ECSE 321

Introduction to Software Engineering

Deliverable 1

Requirements Analysis & Domain Modeling

February 11, 2018

Group 11

Alexander Harris - 260688155

Abbas Yadollahi - 260680343

Filip Bernevec - 260689062

Yunus Can Cukran - 260669715

Gareth Peters - 260678626

Table of Contents

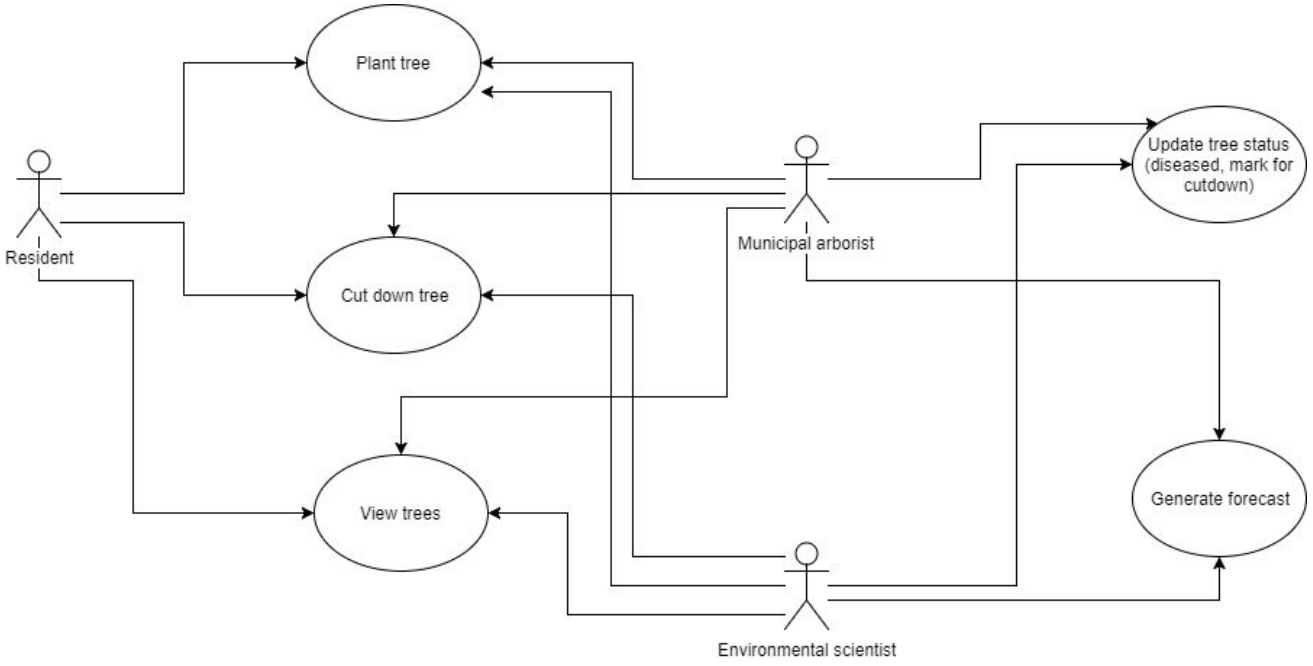
System Requirements

For both Web and Android applications of the Tree Planning Environment (TreePLE) software, we created a set of system requirements that the project should satisfy. The requirements are classified using a scale from 1 (highest priority) to 5 (lowest priority). There are a total of 16 requirements:

