HAW
HAMBURG

BACHELOR THESIS
Viviam Clara Ribeiro Guimaraes

Modeling and Predicting Routes of Internally Displaced Persons: An Agent-Based Approach Using the MARS Framework in the Context of the Syrian Refugee Crisis

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Viviam Clara Ribeiro Guimaraes

Modeling and Predicting Routes of Internally Displaced Persons: An Agent-Based Approach Using the MARS Framework in the Context of the Syrian Refugee Crisis

Bachelor thesis submitted for examination in Bachelor´s degree
in the study course *Bachelor of Science Applied Computer Science*
at the Department Computer Science
at the Faculty of Engineering and Computer Science
at University of Applied Science Hamburg

Supervisor: Prof. Dr. Thomas Clemen
Supervisor: Prof. Dr. Stefan Sarstedt

Submitted on: 20 October 2023

Viviam Clara Ribeiro Guimaraes

Title of Thesis

Modeling and Predicting Routes of Internally Displaced Persons: An Agent-Based Approach Using the MARS Framework in the Context of the Syrian Refugee Crisis

Keywords

Agent-Based Modeling, Internal Displacement, Computational Simulation, Predictive Modeling, Forced Migration

Abstract

This thesis investigates the application of an agent-based model (ABM) to predict the distribution of Syrian internally displaced persons (IDPs) and assess its utility for humanitarian resource allocation. Leveraging the MARS framework, the study adapts an existing ABM designed for Syrian refugee distribution in Turkey to simulate the movement of internally displaced persons within Syria. The research highlights the model's potential for informing resource allocation decisions at the statewide level, accurately identifying governorates with the highest IDP populations. However, limitations in predicting finer-grained administrative levels and migration routes are identified, emphasizing the need for further research and enhancements. Despite these challenges, this work paves the way for future research to improve the model's accuracy and usefulness for humanitarian actors in active conflict zones.

Kurzzusammenfassung

Diese Arbeit untersucht die Anwendung eines agentenbasierten Modells (ABM) zur Vorhersage der Verteilung syrischer Binnenvertriebener und deren Eignung für die Zuweisung humanitärer Ressourcen. Unter Verwendung des MARS-Frameworks wurde ein vorhandenes ABM angepasst, um die Bewegung der Binnenvertriebenen innerhalb Syriens zu simulieren. Die Studie betont das Potenzial des Modells, fundierte Entscheidungen zur Ressourcenzuweisung auf Landesebene zu unterstützen und Gouvernements mit der höchsten IDP-Bevölkerung genau zu identifizieren. Es gibt jedoch Einschränkungen bei der Vorhersage auf feineren Verwaltungsebenen und bei Migrationsrouten, was weitere Forschung und Verbesserungen erfordert.

Contents

List of Figures

List of Tables

1 Introduction

In a world where conflicts and crises continue to displace countless individuals from their homes, the accurate assessment of affected populations is more critical than ever before. Among these displaced populations, Internally Displaced Persons (IDPs) represent a particularly vulnerable group, as their size and locations often remain obscure, hindering the efficient provision of essential humanitarian assistance (Doocy et al., 2015). A model that could accurately predict flight routes and destinations based on real-time conflict events would improve the targeting and efficiency of relief and assistance responses (Harrison, 2016).

Extensive research has been conducted on forced migration, with various methodologies employed to model and predict the movement of displaced populations with statistical modeling being the most common one (Hébert et al., 2018), (Sokolowski et al., 2014). Prior efforts have predominantly focused on modeling the displacement of refugees rather than IDPs, primarily due to the scarcity of systematic data available for the latter (Suleimenova et al., 2017). However, existing models face challenges which hinders their applicability.

Notable works in the field of forced migration research include Hébert and Perez's agent-based model for Syrian refugee movement (Hébert et al., 2018) and Sokowinski and Banks' model for displacement in Aleppo (Sokolowski et al., 2014), both of which overestimate refugee numbers. Harrison's research which explores linear regression models for discovering internal displacement influencers in Syria (Harrison, 2016). Gulden, Harrison, and Crooks' hybrid spatial interaction agent-based model which seeks to understand displacement dynamics in East Africa but doesn't produce valid results (Gulden et al., 2011). Güngör, Günnec and Salman's agent-based model for the prediction of migration paths in Syria which could not be validated (Güngör et al., 2022). Suleimenova and Bell's generalized simulation approach with the FLEE model (Suleimenova et al., 2017), the validity and practicability of which has been criticised (Richey, 2020). Richey's pioneering work on scalable agent-based modeling, which is the first empirically validated model of forced migration to include the influence of social networks providing a promising foundation for further research (Richey, 2020).

1.1 Goal

The overarching goal of this thesis is twofold: to redesign the existing agent-based model as outlined in Richey's work (Richey, 2020) using the MARS framework. The second goal is to extend the MARS ABM to encompass the simulation of Internally Displaced Persons (IDPs) emerging from the Syrian Refugee Crisis. The aim is to assess whether the modified model, inclusive of IDPs holds the potential to be useful for humanitarian organizations in their resource distribution planning. Should the model's predictive accuracy fall short, the main sources of error will be identified, contributing valuable insights to enhance the predictive capabilities of future research.

1.2 Structure Outline

To achieve the goal described in the last section, the thesis will progress through various phases, including sociological background research, the presentation of methods and materials, defining requirements, conceptual model development, adaptation design for the MARS Framework, implementation details, verification and validation procedures, predictive scenarios description, results evaluation, a comprehensive discussion of findings, and ultimately, a conclusion outlining the model's utility and potential next steps for research in this domain.

2 Sociological Background

As mentioned in the previous chapter the model aims to simulate the movement of Syrian refugees and IDPs emerging from the Syrian refugee crisis. This chapter offers a concise overview of key statistics and information pertaining to this crisis. Furthermore different types of decision making mechanisms in agent-based models of forced migration are analysed.

2.1 Syrian Refugee Crisis

The Syrian Refugee Crisis, ongoing for more than a decade, is the world's largest refugee crisis. Since 2011, over 14 million Syrians have been forced to flee their homes in search of safety (UNHCR, 2023). Those who remain within Syria's borders, without crossing international boundaries, are referred to as Internally Displaced Persons (IDPs) (Thibos, 2014). Alarmingly, more than 6.8 million Syrians are internally displaced, with approximately 70 percent of the population requiring humanitarian assistance. Moreover, approximately 5.5 million Syrian refugees reside in the neighboring countries of Turkey, Lebanon, Jordan, Iraq, and Egypt (UNHCR, 2023). This crisis has placed significant strain on both Syria and its neighboring nations, necessitating a deeper understanding of the factors influencing displacement for resource allocation.

2.2 Behavioral Theory

A critical part of agent-based modelling is the description of the mechanisms of decision making and social interactions of agents (Klabunde, Willekens, 2016). This section provides an overview of the decision-making theories commonly employed in ABMs of migration and examines their appropriateness for modeling internal displacement.

Decision Making in Agent-Based Models Klabunde and Willekens emphasize the critical role of decision-making mechanisms in ABMs, ranging from simple to complex processes. These mechanisms typically involve agents evaluating their options, translating these evaluations into decisions, and often incorporating random elements to determine action. Empirical data can also play a crucial role in parameter estimation and model validation (Klabunde, Willekens, 2016).

According to Klabunde and Willekens, ABMs of migration can be categorized into six types based on the theories of decision making employed. These are shown in table 2.1.

Model Type	**Characteristics**	**Pros**	**Cons**
Minimalist	Simple rules producing macrooutcomes	Simplicity	Lacking empirical evidence
Expected Utility	Alternative with the highest expected utility is chosen	Has shown accurate predictions	Assumes rational behavior
Psycho-Social	Rules derived from social psychology	Demonstrated empirical relevance	High complexity due to the numerous influencing factors
Heuristics-Based	Uses heuristics to simplify decision making	Simple; allows for social influence	Oversimplification of decision theory
Decision Theory with Empirical Calibration	Decision theory combined with direct observation. Coefficients calibrated using empirical data	Benefits from decision theory and empirical accuracy	Lacking generalizability
Direct Observation and Rule Estimation	Rules estimated from data	Empirical accuracy	Lacking generalizability; complexity; numerous possible factors to include

Table 2.1: Table of types of ABMs of migration.

Decision-Making Logic in ABMs of Forced Migration Several ABMs of forced migration have been developed, each employing distinct decision-making theories. Here are some examples and their outcomes:

1. Hébert et al. (2018): They utilized the theory of planned behavior from social psychology which resulted in a complex decision-making process, overestimating the number of refugees. This theory may not be suitable for forced migration due to its long-term planning assumptions and the limited applicability of rational behavior (Richey, 2020).
2. Güngör et al. (2022): Their model relied on minimal decision theory, with the choice to flee primarily based on violence tolerance and destination selection focused on proximity. However, its validity is questionable, as it relies on a self-fulfilling prophecy validation process.
3. Sokolowski et al. (2014): This model was based on a decision theory informed by the agent zero construct proposed by Epstein (Epstein, 2014) which can be categorized as a psychology-based theory. However, it overestimated refugee numbers, suggesting potential shortcomings in this approach.
4. Suleimenova et al. (2017): They combined utility maximization theory with empirical data for parameter calibration. This approach is criticized because it assumes rational behavior which is not always the case in a forced migration environment (Richey, 2020). While effective to some extent, criticism about its validity remain (Richey, 2020).
5. Richey (2020): Richey's model employed utility maximization theory while incorporating social network influences, making it more suitable for explaining the behavior of forced migrants since sub-optimal destinations can be chosen. However, there is still room for improvement because agents congregate near the border after a while. This short-coming shouldn't affect the simulation of IDPs because border crossings aren't simulated. Since Richey's model's decision making process includes a decision theory as well as empirical calibration (model type 5) it needs to be re-calibrated to fit the behavior of IDPs in Syria.

Behavior of IDPs in Syria Research into the behavior of IDPs in Syria can guide the changes necessary to apply Richey's decision making process to the simulation of IDPs' movement. It reveals that conflict-prone areas experience higher rates of displacement.

But this is not always the case when analysing at a governorate level since people flee to safer districts located within high-conflict governorates (Doocy et al., 2015). IDPs often move multiple times due to changing battle lines. The majority of IDPs either live with family and friends or rent private accommodations. Others squat in vacant places, or reside in IDP camps (Thibos, 2014). Notably, IDPs tend to congregate in areas with robust infrastructure (Harrison, 2016).

3 Materials And Methods

In the previous chapter, sociological background information pivotal for later defining the model requirements were gathered. In this chapter, key concepts of the reference model which will also play an important role in the requirements analysis are described. Finally, the chosen methods and tools for realising this project are outlined.

3.1 Reference Model Description

The reference model developed by Richey (Richey, 2020) to be adapted is an agent-based model intended to simulate the migration flow of refugees in a host country over a period of 30-90 days. The simulation environment is made up of two components: the location nodes and the refugee agents. This section describes the core concepts of Richey's model design in detail. All the information in this section is extracted from Richey's paper.

3.1.1 Location Nodes

The location nodes are a part of a large, undirected geographical network which is created using geographical centroids of the administrative units of the host country in the desired granularity. In the case of Turkey, the administrative level 2 was chosen resulting in 929 nodes. Each node is connected bidirectionally to every other node in the same country with which it shares a geographic border.

