Intermediate Report

3 Ba INF 2018-2019

Evelien Daems
Benjamin Vandersmissen
Frank Jan Fekkes
Lars Van Roy

1 Introduction

This report gives a brief overview of the current status of our project. It lists the completed segments, as well as the segments that are not. We will end every segment by a summary of the altered files and or the files that are related to the given segment.

2 Demographic Profile

2.1 progress

The demographic profile section is completely finished. It currently reads the proper configuration and applies the provided information where needed. The used algorithms are described in section 5.2 of the user manual.

There are unit and scenario test that test the implementation of demographic profile. The scenario tests will test whether the information is properly read and that this information is properly applied in the respective generators. These tests will only test the generator for extreme values, where there are only a few types of ages, since the distribution can not exactly be tested if there are multiple (as randomness would influence this). I did however manually test a couple of scenarios where the results were all reasonable and within the expected bounds. The available tests and the functionality they intend to test are all documented in the test plan under section 5.2

2.2 Affected Files

`reading input data`: code that handles reading the data

- `main/cpp/pop/GeoPopBuilder.cpp`:
  The readers will be called here
- `main/cpp/geopop/io/HouseholdReader.h`:
  The super class for all workplace readers
- `main/cpp/geopop/io/HouseholdCSVReader (.h/.cpp)`:
  The specific reader implementation for CSV

- `main/cpp/geopop/io/MajorCitiesReader (.h/.cpp)`:
  The reader that reads which cities are major

`storing input data`: code that handles storing the data

- `main/cpp/geopop/GeoGridConfig (.h/.cpp)`:
  This will store all needed data

`generators`: code that handles generating of pools

- `main/cpp/geopop/generators/HouseholdGenerator.cpp`:
  Altered generator that generates household pools
- `main/cpp/geopop/generators/WorkplaceGenerator.cpp`:
  Altered generator that generates workplaces pools
- `main/cpp/geopop/generators/K12SchoolGenerator.cpp`:
  Altered generator that generates K12schools pools
- `main/cpp/geopop/generators/PreSchoolGenerator.cpp`:
  Altered generator that generates preschools pools
- `main/cpp/geopop/generators/DaycareGenerator.cpp`:
  Altered generator that generates daycare pools

`generators`: code that handles generating of pools

- `main/cpp/geopop/populators/HouseholdPopulator.cpp`:
  Altered populator that populates household pools
- `main/cpp/geopop/populators/WorkplacePopulator.cpp`:
  Altered populator that populates workplaces pools
- `main/cpp/geopop/populators/K12SchoolPopulator.cpp`:
  Altered populator that populates K12schools pools
- `main/cpp/geopop/populators/PreSchoolPopulator.cpp`:
  Altered populator that populates preschools pools
- `main/cpp/geopop/populators/DaycarePopulator.cpp`:
  Altered populator that populates daycare pools

3 Workplace Distribution

3.1 progress

The workplace distribution is completely finished. It currently reads from an input file in case this file is given, or, in case this was not given, it will fall back to the original implementation of the workplace distributions. In case the workplace distribution configuration was given, the algorithm described in the user manual under section 5.3 will be used.

There are also tests, that will test most of the functionality of this section. It is both tested in unit tests, and scenario tests of which the last ones will test if the reader will properly pass the read information to the generator. The tests will only test the generator for the more extreme values where there is only one

type of workplace size, with a 100% chance of occurring. It has been manually tested for multiple kinds of workplaces and this does seem correct, but this is far harder to test as the generated values could be widely spread out due to randomness. The made tests and the functionality they intend to test is all documented in the test plan under section 5.3

3.2 Affected Files

`reading input data`: code that handles reading the data

- `main/cpp/pop/GeoPopBuilder.cpp`:
  The reader will be called here
- `main/cpp/geopop/io/WorkplaceReader.h`:
  The super class for all workplace readers
- `main/cpp/geopop/io/WorkplaceCSVReader (.h/.cpp)`:
  The specific reader implementation for CSV

`storing input data`: code that handles storing the data

- `main/cpp/geopop/GeoGridConfig (.h/.cpp)`:
  This will store all needed data

`workplace generator`: code that handles generator of workplaces

- `main/cpp/geopop/generators/WorkplaceGenerator.cpp`:
  Altered generator that generates workplaces

`workplace populator`: code that handles population of workplaces

- `main/cpp/geopop/populators/WorkplacePopulator.cpp`:
  Altered populator that populates workplaces

4 Data Formats

4.1 JSON: GeoGrid

Code

- `main/cpp/geopop/io/GeoGridJSONReader (.h/.cpp)`
- `main/cpp/geopop/io/GeoGridJSONWriter (.h/.cpp)`
- UserManual:
  - `section 5.4.1`

Tests

- `test/cpp/gtester/geopop/io/GeoGridJSONReaderUnitTest.cpp`
  - Testplan: `section 3.1.4 - 3.1.11`
- `test/cpp/gtester/geopop/io/GeoGridJSONWriterUnitTest.cpp`
  - Testplan: `section 3.1.12 - 3.1.15`
- `test/cpp/gtester/geopop/io/GeoGridJSONReaderWriterScenarioTest.cpp`
  - Testplan: `section 3.1.16`

4.2 JSON: Household

Code

- `main/cpp/geopop/io/HouseholdJSONReader (.h/.cpp)`
- `main/cpp/geopop/io/ReaderFactory.cpp`
- UserManual:
  - `section 5.4.2`

Tests

- `test/cpp/gtester/geopop/io/HouseholdJSONReaderTest.cpp`
  - Testplan: `section 3.1.1 - 3.1.2`