1. The system shall have a login page to differentiate each user between resident or specialist. (1)
2. The system shall store the exact location, municipality, species, status, land type, height, and diameter of canopy for each tree while giving it a unique id. (1)
3. The system shall provide information on the status of each tree (planted, diseased, marked for cutdown or cutdown). (2)
4. The system shall update the tree database whenever a user adds or edits any information about a tree. (1)
5. The system shall load the data of each tree from a database stored in a file. (2)
6. The system shall provide an overview of all trees through a web application, allowing the user to sort all trees by municipality, species, status, height, diameter or land. (2)
7. The system shall allow the user to locate specific trees on a map. (3)
8. The system shall display the attributes of a tree. The attributes to be displayed shall include sustainability attributes, tree height, canopy diameter, species, municipality, location, and status of the tree. (2)
9. The system shall allow for residents to mark that a tree has been planted or that a tree has been cut down on their own property through the use of an Android application. (2)
10. The system shall allow for municipal arborists and environmental scientists to mark trees to be cutdown or trees that are diseased via an Android application. (2)
11. The system shall automatically calculate sustainability attributes from each tree or an area of trees. This includes attributes such as stormwater intercepted, size of produced shade, energy conserved, CO_2 reduction, and biodiversity index. (2)
12. The system shall provide forecasts of different scenarios (impacts of removing or adding trees in a specific area) to users that are arborists or environmental scientists via the web application. (2)

13. The system shall allow for reports to be generated at any time based on specified forecast scenarios. (3)
14. The system shall display the attributes of the tree being surveyed along side the person who surveyed the tree and the date that the survey was performed. (3)
15. The system shall update the data of the trees within 2 seconds of an edit or addition of a tree to prevent duplicate data from being registered in the database. (2)
16. The system shall be accessible to web and Android users. (4)

Use Cases



1 - Plant Tree:

Use Case: Plant Tree

Successful Outcomes: Primary Actor creates a new Tree object and adds it to the database.

Actor(s): Primary Actor: Resident, Municipal arborist, Environmental scientist

Precondition: User has successfully logged in.

Traceability: R9

Main Flow:

1. Primary Actor indicates intention to plant a tree.
2. System prompts Primary Actor to enter tree information.
3. Primary Actor enters information.
4. System creates new Tree object using provided attributes and updates the database.
5. *Use case ends successfully.*

Alternative Flows:

A - Invalid data entry

1. Primary Actor enters incomplete/invalid information.
2. System returns to step 2 in Main Flow and prompts user to enter re-enter correct information.

B - Invalid entry causes process to fail

1. Primary Actor enters information that causes registration process to fail.
2. User aborts.
3. *Use case ends unsuccessfully.*

2 - Cut Down Tree:

Use Case: Cut Down Tree

Successful Outcomes: Primary Actor retrieves desired Tree object and removes the matching entry from the database.

Actor(s): Primary Actor: Resident, Municipal arborist, Environmental scientist

Precondition: User has successfully logged in.

Traceability: R9

Main Flow:

1. Primary Actor indicates intention to cut down a tree.
2. System prompts Primary Actor to specify which tree to cut down.
3. Primary Actor specifies tree to cut down.
4. System retrieves Tree object and removes it from the database.
5. *Use case ends successfully.*

Alternative Flow:

1. Primary Actor specifies tree that he does not have the permission to cut.
2. System returns to step 2 in Main Flow and prompts user to select another tree.

3 - View Trees:

Use Case: View Trees

Successful Outcomes: Primary Actor retrieves a list of all trees in the database corresponding to specified criteria.

Actor(s): Primary Actor: Resident, Municipal arborist, Environmental scientist

Precondition: User has successfully logged in.

Traceability: R6, R8

Main Flow:

1. Primary Actor requests list of trees.
2. System retrieves list of all trees in the database and displays them in the frontend.
3. *Use case ends successfully.*

Alternative Flow:

1. Primary Actor requests list of trees when there are none in the database.
2. System returns error message saying there are no trees to display.
3. *Use case ends unsuccessfully.*

4 - Update Tree Status:

Use Case: Update Tree Status

Successful Outcomes: Primary Actor updates status of specified tree.

Actor(s): Primary Actor: Municipal arborist, Environmental scientist

Precondition: User has successfully logged in, has needed permissions.

Traceability: R10

Main Flow:

1. Primary Actor selects desired tree from list of trees.
2. Primary Actor changes status of selected tree.
3. System updates status of corresponding Tree object in database.
4. *Use case ends successfully.*

Alternative Flow:

1. Primary Actor changes status of tree to current status.
2. System returns error message saying the tree is already in this state.
3. *Use case ends unsuccessfully.*

5 - Generate Forecast:

Use Case: Generate Forecast

Successful Outcomes: Primary Actor generates a forecast based on specified scenario.