Anchor Location The model includes an anchor location which is a geographical location representing a general direction of refugee movement. In the case of Syrian refugees, a lot of them intend to leave Turkey for Europe so the anchor location should be the coordinates of a city in Europe. The distance from each location node to the anchor location is calculated, represented by the anchor location score of a node.

Location Node Information Each node contains information on the number of refugees at the corresponding location, the anchor location score, the number of conflicts taking place at the location in the simulated time period and the number of camps at that location. A node can also be marked as being a border crossing point from which new refugees will enter the country and with it, the simulation.

Node Score Every location node is assigned a score value which is calculated in each simulation time step. The equation for the score calculation is as follows:

$$E = (p \times w_1) + (l \times w_2) + (c \times w_3) + (d \times (-1) \times w_4) \tag{3.1}$$

The meaning of the variables in the equation is as follows:

p: normalized refugee population at node
l: normalized anchor location score
c: normalized number of camps
d: normalized number of conflicts
w(1-4): weight of each parameter

3.1.2 Refugee Agents

The refugee agents are instances of the refugee class. They are initialized with a reference to their initial node location and two empty lists for friends and kins. These lists are then initialized by creating random social links between agents. After initialization, the lists contain references to the friends and kins of the agents.

Agent Activation In every simulation time step each agent makes the decision to stay at their current location or to leave it. The decision is made by drawing from a probability distribution. The probability of leaving or staying depends on whether their current location hosts a conflict, a camp or none of the above. For each of those cases there is a probability parameter.

Choosing the next destination If an agent has been activated, the next step is to choose their next destination. To do that the agent inspects the locations adjacent to their current location and assigns a desirability score to each candidate location. Finally the node with the highest desirability score is chosen. The equation for the desirability score of a node is as follows:

$$N = k \times w_5 + f \times w_6 + E \tag{3.2}$$

The meaning of the variables in the equation is as follows:

k: normalized number of kins present at the inspected node
f: normalized number of friends present at the inspected node
E: node score
w(5-6): weight of each parameter

Since two agents at the same origin location have different friendship and kinship ties, they can come to different desirability scores for the same location.

Interactions Agents interact with the environment by accessing node scores of neighboring nodes. Agents interact with each other by knowing the geographic location of all their friends and kins.

Moving Agents essentially teleport to the location made, which is possible by the granularity of the simulation environment. The granularity of the simulation environment results in transit distances of no more than a day.

Agent Spawning Agents are spawned at the beginning of the simulation according to the existing refugee population at each administrative unit of the host country. Then new agents are spawned at border crossing nodes. The number of agents spawned at each border crossing point is determined by a parameter, the value of which is established through calibration.

3.1.3 Update Behavior

The node scores and the social networks of agents are updated after every simulation time step.

Node Score Update The node score is recalculated after every simulation time step because its value depends on the number of refugees present at the location and this number changes after each time step as agents leave and enter a location.

Social Network Update Richey makes the assumption that new friendships are formed after having stayed at the same camp. Hence new friendship ties are randomly added between agents at the same camp node.

3.1.4 Input Data

The model incorporates the following data sources:

1. Shapefile with locations of the host country at the desired administrative level used to build geographical network of the simulation.
2. Shapefile of refugee population by province (administrative level 1 is mostly the only data available). If the simulation is to be run with locations of a higher granularity the population numbers are re-projected.
3. Shapefile of refugee camp coordinates data. Used to extract the number of camps at a location.
4. CSV file of conflict events data including conflict coordinates. Used to extract the number of conflicts at a location.

The data is processed with a Python script which extracts the needed information for each location node from the various data sources and joins it all in one data file. The produced data file is then accessed to build a graph. Every row of the data file is used to create a node in the graph. The graph forms the geographical network of the simulation.

3.1.5 Calibration

The probability parameters needed for agent activation, the weights used in the decision making, the initial number range of kins and friends of each agent, the number range of new friendship ties to be formed after every simulation step and the number of agents to be spawned at each border crossing are all calibrated keeping with Gilbert and Troitzsch's methodology (Gilbert, Troitzsch, 2005).

3.2 Agent-based Modelling

Agent-Based Models (ABM) stand as a fitting methodology for the task at hand - predicting the movements of forced migrants. This section serves to illuminate the reasons behind the selection of ABM as the preferred modeling technique, and providing an understanding of its fundamental principles.

Why Agent-Based Modeling for Forced Migration?

1. Modeling Social Interaction and Networks ABM distinguishes itself by allowing for the explicit modeling of social interactions and the resultant social networks (Klabunde, Willekens, 2016). Given that migration decisions are influenced by interpersonal connections, this capability is essential for an accurate representation of real-world migration dynamics.

2. Individual Decision-Making Migration decisions are inherently personal. ABM excels at capturing this individual decision-making process within a simulation environment, offering the flexibility to observe how these autonomous choices drive the evolution of the system over time (Richey, 2020).

3. Heterogeneous Decision-Making In the realm of refugee migration, the decision-making process involves a diverse array of individuals with unique characteristics and considerations. ABM excels at modeling such heterogeneity, allowing individuals represented by agents to interact based on straightforward rules, leading to the emergence of complex behaviors and patterns (Hébert et al., 2018).

Understanding Agent-Based Modeling At the core of ABM lies the concept of software agents, one of the definitions of which is that an agent is an autonomous system situated within an environment. The autonomy of agents grants them the ability to make individual decisions, setting them apart from objects (Padgham, 2004).

Key attributes of software agents as defined by Padgham include:

Attribute	Description
Reactivity	Can respond to changes in their environment
Persistence	Can pursue their goals even when their actions fail
Flexibility	Can concurrently pursue multiple goals
Interactivity	Can interact with other agents
Rationality	Don't pursue contradictory goals

Table 3.1: Table of agents' key attributes.

Critically, the environment in which agents operate is dynamic and unpredictable. The environment may change while agents are in the process of achieving their goals, and agents cannot foresee future environmental states. Agents gather information from their environment to inform their decision-making, and their subsequent actions influence the environment itself (Padgham, 2004).

3.3 The MARS Framework

This section provides an insightful overview of the MARS Framework, an integral component of this study's methodology. Developed by the MARS Group, an academic research project at the Hamburg University of Applied Sciences, the MARS Framework is a MIT-licensed software which provides functionalities for implementing and simulating ABMs (MARS-Group, 2023). The MARS System is written in C Sharp (C#), an object oriented programming language. Central to its design is the hierarchical inheritance structure for the agent type and various environment types, the implementation of which can be extended by the user of the framework.

The MARS Framework has become an essential tool in contemporary research, with applications spanning various domains. In smart city planning, it supports large-scale traffic simulations, enabling a deeper understanding of urban mobility (Clemen et al., 2021a).

In conservation areas, a MARS model has been developed to enhance decision support for Kruger National Park's management activities (Clemen et al., 2021b). Moreover, the MARS Framework's role extends to providing Modeling and Simulation as a Service (MSaaS). It enables distributed and scalable simulations (Hüning et al., 2016) making it an ideal choice for this work, since it aligns perfectly with the reference model by Richey which is designed to be scalable. However, at the time of writing, distributed simulations cannot be run because the MARS cluster is currently not available for usage.

A typical ABM written in MARS features three key concepts. These are:

Agents In MARS, agents take center stage, with each instance of an agent type executed as an independent thread. This concurrent execution of behavior routines by multiple agents allows them to simultaneously interact with and influence both their environment and fellow agents. A MARS agent is characterized by a set of properties encompassing its state, knowledge, and memories. Notably, agents undergo an initialization routine executed once at the start of the simulation, followed by a behavior routine situated within the tick method, which executes during each simulation step.

Entity Entities, in the context of MARS, refer to passive objects that agents can utilize as resources.

Layers A layer is a section of a MARS model's environment. It usually encompasses objects and/or information of the environment that belong to the same class or category. In most cases, layers provide information for agents to interact with. Depending on the model, a layer can be spatially referenced. Such layer types can integrate geodata that enrich a model with real-world georeferenced information. Layers can also be without spatial reference. In that case, their functions tend to be administrative: spawning agents or holding resources for agents during a simulation. Notably, MARS accommodates active layers, which possess behavior routines executed before, during, and after each simulation step. These routines are encapsulated within the layer's pretick, tick, and postick methods. Importantly, all input data of a simulation is integrated into it through layers. The data types that can handled by MARS are: CSV, ASC, Shapefile and GEOJSON (MARS-Group, 2023).

3.4 Data Preparation and Analysis Tool: Jupyter Notebook

In this section, we introduce Jupyter Notebook, a potent tool at the heart of the data preparation and analysis endeavors. Jupyter Notebook is an open-source project under the umbrella of Project Jupyter, offering a versatile platform that empowers interactive data science and scientific computing across diverse programming languages (Project Jupyter, 2023). It allows developers to visualize data sets and test code blocks separately, leading to more flexibility while programming and analysing. Jupyter Notebook is also very open as it facilitates the export to a wide array of data sources (Rauch, 2022).

Within the scope of this work, Jupyter Notebook assumed a central role in several crucial aspects, such as: data cleaning, model output analysis and prediction error calculation. The IPython kernel of the Jupyter Notebook with python modules such as Pandas and Geopandas were used to prepare the data sources used in the model construction and validation as well as to visualize the model outputs using maps and graphs.

4 Requirements

With the core functionalities of the reference model carefully laid out and the sociological backdrop illuminated in the preceding chapters, the requirements for the intended model can be derived from these foundations. Thus this chapter serves as a compass which guides the next steps of model development.

The mind map shown on Figure 4.1 provides an overview of the model requirements detailed in this chapter.

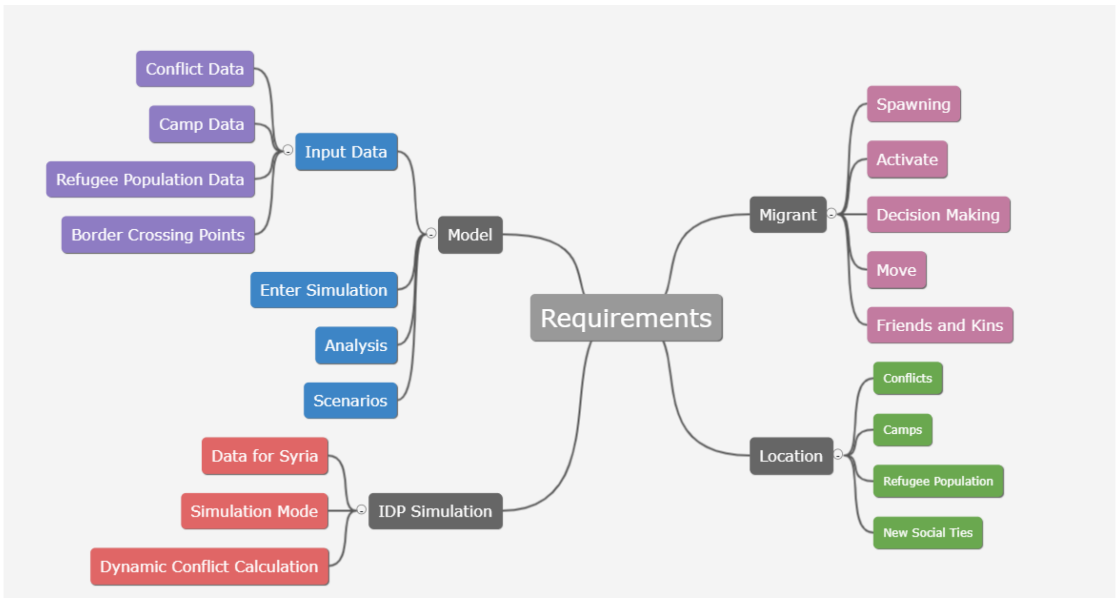


Figure 4.1: Mind map of model requirements

4.1 Replication Requirements

In this section the required functionalities in order to faithfully replicate the reference model are outlined.

