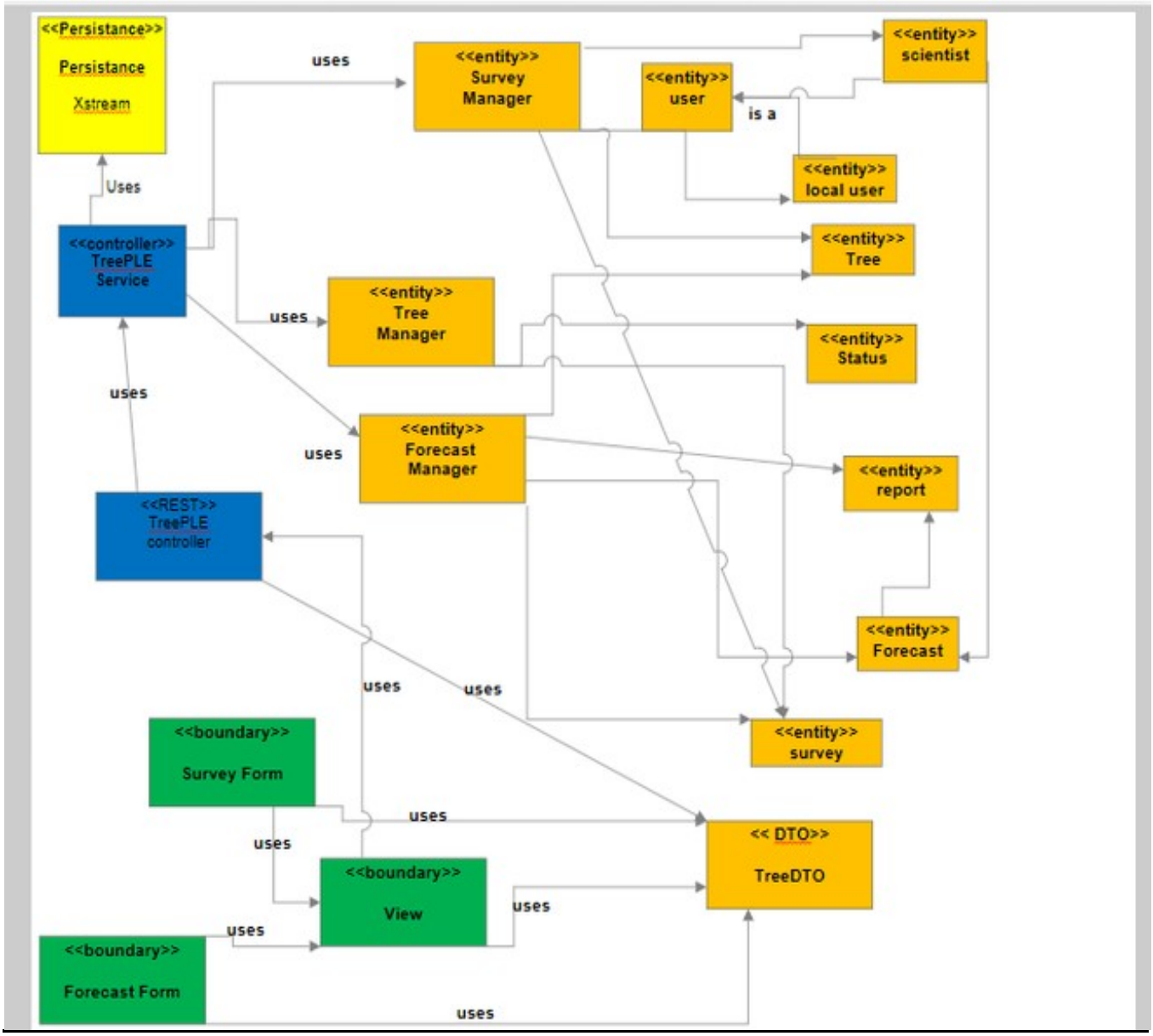
Data to and from the user interface is in the form of Dtos while all the other data transfers between the rest of the subsystems are made using the actual business objects.
Functionalities visible to the user are different depending on the type of user. For a local user, only the Survey System functionalities will be visible, while for an expert the Survey System, the Tree Management System, the Forecast system, and the Attribute calculation System functionalities will be visible.

Architecture:

As a team we started off thinking about the overall project from an MVC point of view. Our system consists of the user interface, the Controller, the Service, and the database. From the MVC point of view, the model can be viewed as our database(persistence), Service and Controller as our Controller, and User Interface as our View. Predicting what the project will look like in the future, there is a possibility that it will start leaning more towards the layered architecture once we start implementing a user authentication system to identify if the user is a local or an expert.