Actor(s): Primary Actor: Municipal arborist, Environmental scientist

Precondition: User has successfully logged in, has needed permissions.

Traceability: R12, R13

Main Flow:

1. Primary Actor indicates intention to generate a forecast.
2. System prompts Primary Actor to specify forecast details.
3. Primary Actor enters forecast details and executes forecast generation.
4. System generates forecast and displays it in the frontend application.

5. *Use case ends successfully.*

Alternative Flows:

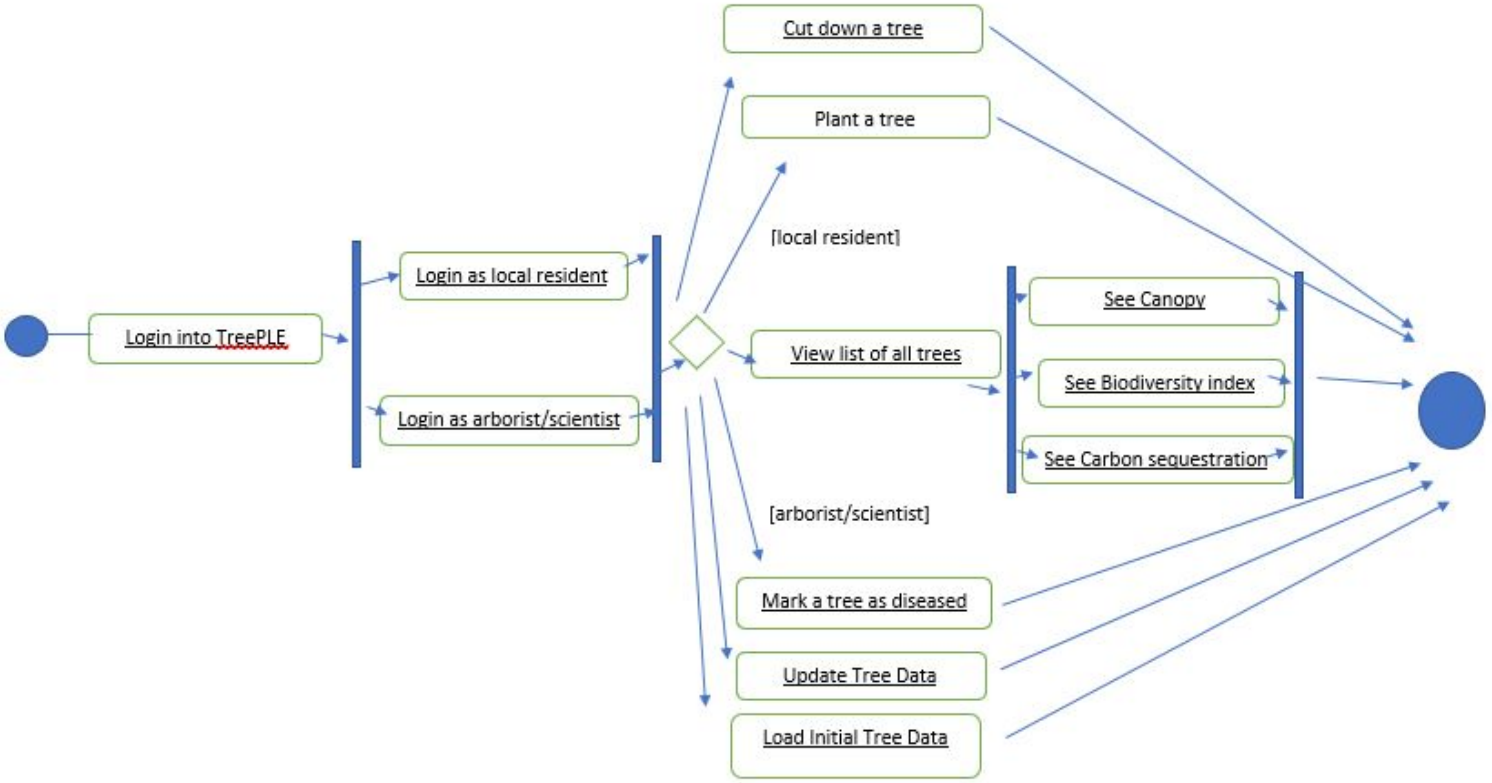
A - Invalid data entry

1. Primary Actor enters invalid/incomplete forecast information.
2. System return to step 2 in Main Flow and prompts Primary Actor to re-enter correct information.

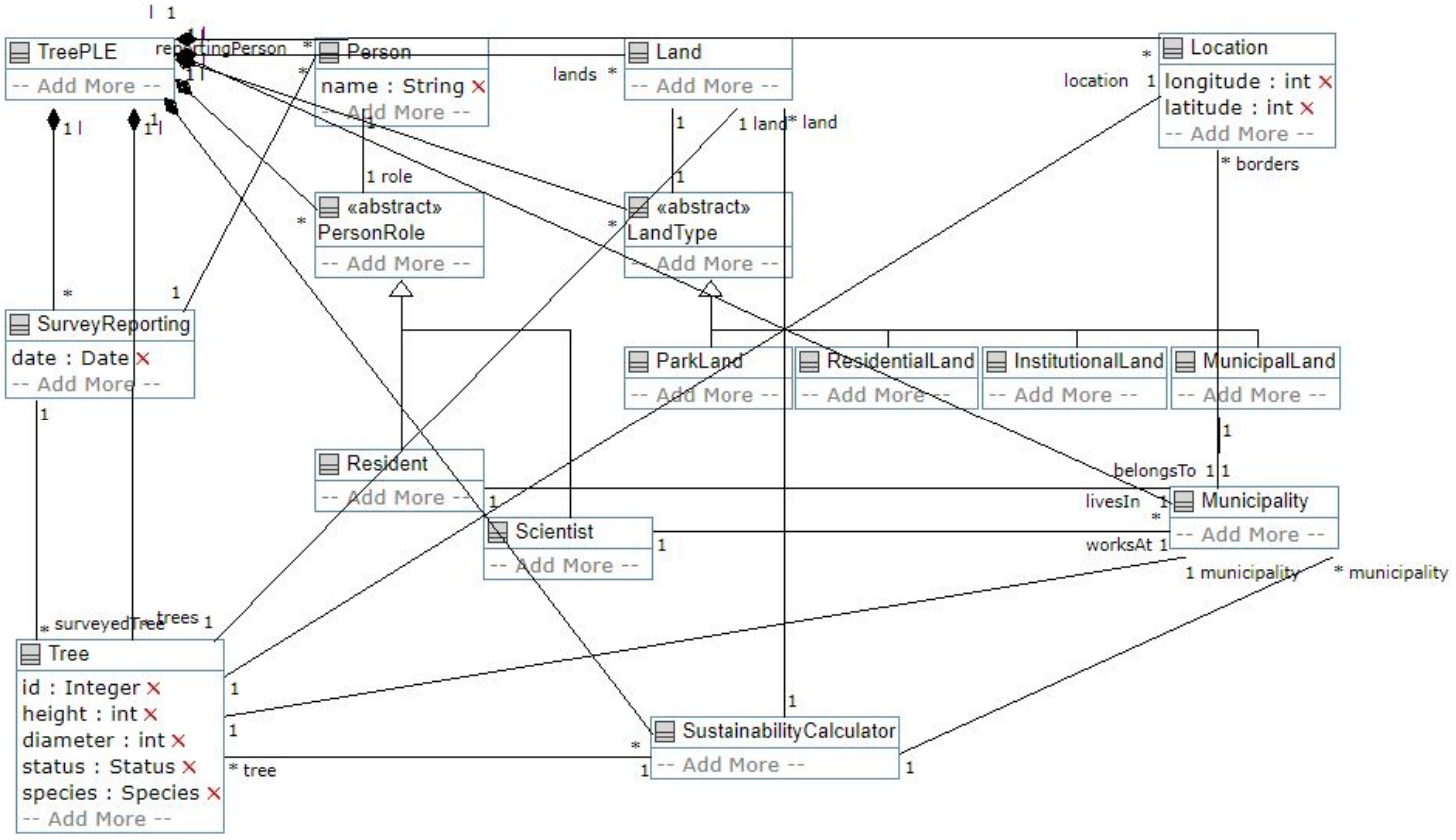
B - Invalid entry causes process to fail