4.1.1 Migrant Agent Requirements

Each migrant agent represents a displaced individual who makes decisions in every simulation step.

Requirement 1.1: Spawning At the beginning of the simulation the existing refugee population in each administrative unit is spawned. Then new agents are spawned at border crossing points in each step.

Requirement 1.2: Activation function At each simulation step the agents decide whether to move. To do that they calculate an activation function which takes the number of camps and conflicts at their current location and combine a stochastic component.

Requirement 1.3: Interaction with Locations Agents need to decide where to go. To do that they must evaluate their surroundings, so they are required to know the neighboring nodes of their current location and their scores. After the decision is made, the agents move by changing the current node.

Requirement 1.4: Social network The social network aspect is a pivotal design point of the target model. It is achieved through the agents having randomly selected friends, kins and knowledge of their locations. New friendship ties can be added to the social network.

Requirement 1.5: Decision making To choose their next location, agents assign a desirability value to each neighboring node by calculating a desirability function using the equation of the reference model. Then they move to the node with the highest desirability value.

4.1.2 Locations Requirements

Each location represents an administrative unit of the simulated country. The administrative level of the units can be freely chosen.

Requirement 2.1: Location Score A score is calculated for each location with the equation of the reference model. The score is used by agents to evaluate their surroundings. Since the migrant population parameter of the equation changes after every simulation step, the score is always recalculated afterwards.

Requirement 2.2: Conflicts To calculate the location score, the equation of the reference model requires the number of conflicts taking place at the location in the time period of the simulation.

Requirement 2.3: Camps To calculate the location score, the equation of the reference model requires the number of active camps at the location in the time period of the simulation.

Requirement 2.4: Existing refugee population To calculate the location score, the equation of the reference model requires the number of existing refugees at the location in the beginning of the simulation time period. The refugee population at a location is updated when agents leave it or when agents enter it.

Requirement 2.5: Forming of New Social Ties A random number of friendship ties between random agents at the same location are formed if the location contains a camp.

4.1.3 Model Requirements

Requirement 3.1: Input Data

Real world data is used to create the simulation so the agents can interact with an environment resembling real life conditions. This way the model outputs can be compared to real data of the refugee crisis. All data sources must be in accordance with the chosen simulation time period so real-life conditions can be as close to realistic as possible.

Requirement 3.1.1: Conflict Data Data on conflicts taking place in Turkey that includes date and coordinates is needed to determine the number of conflicts at a location.

Requirement 3.1.2: Refugee Camp Data Data on active refugee camps in Turkey that includes coordinates is needed to determine the number of camps at a location.

Requirement 3.1.3: Refugee Population Data Data on existing refugee population per chosen administrative level is needed to recreate the initial state of the country. If the data is only available at a higher aggregation level, each administrative unit is initialized with an equal portion of the total population of the corresponding higher administrative level unit.

Requirement 3.1.4: Border Crossing Points Names of locations containing border crossing points in Turkey is needed to spawn refugees at the border.

Requirement 3.1.5: Geodata of Turkey Geodata of administrative divisions of the simulated country is needed to create the environment the agents navigate.

Requirement 3.2: Entering and Leaving the Simulation Agents enter the simulation when they cross the border into Turkey but they do not leave the simulation.

Requirement 3.3: Analysis In order to access the simulation results, the model must produce outputs that can be further processed and analyzed. The required outputs are: the refugee population per administrative unit a the beginning of the simulation, the refugee population per administrative unit at the end of the simulation and the routes taken by agents in the course of the simulation, including the number of agents that followed each route.

Requirement 3.4: Parameterization of Scenarios In order to simulate different scenarios it must be possible to change the simulation parameters. These are the simulation time period and length, existing population, camp and conflict data and decision making parameters.

4.2 IDP Simulation Requirements

In order to use the model to predict the migration routes of IDPs the following requirements are added:

Requirement 4.1: Conflict Data in Syria In the simulation of IDPs the agents are making decisions about where to move inside their home country. So the conflict data must contain data points on conflict events taking place in Syria.

Requirement 4.2: IDP camps Refugee camps house people who were forced to leave their home country. Internally Displaced Persons can find shelter in IDP camps. For this reason, data on IDP camps in Syria is required instead of data on refugee camps.

Requirement 4.3: Existing IDP Population Instead of data on the existing refugee population in Turkey, data on the existing IDP population per administrative unit in Syria is required. Since these numbers can be too large for the computational resources available, the proportions of the distribution of IDPs across administrative units can be approximated by using data on IDP arrivals per administrative unit in the time period previous to the simulated time period.

Requirement 4.4: Spawning of IDPs At the beginning of the simulation the sum of IDP arrivals per administrative unit in the previous month or the previous year is spawned. Then new agents are spawned at each administrative unit in each time step. The number of new agents spawned in each step is proportional to the resident population of the administrative unit and the number of conflicts.

Requirement 4.5: Resident Population Data To fulfill the requirement of spawning new agents proportionally to the size of the resident population, data on the resident population per administrative unit of Syria is needed.

Requirement 4.6: Simulation Modes In order to quickly switch between the Turkey simulation and the Syria simulation, the model must have a "mode" parameter which can be switched between Syria and Turkey, dictating the model configuration to be loaded.

Requirement 4.7: Dynamic Conflict Number Calculation Since Syria is an active war zone with evolving battle lines, the number of conflicts in a location should be calculated daily instead of once at the beginning of the simulation, as is specified by the reference model.

5 Conceptual Model

Having collected the model requirements, the next step in the model development process is that of conceptual modeling, which is an activity performed in the analysis phase of a simulation engineering project. Its main purpose is to capture, as faithfully as possible, a relevant part of the real-world domain under consideration, using a well-defined modeling language (Guizzardi, Wagner, 2012). Guizzardi and Wagner propose using a version of UML called Onto-UML, developed previously by Guizzardi (Guizzardi, 2005).

The simulation's implementation design is then built upon the conceptual model of the domain leading to a higher overall quality of the simulation software system (Guizzardi, Wagner, 2012). To create the conceptual model for this work, the tutorial by Guizzardi and Wagner (Guizzardi, Wagner, 2012) is followed.

Why Onto-UML? Conceptual modeling languages should be highly expressive, allowing for the representation of reality truthfully (Guizzardi, 2005). While Unified Modeling Language (UML) and Business Process Modeling Notations (BPMN) are well established notations in the field of software engineering, and are recognized for their expressiveness, their original design purpose was software design rather than conceptual modeling (Specification, 2003). Guizzardi found a number of short-comings of UML for conceptual modeling such as ambiguity (e.g. class associations) and incompleteness (meaning that aspects of the domain can't be represented by UML constructs) (Guizzardi, 2005). Consequently, he developed extensions to the language in order to produce a version of UML class diagrams that is more suitable for conceptual modeling in terms of semantic clarity (Guizzardi, 2005). The produced version is called Onto-UML.

Conceptual Model Figure 5.1 shows the conceptual model of the domain derived from the requirements of the previous chapter. The white elements are the objects. Objects are entities that can be changed by events. Events are represented by the purple elements and can either be triggered externally or they can be action events, which are triggered by the objects.

Textual Description Figure 5.1 shows the developed conceptual model of the domain to be simulated. A migrant is an agent that has a location, kins and friends. A location has zero or many migrants situated in it. A location also can also have zero or many camps and conflicts.

A migrant agent is created by an external event of entering through a border crossing location. They can also make new friends when located at a camp location.

A migrant can decide to leave. A causal law is a sub-process which is triggered by events and can trigger other events (Guizzardi, Wagner, 2012). The causal law related to the decision of leaving is omitted for simplicity. When a migrant decides to leave their current location it triggers the LocationAssessmentLaw which is the sub-process of weighing the candidate locations against each other and choosing the most desirable one.

When a desirable location has been identified the MoveToChosenLocation action event is triggered.

Figure 5.2 shows the previously omitted causal law related to the decision to leave. Migrants are always analysing their current location to make the decision of whether they should move. This action event triggers the DecideToMoveOrNot sub-process which can cause the DecisionToLeave event based on the state of the migrant's current location.

The requirements show that the domain looks slightly different when the migrants are internally displaced persons. Since they stay within their home country's borders, there is no external border crossing event. Instead there is an externally triggered internal displacement event which creates migrants. This change can be seen on figure 5.3.

