Software Design Document

Project Vayu

Lab 2, Group 5

Revision: 0.1

27 Feb, 2020

Oussama Saoudi, Lennon Yu
Christina Korsman, Diego Soriano

SFWRENG 2XB3
Software Engineering Practice and Experience:
Binding Theory to Practice
Department of Computing and Software
McMaster University

Revisions

Revision	Date	Changes
V0.1	27 Feb 2020	Added skeleton

Team Members

Name	Student No.	Role
Oussama Saoudi	400172153	Project Lead, Search Alg. Dev.
Lennon Yu	400183521	Doc. Maintainer, Graph Alg. Dev.
Christina Korsman	400192880	Transcriber, UI/UX Dev.
Diego Soriano	400172910	Grunk, Sort Alg. Dev.

By virtue of submitting this document we electronically sign and date that the work being submitted by all the individuals in the group is their exclusive work as a group and we consent to make available the application developed through SE-2XB3 project, the reports, presentations, and assignments (not including my name and student number) for future teaching purposes.

Contributions

Name	Roles	Contributions	Comments

Summary

Here is the summary.

Contents

1 SDD Identification

1.1 Scope

1.2 Authorship

1.3 Context

1.4 References

[1]Cengproject.cankaya.edu.tr, 2020. [Online]. Available: http://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf. [Accessed: 22- Mar- 2020].

1.5 Context

1.6 Design Languages

The Design language that will be use is UML.

1.7 Body

1.8 Summary

1.9 Glossary

2 Design Stakeholders

The stakeholder of the design subject with respective design concerns are the following:

- Governments
  - Disaster Regions
  - Casualties
  - Severity Indicator
- Non-profit Organizations
  - Disaster Regions
  - Casualties
  - Severity Indicator

- Insurance Companies
  - Property Damage
  - Severity Indicator

3 Design Views

4 Design Viewpoints

4.1 Context viewpoint

Context viewpoint depicts all the services provided.

4.1.1 Design concerns

4.1.2 Design entities

The external active elements that the system will be working with, is the user and the data set.

4.1.3 Design relationships

The system will receive location data , or filter data from the user. With this input the system will out the severity and disaster from the surround area. if applicable the filter on the type of disaster in that area.

4.1.4 Design Constraints

4.2 Composition viewpoint

Summary of system composition here

4.2.1 Design concerns

NYI

4.2.2 Design entities

NYI

4.2.3 Design Relationship

NYI

4.2.4 Design attributes

NYI

4.3 Logical viewpoint

QuickSort

Module QuickSort

Uses

N/A

Syntax

* Exported Constants None

Exported Types

Exported Access Programs

Routine Name	In	Out	Exceptions

Semantics

State Variables

* State Invariant None

QuickSort

Module QuickSort

Uses

N/A

Syntax

* Exported Constants None

Exported Types

Exported Access Programs

Routine Name	In	Out	Exceptions

Semantics

State Variables

* State Invariant None

DisasterPoint

Module

DisasterPoint

Uses

N/A

Syntax

4.3.1

* Exported Constants None

Exported Types

DisasterPoint = ?

Exported Access Programs

Routine Name	In	Out	Description
disaster		DisasterType	Returns the disaster type of the datapoint.
latitude		double	Returns the latitude position of the datapoint.
longitude		double	Returns the longitude position of the datapoint.
casualties		int	Returns the number of casulaties caused by the disaster.
damage		int	Returns the amount of property damge in USD.

Semantics

State Variables

$disaster : DisasterType$
$lat : int$
$long : int$

$casualties : int$
$damage : int$

State Invariant

None

Design Concerns

KdTree

Module

KdTree

Uses

DisasterPoint

Syntax

4.3.2

* Exported Constants None

Exported Types

KdTree = ?