1. Primary Actor enters information that causes forecast generation process to fail.
2. User aborts.
3. *Use case ends unsuccessfully.*

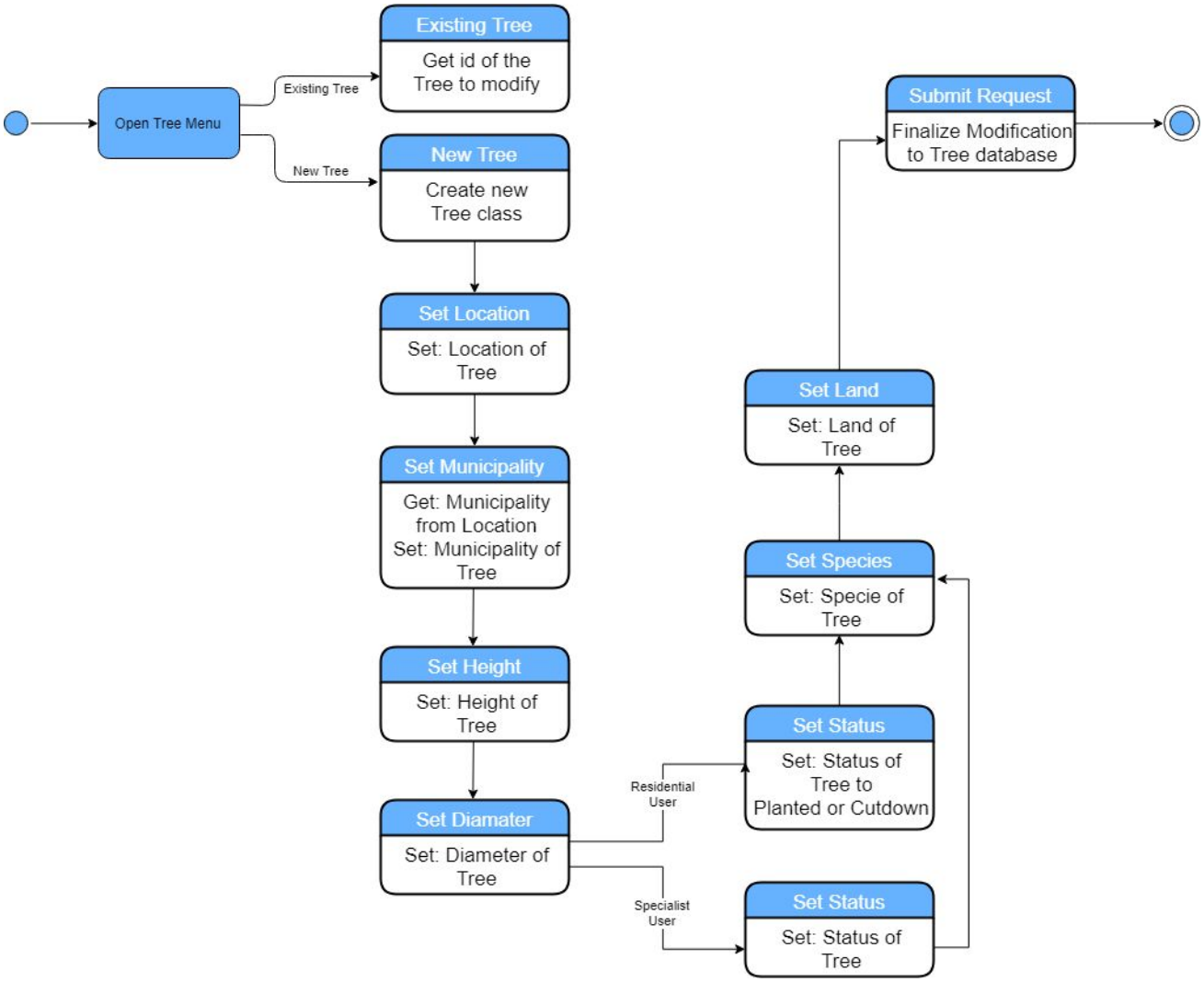
Requirements-Level Activity Diagram



Domain Model



Tree Class Statechart



Work Plan

Week 1 (Week of February 12th)

- Develop a system architecture including block diagrams.
- Describe a detailed solution including class diagrams.
- Develop a prototype including “Plant Tree” and “Cut Down Tree” use cases for the Java Spring backend and the Web and Android frontends.

Week 2 (Week of February 19th)

- Develop a prototype implementation containing the “List All Trees” use case in the backend and frontend.
- Create a implementation-level sequence diagram for “Plant Tree” and “List All Trees” use cases covering all architectural layers.
- Write up Deliverable 2.

Week 3 and 4 (Week of February 26th and March 5th)

- Develop a prototype implementation for the “Update Tree Status” and “Generate Forecast” use cases in the backend and frontend.
- Develop test cases for unit testing, system testing and component testing.
- Start testing the software prototype.

Week 5 and 6 (Week of March 12th and March 19th)

- Write up Deliverable 3.
- Provide a description of the Release Pipeline plan.
- Implementation of extra features and testing those extra features.
- Finalization of Java Spring backend.
- Write up Deliverable 4.

Week 7 (Week of March 26th)

- Finalization of the Android and Web Frontends.
- Prepare for group presentations for the following week.

Week 8 and 9 (Week of April 2nd and April 9th)

- Present the project.
- Finalize all of the source code.