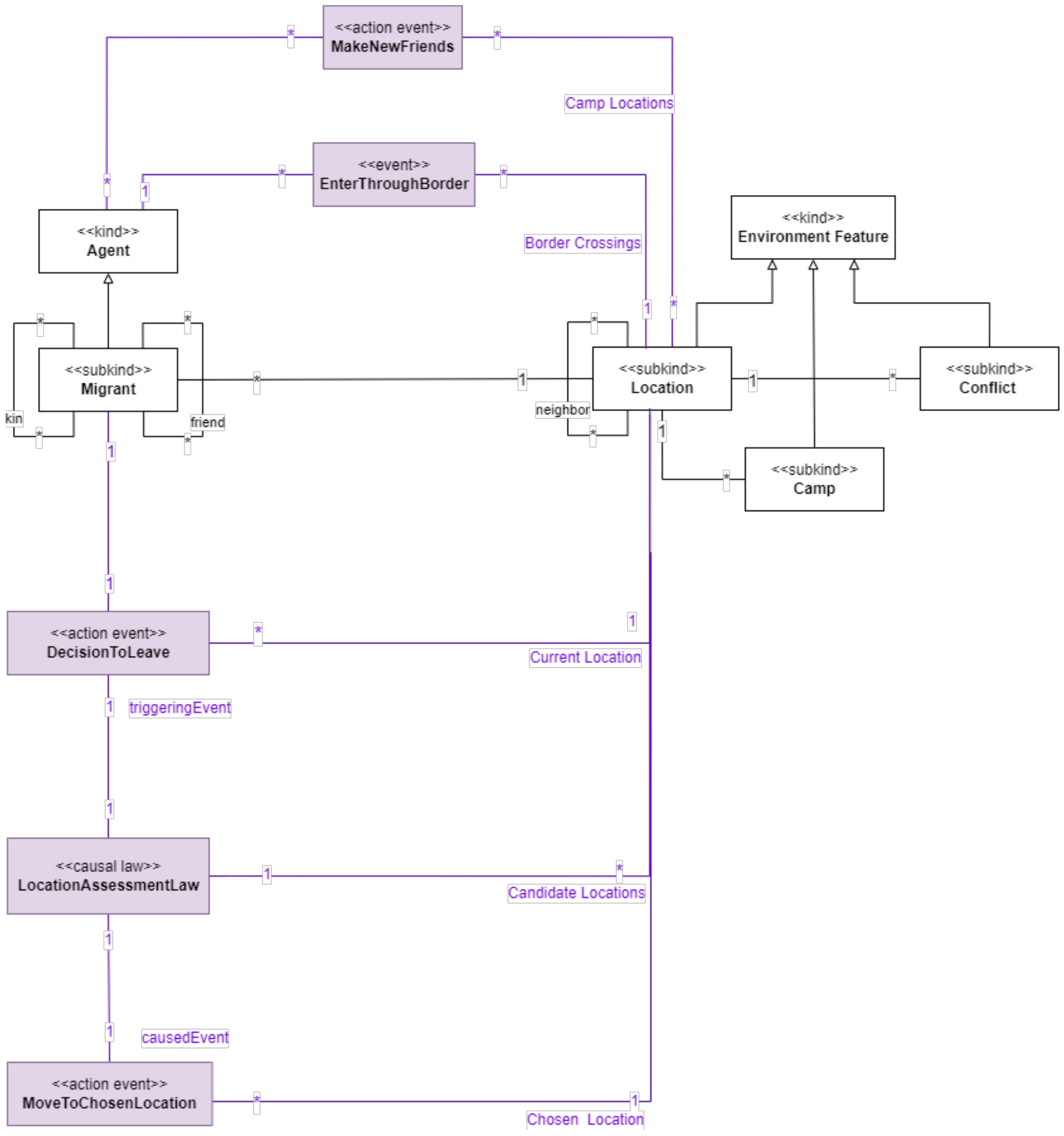


Figure 5.1: Diagram showing the conceptual model of the domain

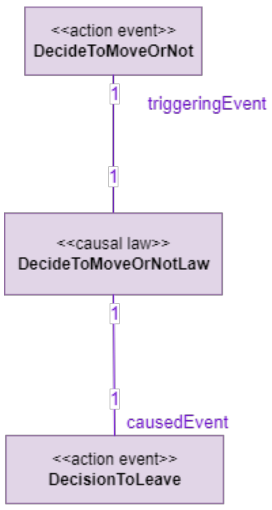


Figure 5.2: Diagram showing the omitted DecisionToLeaveLaw

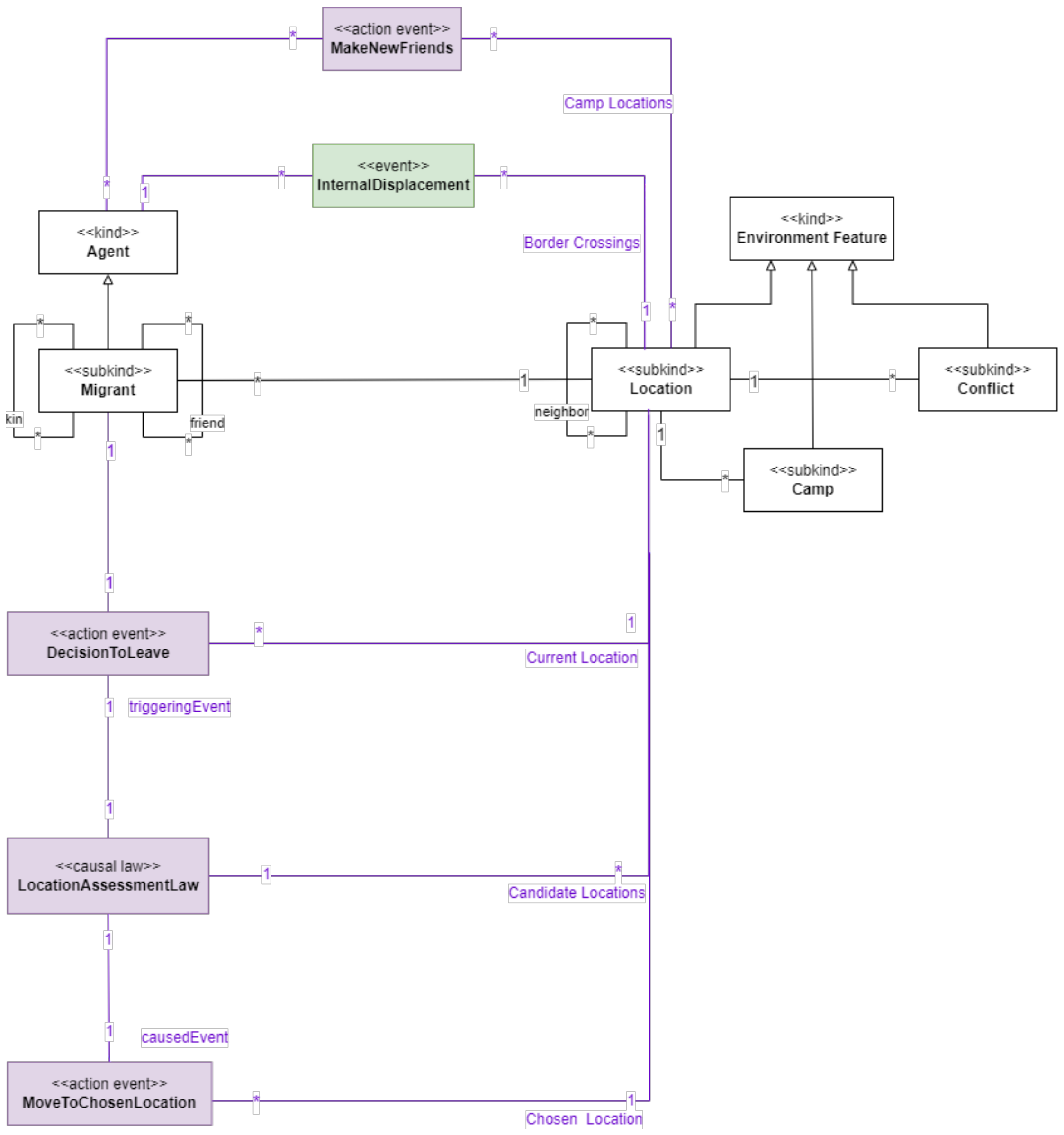


Figure 5.3: Diagram showing the conceptual model including difference in the domain of modeling IDPs

6 Design For Implementation in MARS

In this chapter, a design for the implementation in the MARS framework is derived from the conceptual model shown previously. It is clear from the conceptual model that there are two main components at the center of the model domain. These are the migrants and the locations. For this reason, the simulation model's design is split between these two components. This chapter also outlines the kinds of outputs the model produces, the method of model configuration and the simulation mode functionality.

6.1 Migrant Component

In this section, diagrams displaying the layers and agents of the migrant component as well as their relationships and their connection to the MARS system are shown and explained.

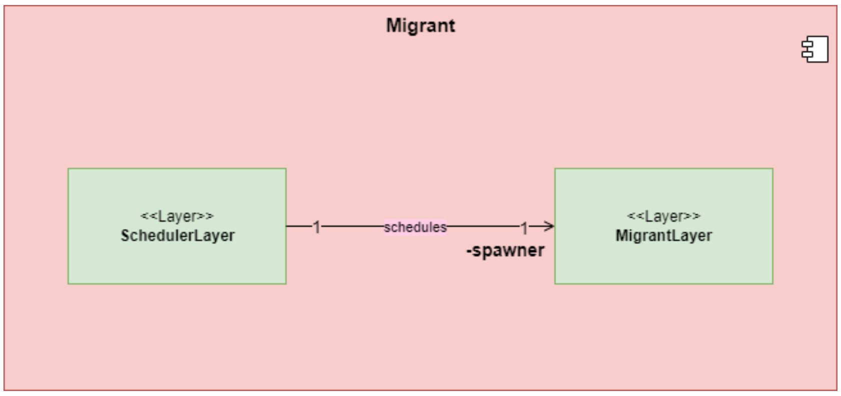


Figure 6.1: Layers in the migrant component

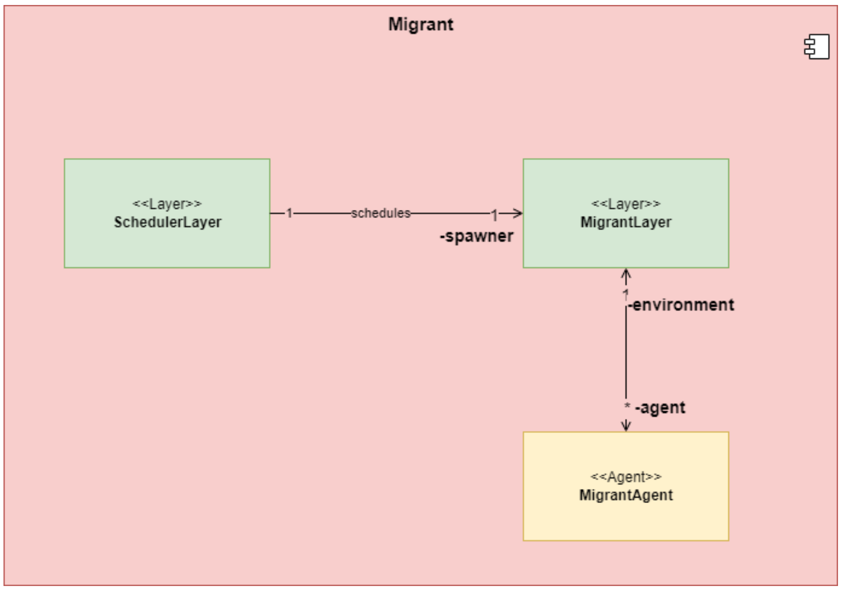


Figure 6.2: Migrant agent in the migrant component

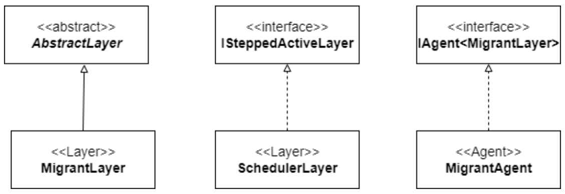


Figure 6.3: Connection of layer types and agent type to the MARS Framework

Layers The migrant component contains two layer types: the SchedulerLayer and the MigrantLayer. The MigrantLayer contains the agent spawning logic. The SchedulerLayer calls the spawning method of the MigrantLayer before each simulation step.

Migrant Agent The MigrantAgent instances' initialization methods are called by the MigrantLayer. The MigrantAgent contains the social network functionality and all the agent behavior logic including the decision to leave or to stay, the assessment of neighboring location and moving. The movement necessitates the access to an environment provided by the MigrantLayer. The behavior defined in the MigrantAgent is executed in every simulation time step.

Connection to the MARS system Figure 6.3 shows that the MigrantLayer inherits from the AbstractLayer class of the MARS Framework providing it only with adminis-

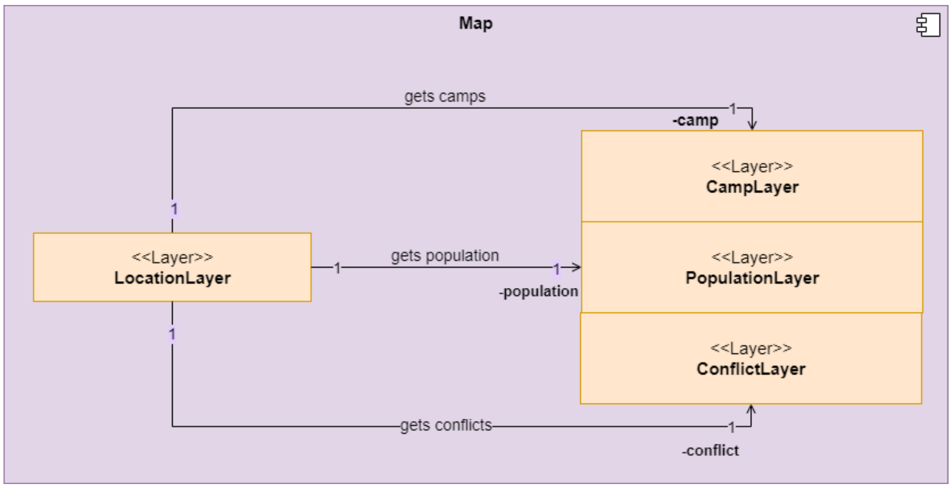


Figure 6.4: Layers in the map component

trative functionalities to spawn agents and to hold resources for agents (MARS-Group, 2023), in this case it holds the environment the agents move through.