2. Description of Detailed Design of Proposed Solution Including Class Diagram

Detailed Class Diagram



Brief Description of the class diagram

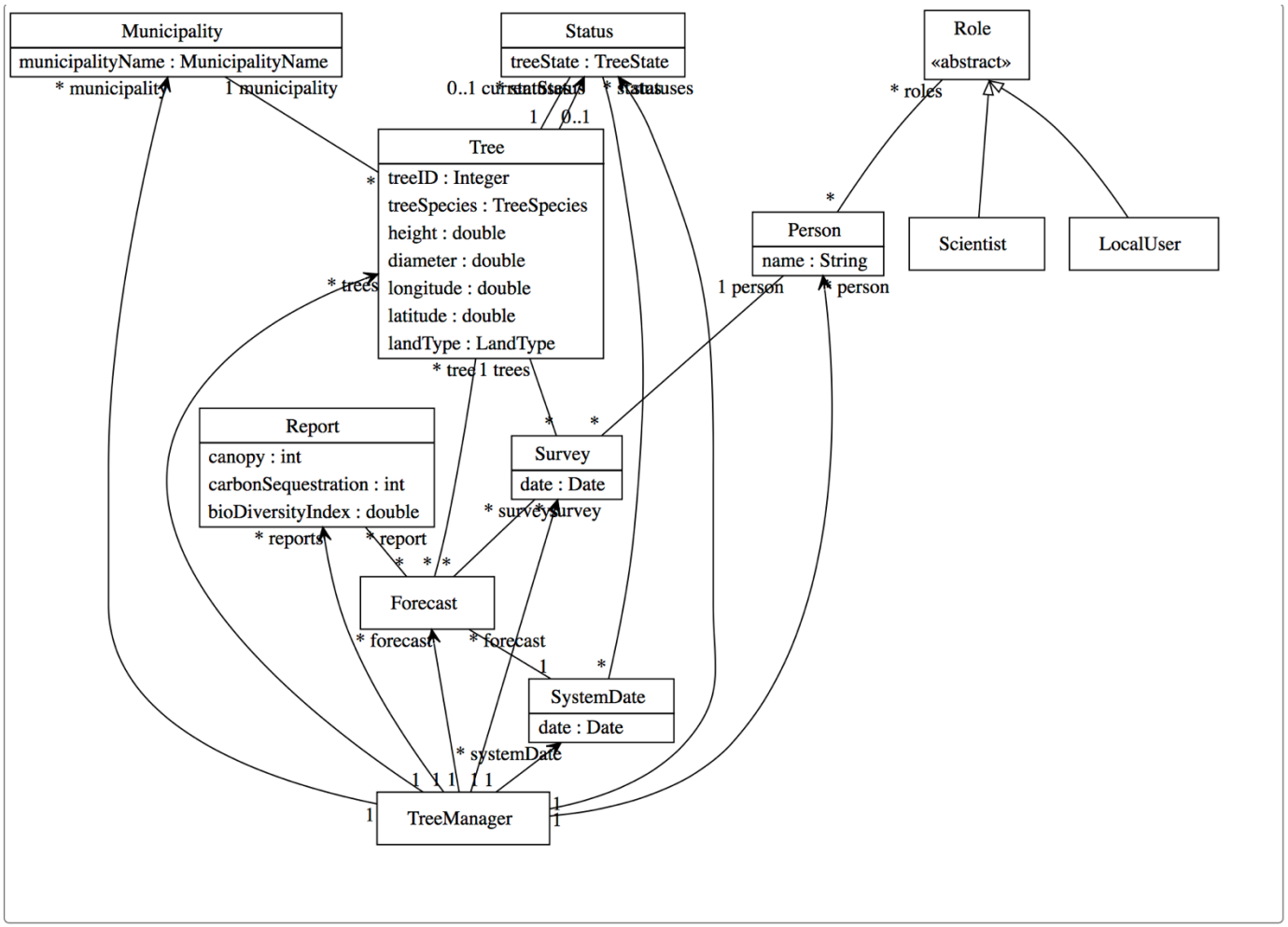
Following the MVC architectural pattern, we have our main classes: Persistence (for which we use Xstream), the controller (TreePLE controller and TreePLE service) and view(consisting of our boundary classes).