- Submit source code and commit history.

Appendix

UMPLE Code

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

class TreePLE
{
  1 l<@>-* Tree trees;
  1 l<@>-* Land lands;
  1 l<@>-* Person;
  1 l<@>-* Location;
  1 l<@>-* Municipality;
  1 l<@>-* SurveyReporting;
  1 l<@>-* SustainabilityCalculator;
  1 l<@>-* PersonRole;
  1 l<@>-* LandType;
}

class Tree
{
  autounique id;
  int height;
  int diameter;
  1 -- 1 Location location;
  1 -- 1 Land land;
  1 -- 1 Municipality municipality;
  Status status;
  Species species;

  enum Status {
    Planted,
    Diseased,
    MarkedForCutdown,
    Cutdown
  };
  enum Species {
    Oak,
    Elm,
    Maple
  };
}
```

```
class Location
{
  int longitude;
  int latitude;
}

class SurveyReporting
{
  Date date;
  1 -- * Person reportingPerson;
  1 -- * Tree surveyedTree;
}

class Municipality
{
  1 -- * Location borders;
}

class Person
{
  String name;
  1 -- 1 PersonRole role;
}

class PersonRole
{
  abstract;
}

class Resident
{
  isA PersonRole;
  * -- 1 Municipality livesIn;
}

class Scientist
{
  isA PersonRole;
  * -- 1 Municipality worksAt;
}

class Land
{
```

```
  1 -- 1 LandType;
}

class LandType
{
  abstract;
}

class ParkLand
{
  isA LandType;
}

class ResidentialLand
{
  isA LandType;
}

class InstitutionalLand
{
  isA LandType;
}

class MunicipalLand
{
  isA LandType;
  1 -- 1 Municipality belongsTo;
}

class SustainabilityCalculator
{
  1 -- * Tree tree;
  1 -- * Land land;
  1 -- * Municipality municipality;
  int getShadeSize(this.tree);
  int getCO2Reduced(this.tree);
  int getEnergyConserved(this.tree);
  int getStormWaterIntercepted(this.tree);
  int getBioDiversityIndex(this.land);
}
```