The SchedulerLayer implements the ISteppedActiveLayer interface making it an active layer which has a behavior routine that can be executed before, during of after each simulation step. In the case of the SchedulerLayer it calls the spawning function of the MigrantLayer before every step.

The MigrantAgent implements the IAgent<LayerType> interface of the MARS Framework which allows the agent class to be instanciated by the layer specified in the LayerType parameter, in this case the MigrantLayer. It also facilities the implementation of a behavior routine which is executed in each simulation time step (MARS-Group, 2023).

6.2 Map

In this section diagrams displaying the layers and vector features of the map component, as well as their relationships and their connection to the MARS system are explained.

Layers The figure 6.4 shows that the map component contains four layer types: the CampLayer, the ConflictLayer, the PopulationLayer and the LocationLayer.

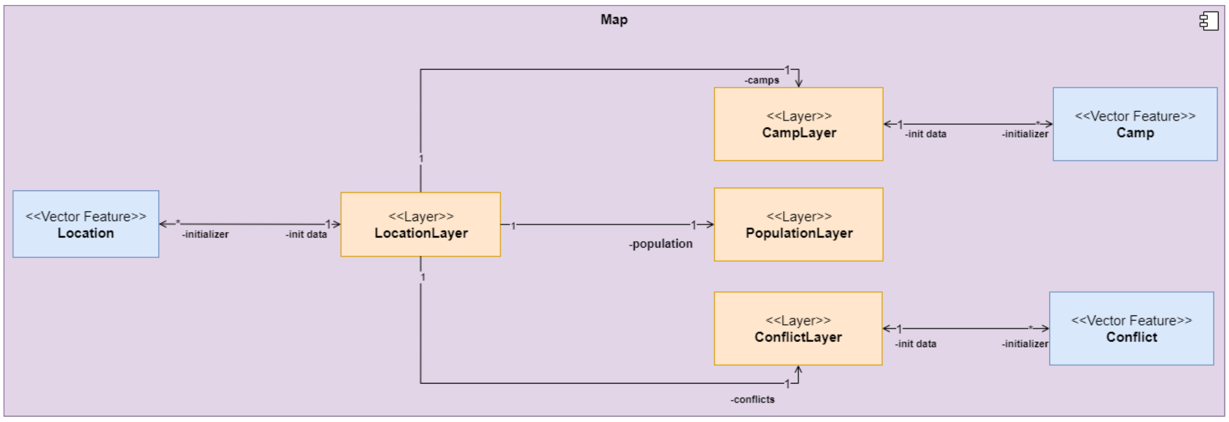


Figure 6.5: Vector Features in the map component

The CampLayer is used to integrate geodata of refugee or IDP camps. It makes this data available to the LocationLayer for further processing.

Likewise the ConflictLayer is used to integrate geodata of conflict events.

The PopulationLayer is used to integrate data on the resident population of Syria per administrative unit in order to spawn new agents in the course of the simulation.

The LocationLayer is used to integrate geo-data of the simulated country and map its administrative units, thus modeling the simulation environment.

Vector Features Vector Features are objects mapped by georeferenced layers. They contain geometries (points, lines, areas) which represent their real-world spatial extent (MARS-Group, 2023).

Figure 6.5 shows that there are three types of vector features in the map component. Each instance of a vector feature is instantiated from a data point in the geodata integrated into its corresponding layer. Each instance is also initialized using the attributes from the corresponding data row.

Furthermore the Location vector feature accesses the data provided by the CampLayer and the ConflictLayer to enrich each Location instance with the number of camps and conflict events inside its borders.

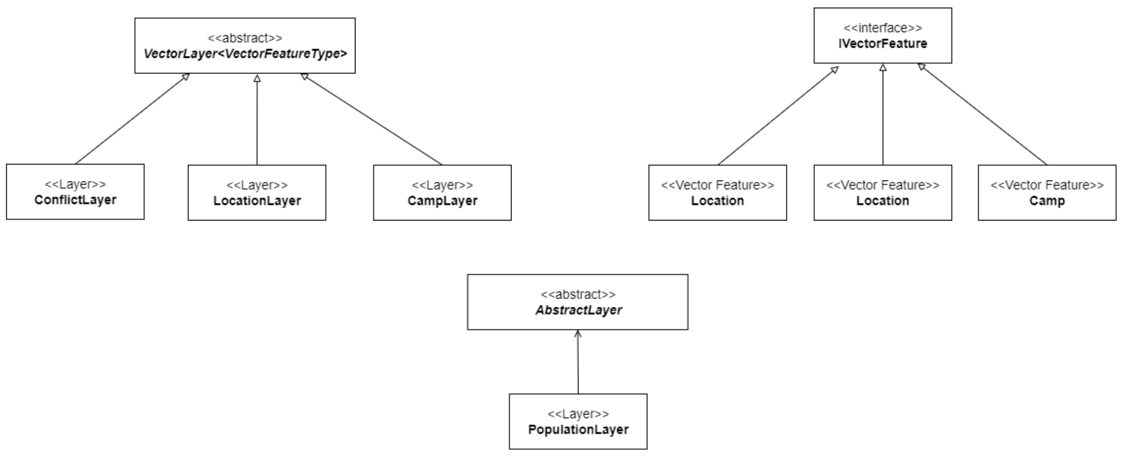


Figure 6.6: Connection of layer types and vector feature types to the MARS Framework

Connection to the MARS system Figure 6.6 shows that all layers of the map component except the PopulationLayer inherit from the VectorLayer<VectorFeatureType> of the MARS Framework. The type parameter of the VectorLayer is set according to the Vector Feature class designated to the layer. For example the LocationLayer inherits from the VectorLayer with the type parameter set to Location.

Inheriting from the VectorLayer class allows the layers to be georeferenced in MARS.

The PopulationLayer inherits from the AbstractLayer which facilitates the holding of resources stemming from input data, in this case, the resident population of each location.

6.3 Relationship Between The Components

In this section a diagram displaying the associations across the migrant and location components (Fig. 6.7) is shown and explained.

The associations across the two components are marked in red in figure 6.7.

MigrantLayer - LocationLayer association The environment the agents move through is provided to the MigrantLayer by the LocationLayer. The LocationLayer also delivers references to Location instances.

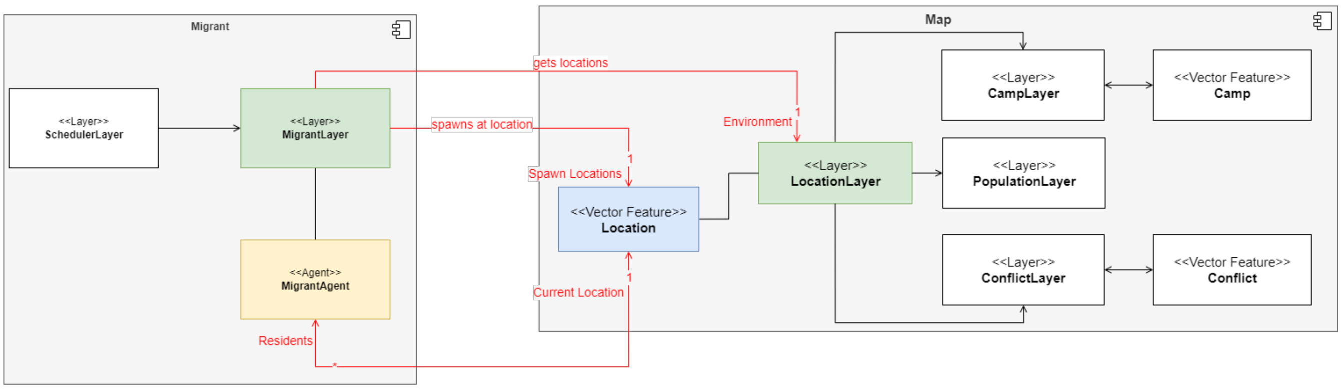


Figure 6.7: Relationship between the migrant and the map components

MigrantLayer - Location association The MigrantLayer gets a location reference from the LocationLayer and utilizes it to spawn agents at that location.

MigrantAgent - Location association The Migrant Agent has a reference to a location instance representing its current location. Locations can explore the environment agents move through in order to access the agents positions, thus being able to filter by the agents residing within its borders.

6.4 Data Integration

In this section a diagram displaying where the data sets required as input for the model are integrated.

In the MARS Framework the data input of the simulation is made available through layers. Each layer can provide access to one data source.

Figure 6.8 shows which layers are used to access the data sources required for the simulation.

To use the data in the simulation the MARS environment reads the columns of the data source by name. Table 6.1 provides an overview of which fields are required for each data source.

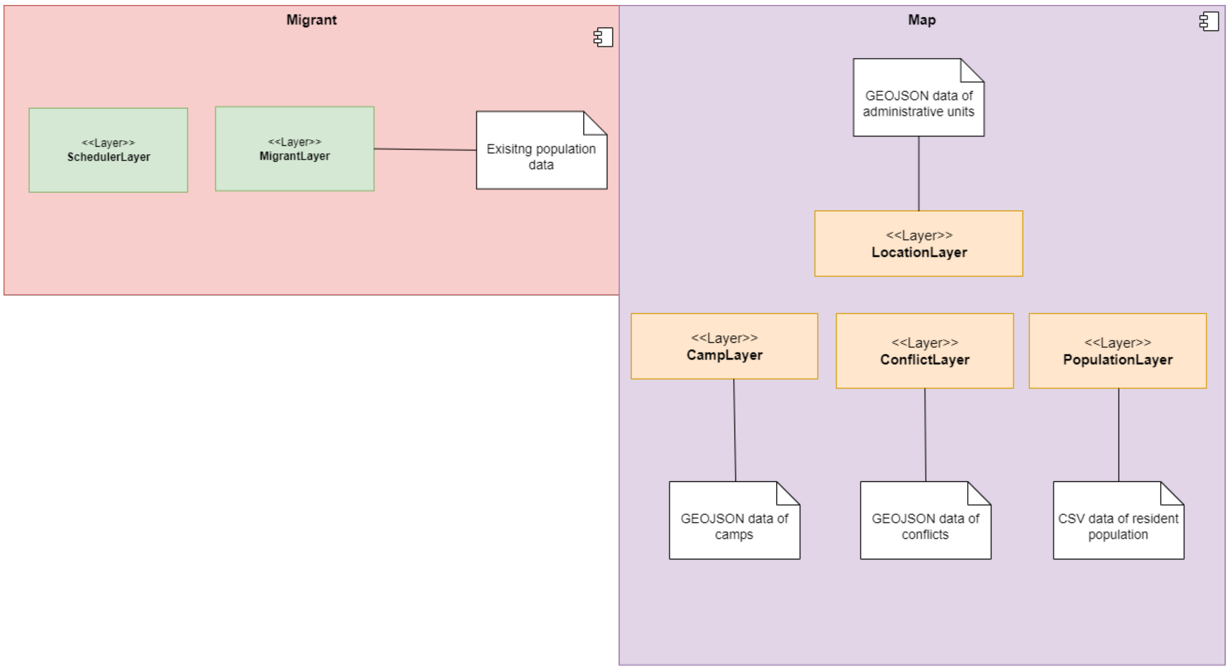


Figure 6.8: Required data sets integrated through MARS layers

Data Source	Required Fields	Description
Existing migrant population data	Region IDPs	Location name Number of migrants
Location Data	ADMX_-EN Geometry	Location name, X is the administrative level Polygon geometry
Camp data	Geometry	Point geometry
Conflict data	Month Geometry	month number of conflict date Point geometry
Population data	Region Total_population	Location name at adm level 3 Number

Table 6.1: Table of required fields of data sources.