For now we are predicting that we will need three managers associated with our controller for different functionalities of the program(Survey manager to handle the user surveys, Tree manager to handle all the other user Tree requests eg:lists and a Forecast manager just for dealing with forecasts and their generated reports). The Service class is also part of the controller, and it is where all the operations on the entity classes happen. Most of the entity classes we will be using are shown on the class diagram. They represent the Model classes that our TreePLE system contains/uses.

Our View right now has 2 forms (Forecast from which the user will utilize to generate forecasts, and the Survey form, which the user will utilize to create surveys). More boundary classes will emerge while we develop the Application, but for now these are the main ones.

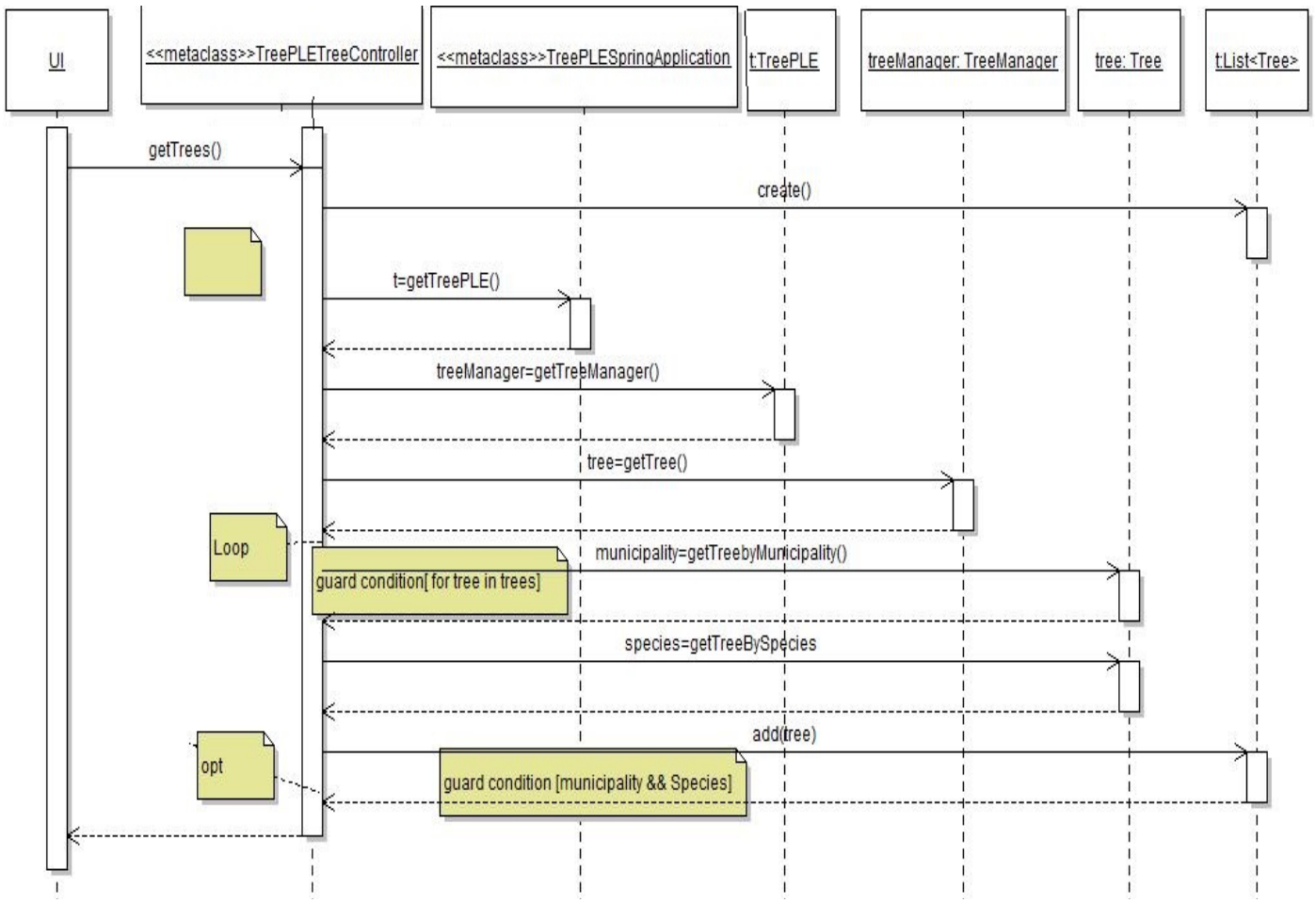
Domain Model Updated

See Umple Code at the end of document

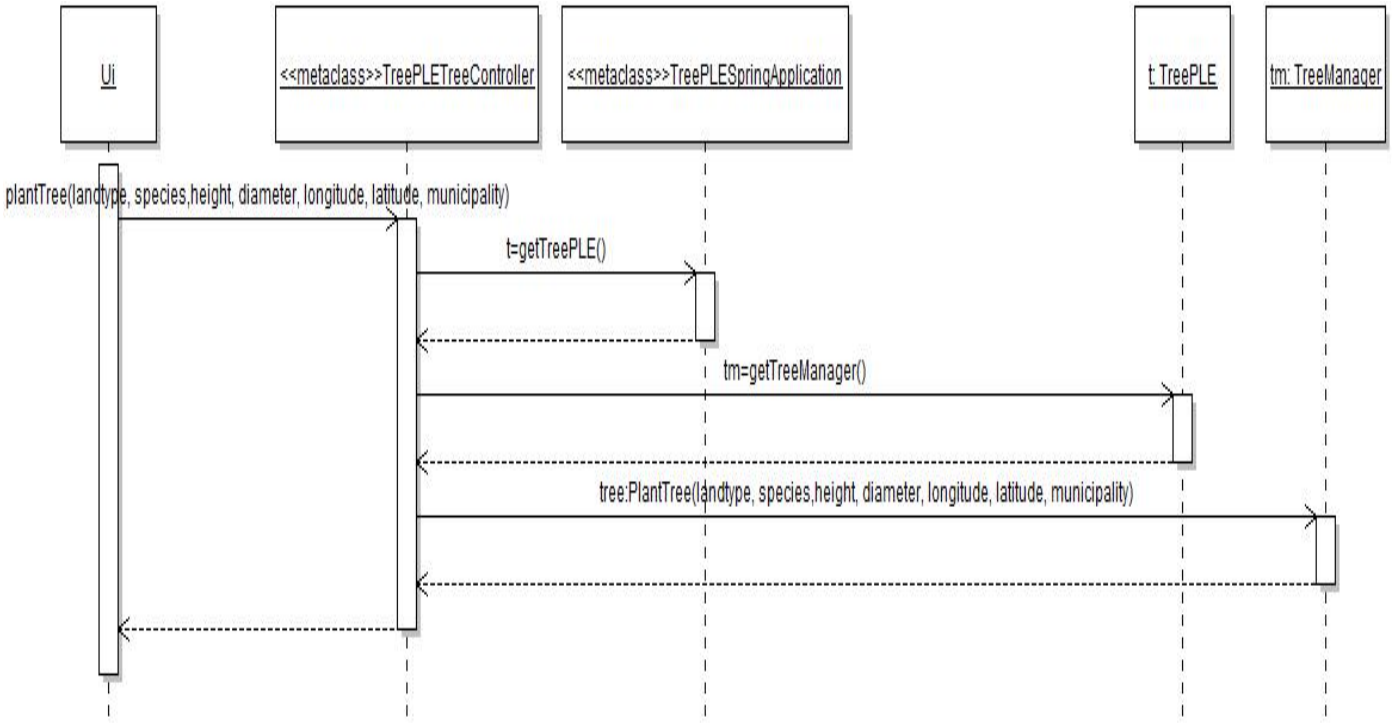


3. Initial Prototype

Sequence Diagram for listAllTrees



Sequence Diagram for plantTree



4. Work Plan and Progress Report

Iteration 1: Requirements Analysis and Domain Modeling

Work Layout:

-Main Tasks:
Functional Requirements: Mahdis and Abed
Non-Functional Requirements: Lea and Edris
Use Cases: Reem
Activity Diagram: Lea and Mahdis
Domain Model : Edris and Abed

Work hours:

-Monday: 2 hours individually
-Thursday: 2 hours individually
-Friday: Group meeting 6:00-10:00pm (Functional requirements and Nonfunctional requirements were done. Everyone was there and decided we should meet Saturday to finalize work, work on use cases, activity diagrams and domain model were started by the assigned individual)
-Saturday: group meeting 4:00 PM to 12:00 AM (The use cases and the diagram, activity diagram and the domain model were completed. The work for Deliverable 1 was finalized.)
Everyone participated and did their part, it was a very good and smooth meeting.