Exported Access Programs

Routine Name	In	Out	Description
sortByPorximity	lat: int, long: int	ArrayList¡DisasterPoint¿	Returns a sor nodes sorted by imity to a given
getPoints	point: DisasterPoint, radius: int	Set¡DisasterPoint¿	Returns set of are within the the DisasterPoin same DisasterTy

Semantics

State Variables

State Invariant

None

Design Concerns

This module facilitates the sorting by proximity functinoal requirment ([FR1.4]) using the method sortByProximity(). In addition it assists in creating connections to the graph, which facilitates the creation of disaster regions for functional requirements [FR2.1] and [FR5.3]

CCFinder

Module

CCFinder

Uses

Graph, DisasterPoint

Syntax

Exported Constants None

Exported Types

None

Exported Access Programs

Routine name	In	Out	Description
getCC	Graph	Set¡Set¡DisasterPoint¿¿	Generates the set of connected components and returns a set of set of nodes, where each set of nodes is a conncected component.

Semantics

State Variables

None

State InvariantNone

Design Concerns

Used to group sets of nodes in a graph into connected components. Used in making convex hull which lead to making DisasterAreas.

Parser

Module

Parser

Uses

N/A

Syntax

Exported Constants
None

Exported Types

getData() = ArrayList<DisasterPoint>

Exported Access Programs

Routine name	In	Out	Exceptions
new Parser	Landtypes	LanduseT	

4.3.3

* Considerations
When implementing in Java, use enums (as shown in Tutorial 06 for ElementT).

4.4 Dependency viewpoint

This viewpoint highlights the relationships and interconnections amongst the different packages and methods in this project.

4.4.1 Design concerns

4.4.2 Design entities

4.4.3 Design relationships

4.4.4 Design Attributes

4.5 Information viewpoint

This viewpoint shows the persistent data structures that will be apply in this project.

4.5.1 Design concerns

Concerns of this viewpoint are the persistent data structures. The way that this will be address is that the data will be stored in a Graph. This covers the functional requirements [FR2.1], [FR2.1]

4.5.2 Design entities

Modules

- QuickSort - A module that implements the quick sort algorithm on the graph
- Casualties Comparator - A module compare the casualties of two disasters
- Property Damage Comparator - A module compare the property damage of two disasters
- Proximity Comparator
- DisasterPoint - A class that store the data relating to a disaster occurrence.
- Weather Type Enum - A Data type for all type of disasters in the data set
- Parser -Take the file and transfers the data to another class to be used
- Filter - A class that filters out certain weather types
- Graph - A graph data strucure
- Connected Components Finder (Algorithm)- A module that implements algorithm that finds connected components on the Graph
- Convex Hull finder(Algorithm) -A module that implements an algorithm that finds the convex hull on the graph

- Convex Hull - A module that implements the Convex Data structure
- KD-Tree (Data Structure)- A module that implements the KD- Tree Data structure

4.5.3 Design relationships

- Quicksort use the DisasterPoint class as it the object type to be sorted.
- Causalities Comparator will use DisasterPoint. It will use the getter accessor methods to allow the comparison
- Property Damage Comparator will use DisasterPoint. It will use the getter accessor methods to allow the comparison
- Promitiy Comparator will use DisasterPoint. It will use the getter accessor methods to allow the comparison
- DisasterPoint use Enum for the disaster type that it will store for that occurrence
- Filter use Weather Type to apply the filter to be used.
- Graph class use the DisasterPoint class to create a graph data structure
- Parser use Nodes, taking data and separating it into the correct places for the DisasterPoint.
- finder use Graph and Convex Hull
- Convex hull will use graph
- KD -tree will use graph

4.5.4 Design Attributes

DisasterPoint will be persistently use through out this software,as it is the main container for the data.

4.6 Patterns use viewpoint

Summary of patterns use viewpoint here.

4.6.1 Design concerns

NYI

4.6.2 Design entities

NYI

4.6.3 Design relationships

NYI

4.6.4 Design Attributes

NYI

4.6.5 Design Constraints

NYI

4.7 Interface viewpoint

4.7.1 Design concerns

4.7.2 Design elements

4.8 Algorithm viewpoint

4.8.1 Design concerns

4.8.2 Design Attributes

5 Design Overlays

6 Design Rationale

7 Review