6.5 Model Output

Apart from the two main components there is a static class called Validation which collects information from the main components to generate three important output files in CSV format. Table 6.2 gives on overview of the output files.

File Name	Description
InitPop.csv	Syrian Refugee or IDP population of each region of the country at the beginning of the simulation
RefPop.csv	Refugee or IDP population of each administrative of the country after the simulation run.
Routes.csv	Routes, defined by origin and destination, taken by the agents during the simulation, and the number of agents who took each route

Table 6.2: Table of model output files.

When running the simulation of IDPs, the population output files are available at different aggregations. There is one file for each of the administrative levels 1 to 3.

6.6 Model Configuration

The configuration of the model is a crucial aspect that defines the dynamics of the simulation and assigns values to behavior parameters. This section outlines the various components of the model configuration, including global simulation parameters, layer properties and agent properties.

The MARS Framework allows the storing of the model configuration in a JSON file, which provides a structured way to set the simulation's parameters and inputs. There are two JSON configuration files made available, one of which is loaded before the execution of the simulation. One file contains the configuration necessary to simulate the movement of Syrian refugees in Turkey. The other contains the necessary configuration to simulate the movement of IDPs in Syria.

Global Parameters The global parameters section in the configuration file focuses on setting overarching properties that govern the entire simulation. One of the fundamental global parameters of the model is the execution time. In the context of the MARS framework, the execution time can be configured in a real-time based manner. This means that the simulation is defined by specifying a start date and an end date, along with the time unit used for simulation steps. This ensures that the simulation progresses in alignment with real-world time intervals.

Layer Properties Configuration The remaining of the configuration file is divided into two main sections. One of them is the layer properties configuration. This sections is divided in six subsections, one for each layer of the model. These subsections include the following parameters:

Parameter Type	Description
File	Data source file located in the 'resources' directory
Number of New Agents	Number to spawn at each step
Weights	Weights used in the calculation of node scores
Social Network Parameters	Number of new social ties to create

Table 6.3: Table of parameter types of the layer configuration.

Agent Properties Configuration The other main section of the configuration is the agent properties configuration. In it the behavior governing parameters of the agents are set. It includes settings such as:

Parameter Type	Description
Behavior Parameters	Guide the impact of social ties on agents' behavior
Move Probabilities	Define the probability of an agent being activated

Table 6.4: Table of parameter types of the agent configuration.

By utilizing the MARS framework and the structured JSON configuration file option, the model achieves a flexible design. This configuration process enables modelers to simulate various scenarios and to easily switch between use cases by choosing the corresponding configuration file.

6.6.1 Simulation Mode

The executable Main method of the simulation contains a mode variable which can be switched between "Turkey" and "Syria". When in Turkey mode, the configuration file containing the input data and parameter configuration for Turkey is loaded into the model. When in Syria mode, the configuration file containing the input data and parameter configuration for Syria is loaded.

7 Implementation Details

This chapter describes the implementation of the layers, vector features and agent introduced in the previous chapter. The descriptions are based on technical UML-diagrams, from which the program code was derived. Helper methods have been omitted for simplicity.

7.1 Migrant Component

7.1.1 MigrantLayer Implementation

The MigrantLayer's main function is to spawn agents. The data structure AgentDistributionData (fig 7.1) provides access to the input data necessary to distribute the agents across the locations according to real-world numbers at the beginning of the simulation. This is done by the InitAgents method.

To fulfill the different requirements of new agents entering the simulation at each step, the MigrantLayer defines two methods. The SpawnNewRefs method is called by the SchedulerLayer when the simulation in run in the Turkey mode i.e. to simulate Syrian refugees in Turkey. Likewise the SpawnNewIDPs method is called in the Syria mode.

When spawning agents as IDPs, the instance variable NumAgentsToSpawn provides a guideline for the total number of new agents spawned per simulation time step. Which fraction of this number is spawned at each location depends on the resident population of the location and the number of conflicts there.

After all agents have been successfully spawned, the InitSocialNetwork method can be called which iterates over all agents and initiates the formation of their social links.

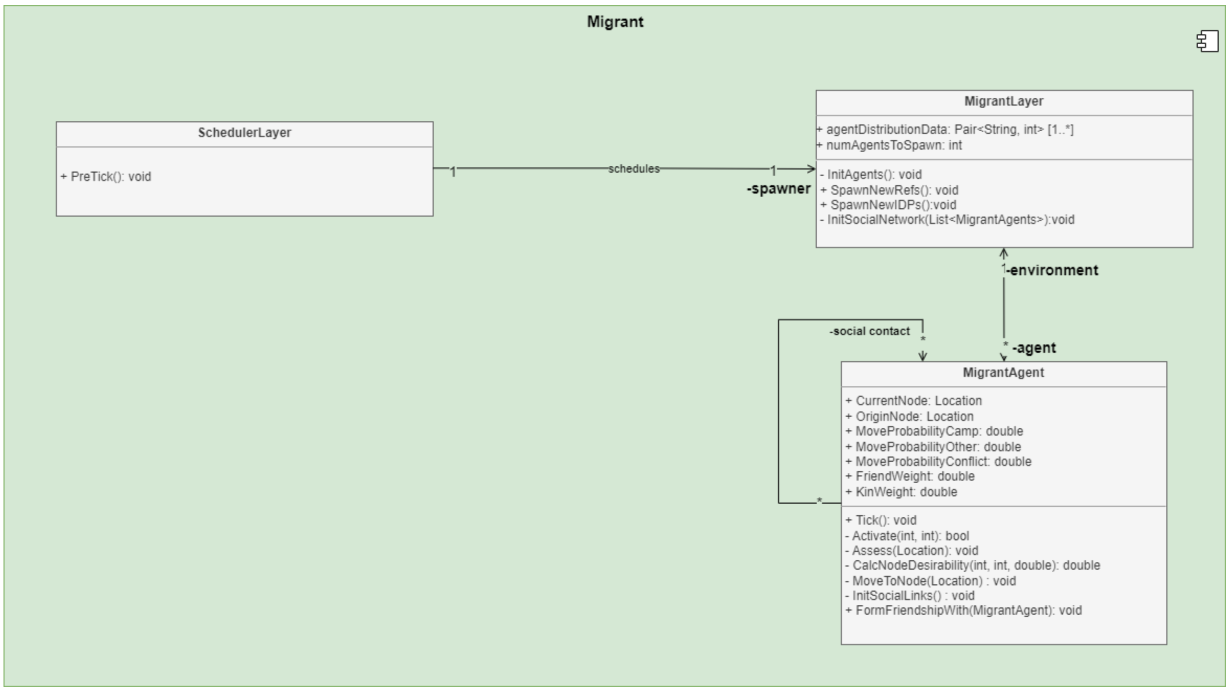

Migrant
SchedulerLayer
+ PreTick(): void
1
schedules
1
-spawner
MigrantLayer
+ agentDistributionData: Pair<String, int> [1..*]
+ numAgentsToSpawn: int
- InitAgents(): void
+ SpawnNewRefs(): void
+ SpawnNewIDPs():void
- InitSocialNetwork(List<MigrantAgents>):void
1 -environment
-social contact
*
* -agent
*
MigrantAgent
+ CurrentNode: Location
+ OriginNode: Location
+ MoveProbabilityCamp: double
+ MoveProbabilityOther: double
+ MoveProbabilityConflict: double
+ FriendWeight: double
+ KinWeight: double
+ Tick(): void
- Activate(int, int): bool
- Assess(Location): void
- CalcNodeDesirability(int, int, double): double
- MoveToNode(Location) : void
- InitSocialLinks() : void
+ FormFriendshipWith(MigrantAgent): void

Figure 7.1: Class diagram showing the implementation of the layer and agent classes of the migrant component

7.1.2 MigrantAgent Implementation

The agents' social network is implemented as sets of kins and friends which contain references to other agents. All agents have a reference to their current location. Consequently, all agents have access to the current location of their friends and kins.

The OriginNode instance variable is used to create the model output of routes mentioned in the previous chapter. The probabilities and weights variables are used in the decision making process. Their values are set through the configuration file also introduced in the previous chapter.

The agents' behavior is contained within the Tick method which is implemented from the IAgent interface.

Figure 7.2 shows the sequence diagram describing the behavior logic contained in the Tick method:

Firstly the agent gets the number of conflicts and camps at their current node, as these parameters are needed to decide whether to leave or not. This decision is made in the

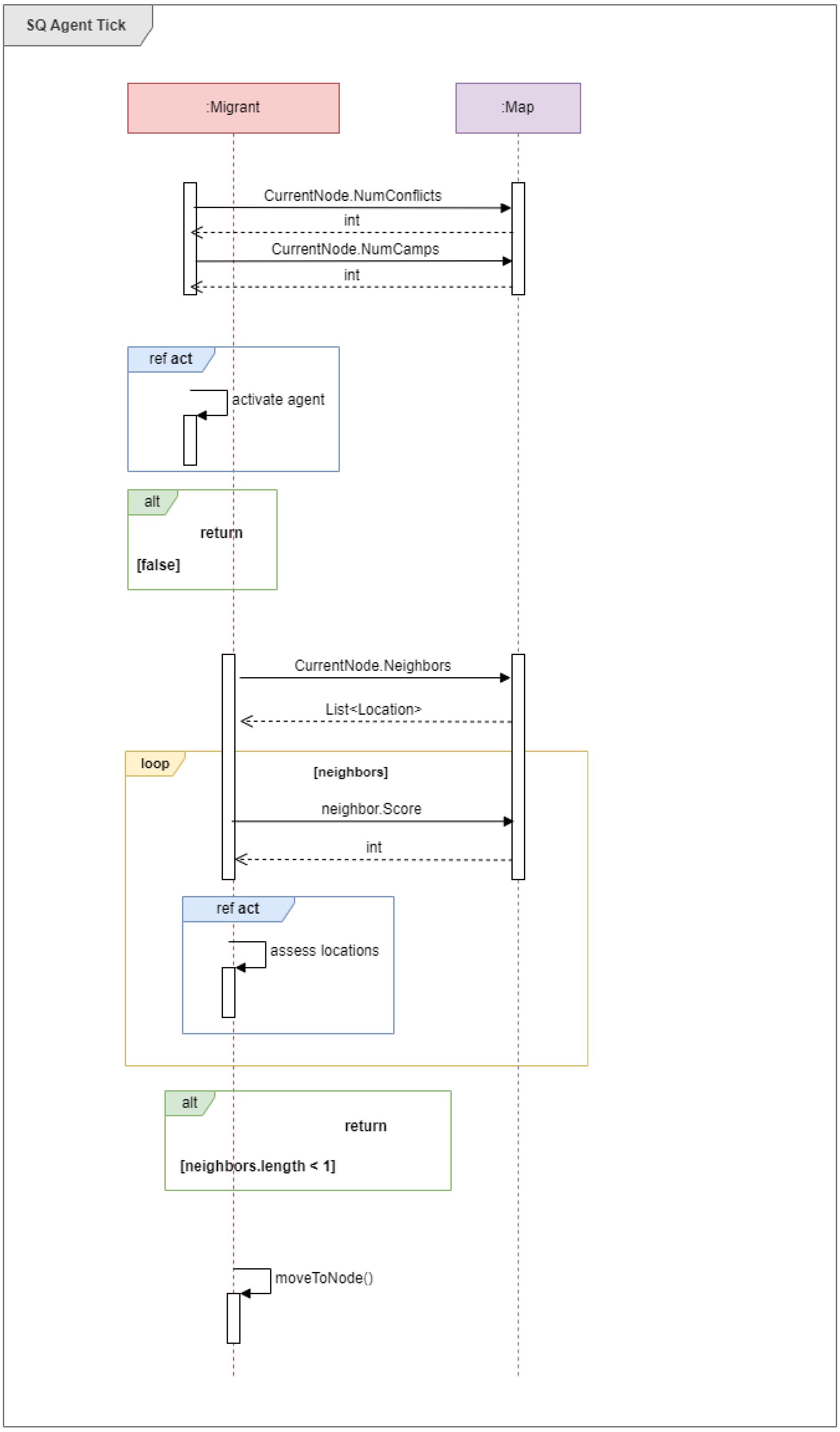


Figure 7.2: Sequence diagram showing the interactions between the migrant agent and the map locations in the Tick method