Iteration 2-6: Left to be done

Iteration 2: Design Specification and Prototype

Time needed to complete: 41 hours – Done February 22nd

Work Layout:

- **Description of architecture, Block diagrams: Reem and Edris**
- **Description of detailed design, class diagram : Reem and Abed**
- **Prototype implementations: Lea, Mahdis , Abed**
- **Implementation-level sequence diagrams: Lea and Mahdis**

Work hours:

- Saturday 17th : 1st group meeting for deliverable 2 – Assigned tasks for everyone (3 hours)
- Sunday 18th : 2nd group meeting – Updating the domain modeling (3 hours)
- Monday 19th : 3rd group meeting – Agreed on code structure, sequence diagrams and class diagram (3 hours)
- Tuesday 20th : Met with our mentor to finalize the domain modeling (2 hours)
- Wednesday 21st : Started coding (5 hours)
- Thursday 22nd : Coding (10 hours)
- Friday 23rd : Coding (5 hours)
- Saturday 24th : Last minute bug fixes (5 hours)
- Sunday 25th : Implemented map API on Web Frontend and finalized PDF (5 hours)

Iteration 3-6: Left to be done

Next steps: Things to keep in mind for later:

- **Our code should be stronger – for now we have the very basic methods to plant tree and cut down tree. Our code should take into account all user input without crashing** *(What is user is planting a tree in a location where a tree already exists- what if user is cutting a tree that doesn't exist – Is the tree the user is cutting/planting in his own municipality?)*
- **Our code should take into account user rights and restrictions – Is it a local user or a scientist***? (A local user can only cut/plant a tree in his own municipality. A local user only has access to the android application. A scientist has access to the website and to the android app. A scientist can also correct mistakes in database and list tree as diseased. Etc.)*
- **Our code should implement an SQL database** *(Using xml files isn't very efficient. Implementing SQL would be much easier, the code will be more efficient and the database can be updated faster with better performance). For now, our persistence layer is similar to the one in eventRegistration (xml) We kept it simple and basic for this deliverable.*
- **We should figure out a way to implement the map in the Android Frontend.** *(The Android VM doesn't have access to the internet. How can we implement the map and update the database?)*
- **We're still not sure about all the DTOs. We will be adding more and updating our domain model regularly depending on our needs. Nothing is finalized yet. We see this deliverable as a "first draft".**

Iteration 3: Quality Assurance Plan:

Estimated time needed to complete 35 hours, **should be done by March 10th**.
Workload will be divided amongst the team members, 3 hours of individual work and at least 12 hours of group work, the work will be done spread over the week of February 27, so as to have more time if needed.

Iteration 4: Release Pipeline Plan

Estimated time needed to complete 25 hours, **should be done by March 20th.**
Over the 10 days from March 1st to 20th ,the team will meet regularly to work together, if more work needs to be done, extra hours will be allocated to work.

Iteration 5: Presentation

Estimated time needed to complete 15 hours, **should be done by April 1st.**
The team will allocate at least 15 hours to prepare the presentation, and get work done. 3-4 times a week the team will meet to work, prepare the presentation.

Iteration 6: Final Application:

Estimated time needed to complete 30 hours, **should be done by April 8th.**
The team will meet regularly to finalize the project, make updates, test, once everything works, source code will be uploaded to GitHub repo.

5. UMPLE CODE

```
namespace ca.mcgill.ecse321.TreePLE.model;

class TreeManager{
1 -> * Tree trees;
1 -> * Person person;
1 -> * Report reports;
1 -> * Municipality municipality;
1 -> * Forecast forecast;
1 -> * SystemDate systemDate;
1 -> * Survey surveys;
 1 -> * Status statuses;
}

class Tree{
autounique treeID;

enum TreeSpecies {Willow}
lazy TreeSpecies treeSpecies;

double height;
double diameter;
double longitude;
double latitude;

enum LandType{residential,institutional, park, municipal}

lazy LandType landType;
1 -- * Status statuses;
 *--1 Municipality municipality;
  0..1  -> 0..1 Status currentStatus;
}

class Municipality{
  enum MunicipalityName {Montreal, Laval}
  lazy MunicipalityName municipalityName;
}

class Status{
 enum TreeState {Planted,   Diseased, ToBeCut, Cut}
 lazy TreeState treeState;
```

```
}

class Person {
name;
 * -- * Role roles;
}

class Role {
 abstract;
}

class Scientist{
isA Role;
}

class LocalUser {
 isA Role;
}

class Report{
int canopy;
int carbonSequestration;
double bioDiversityIndex;


}

class SystemDate {
Date date;
 * -- * Status status;
  1 -- * Forecast forecast;
}

class Forecast {
* -- * Report report;
 * -- * Survey survey;
 * -- * Tree tree;
}

associationClass  Survey {
 Date date;
  * Person person;
 * Tree trees;
}}
```