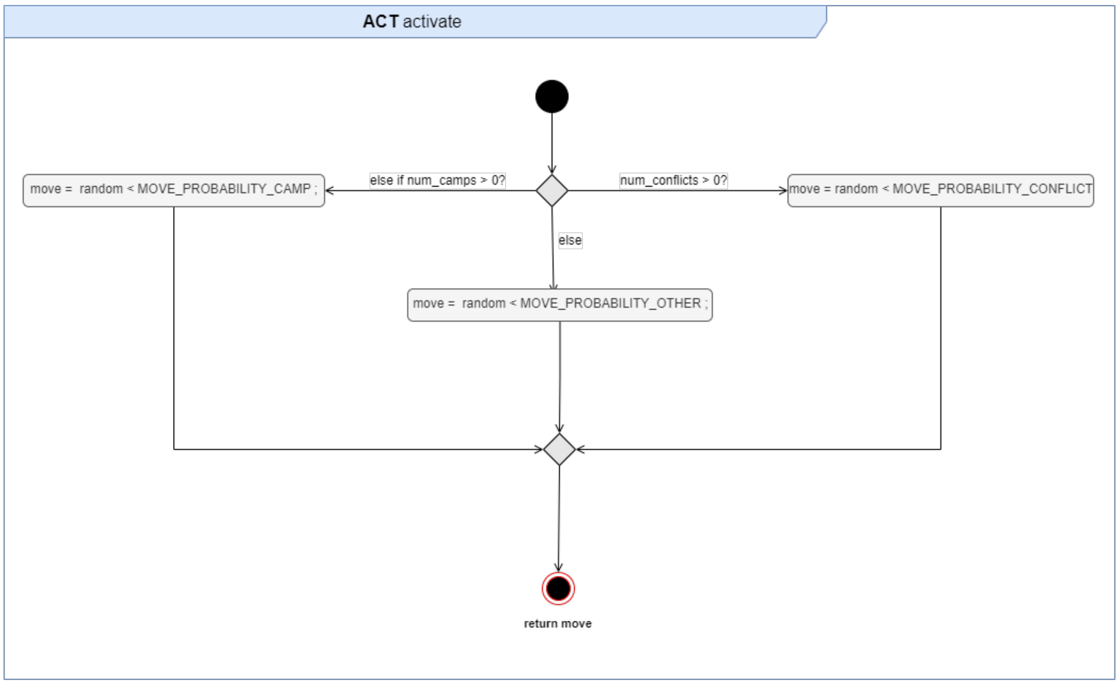

ACT activate
else if num_camps > 0?
num_conflicts > 0?
else
move = random < MOVE_PROBABILITY_CAMP ;
move = random < MOVE_PROBABILITY_CONFLICT
move = random < MOVE_PROBABILITY_OTHER ;
return move

Figure 7.3: Activity diagram showing the decision logic of agent activation

activate method. Details on the activation logic are displayed in the activity diagram in figure 7.3.

Whether an agent leaves a location depends primarily on the presence of conflicts there. If an agent is activated, the neighbors of the current location are assessed and the neighboring location deemed most desirable is chosen as the next destination. Details on the location assessment logic are displayed in the activity diagram in figure 7.4.

The method CalcNodeDesirability simply calculates the equation used in the reference model. This equation is mentioned in the Materials and Methods chapter.

7.2 Map Component

7.2.1 Camps and Conflicts

The CampLayer collects the coordinates of camps and makes it accessible, so locations can count the number of camps within its borders.

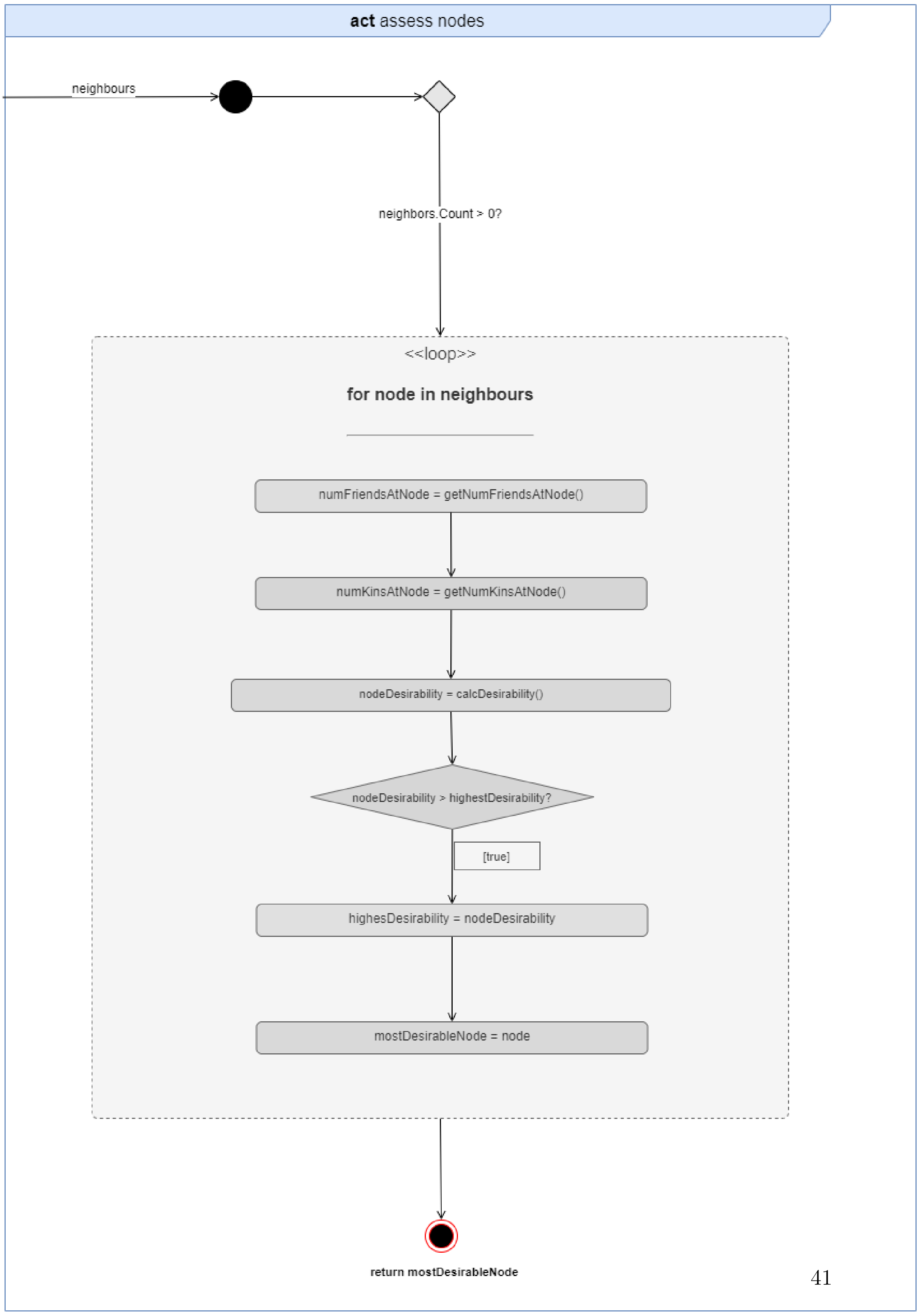

act assess nodes
neighbours
neighbors.Count > 0?
<<loop>>
for node in neighbours
numFriendsAtNode = getNumFriendsAtNode()
numKinsAtNode = getNumKinsAtNode()
nodeDesirability = calcDesirability()
nodeDesirability > highestDesirability?
[true]
highesDesirability = nodeDesirability
mostDesirableNode = node
return mostDesirableNode

Figure 7.4: Activity diagram showing the algorithm of location assessment

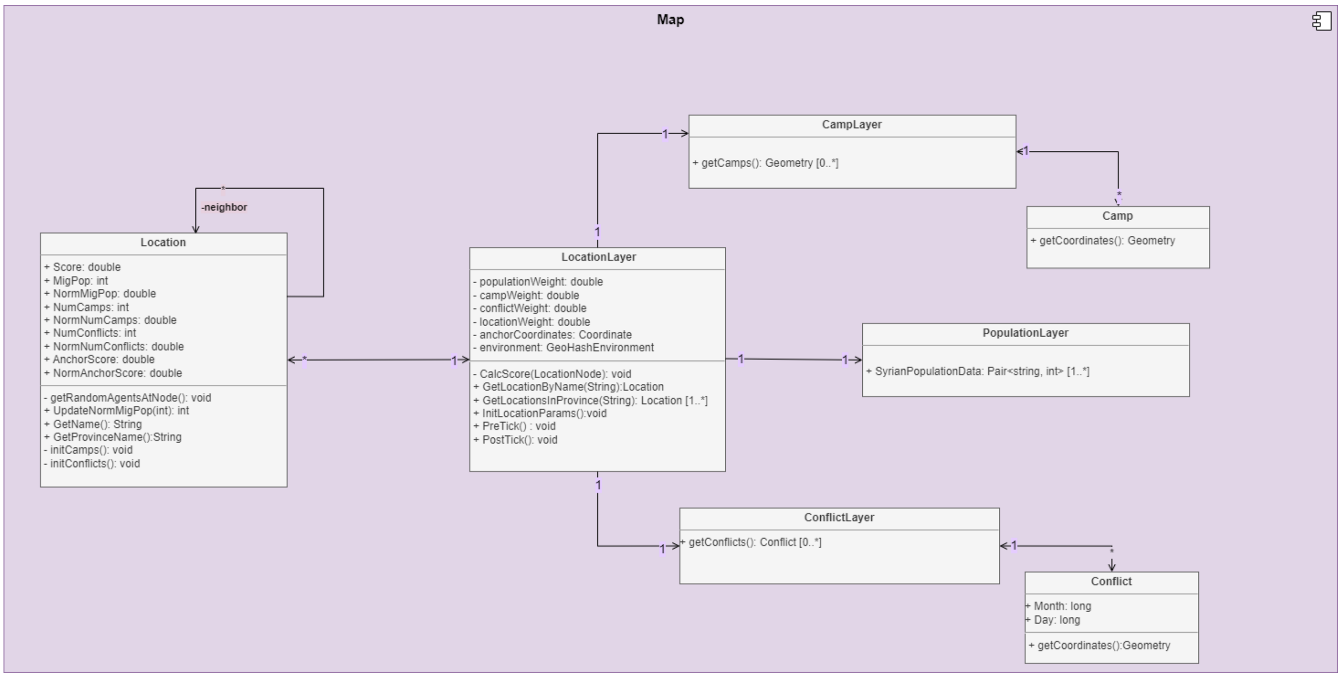

Map
CampLayer
+ getCamps(): Geometry [0..*]
Camp
+ getCoordinates(): Geometry
-neighbor
Location
+ Score: double
+ MigPop: int
+ NormMigPop: double
+ NumCamps: int
+ NormNumCamps: double
+ NumConflicts: int
+ NormNumConflicts: double
+ AnchorScore: double
+ NormAnchorScore: double
- getRandomAgentsAtNode(): void
+ UpdateNormMigPop(int): int
+ GetName(): String
+ GetProvinceName():String
- initCamps(): void
- initConflicts(): void
LocationLayer
- populationWeight: double
- campWeight: double
- conflictWeight: double
- locationWeight: double
- anchorCoordinates: Coordinate
- environment: GeoHashEnvironment
- CalcScore(LocationNode): void
+ GetLocationByName(String):Location
+ GetLocationsInProvince(String): Location [1..*]
+ InitLocationParams():void
+ PreTick() : void
+ PostTick(): void
PopulationLayer
+ SyrianPopulationData: Pair<string, int> [1..*]
ConflictLayer
+ getConflicts(): Conflict [0..*]
Conflict
+ Month: long
+ Day: long
+ getCoordinates():Geometry
1
*

Figure 7.5: Class diagram showing the implementation of the layer and vector feature classes of the map component

The ConflictLayer makes conflict vector feature instances accessible directly, so locations can not only count the number of conflict events within its borders, they can also check that the events are taking place at the same time as the simulation.

7.2.2 PopulationLayer

The PopulationLayer's only function is to hold information on the resident population of each Syrian administrative unit. This information is extracted from the population data file during the initialization of the PopulationLayer and referenced by the SyriaPopulationData instance variable.

7.2.3 LocationLayer

Environment The agents move through an environment, their current positions are made accessible to the map component through this same environment. It is first instantiated at the LocationLayer and provided to the agents by the MigrantLayer.

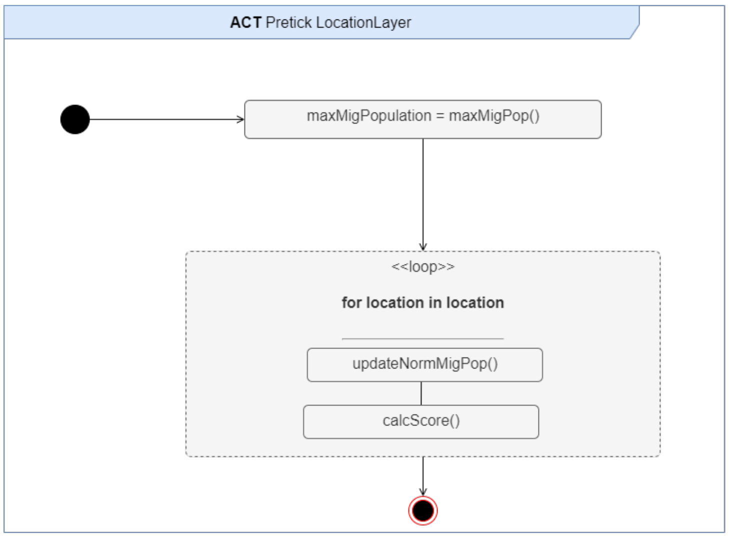


Figure 7.6: Activity diagram showing the behavior of the LocationLayer before every simulation step

Initialization of Location Parameters The LocationLayer calls the InitLocation-Params method to initialize the locations parameters needed for the decision making process. The method normalizes the number of conflicts, the number of camps and the anchor score of the locations. The anchor coordinate variable contains the coordinates of a general location migrants are moving to, and is needed to calculate the anchor score. In this case, London was chosen but the exact location doesn't make a significant difference (Richey, 2020).

Moreover, the neighbors of each location are determined.

Calculating Location Scores The LocationLayer re-calculates the location scores (CalcScore method) before every simulation step (PreTick method). It should not be calculated after the simulation step, so the first step is run with valid score values.

The calculation of location scores necessitates the weight parameters. The values of weights as well as the anchor coordinates are set in the simulation configuration file.

Details on the calculation of node scores are displayed on the activity diagram (fig 7.6).

The reason the location scores need to be re-calculated in the first place is that it is dependant on the migrant population at the location. Since this number changes after every step, the normalized population number has to be calculated again.

Updating The Social Network The LocationLayer initiates the updating of social networks by calling the locations which contain camps. The method called chooses two agents at random at the location and makes a call to form a friendship tie between both of them. This procedure needs to be executed after each simulation step, which is why it is implemented in the PostTick method.

The sequence diagram in figure 7.7 displays the behavior logic described above.

Access to Locations and Locations in a Province The LocationLayer has two methods which return references to locations:

The method GetLocationByName returns the location whose name corresponds to the call parameter. If there is no location instance to the name an exception is thrown.

The method GetLocationsInProvince returns a list of locations situated in the province given in the call parameter. A province is defined as the administrative level 1. The locations returned are the administrative units in the level input in the configuration file. If the model is configured to run in the administrative level 1, then the location whose name corresponds to the call parameter is returned. If the province cannot be found, the system tries to find a district or sub-district (administrative levels 1 and 3) whose name corresponds to the call parameter. If one isn't found, an exception is thrown. This method contributes to the flexibility of granularity of the administrative level chosen for the simulation. It does so by allowing agents to be spawned at the beginning of the simulation at locations corresponding to lower aggregate administrative units than the available existing population data.

7.2.4 Location Vector Feature

Initialization At the time of initialization each location instance extracts the name of the administrative unit it represents and its province name from the data row provided by the LocationLayer. These names can be queried by calling the method GetName and GetProvince name.

Furthermore the anchor score is set at initialization by calculating the distance of the location to the anchor location. The methods InitConflicts and InitCamps iterate over the coordinates of all camp and conflict instances and counts the number of camps and conflicts within the location's borders.

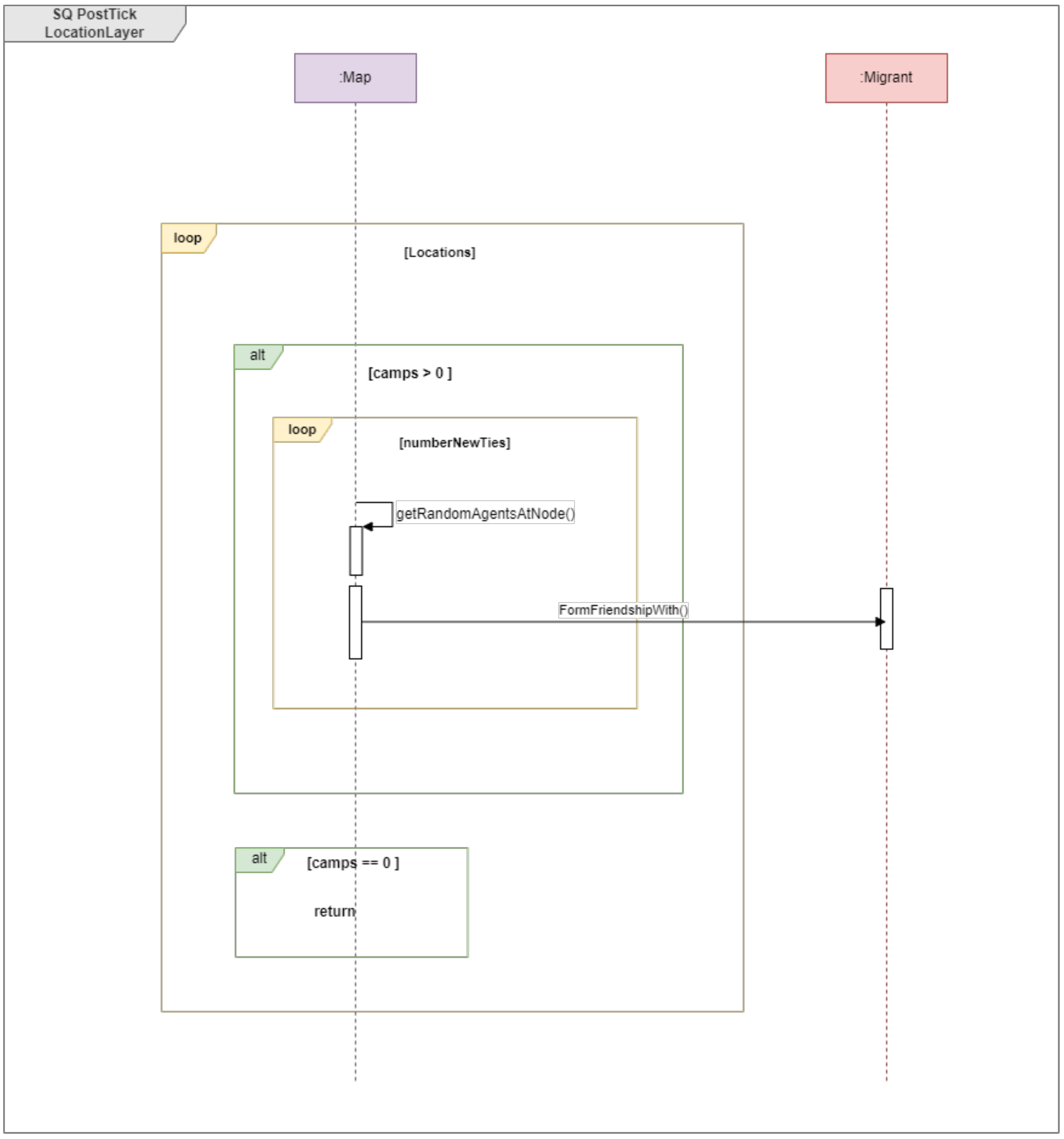


Figure 7.7: Sequence diagram showing the method calls made to update the social networks of agents after every step

The conflicts are further filtered by the events taking place at the time period of the simulation.

Updating The Social Network The method getRandomAgentsAtNode is called by the LocationLayer if the location contains a camp. In this method the environment providing the positions of all agents is accessed and the agents situated at the location are filtered. Then two agents are picked at random to become friends.

8 Data Sourcing and Preparation

In the preceding chapters the necessary data sources for the model were established, the methodology for integrating these data sources into the simulation environment was outlined and the manner in which the data was utilized in the model was touched on. This chapter delves into the aspect of data sourcing and preparation, detailing the path taken to acquire and preparing the input data.

8.1 Data Sourcing

Data Sources for the Turkey Simulation For the Turkey simulation all the data sources required to replicate the reference model were retrieved from its GitHub repository (`https://github.com/mrichey17/mig/tree/master`). The advantage is that the data aligns with the reference model, ensuring consistency and comparability. All data sets were available as shapefiles, with the exception of the conflict data, which is in CSV format.

Data Sources for the Syrian IDP Simulation For the simulation concerning Syrian IDPs, data was sought from various sources. The data origins include the Humanitarian Data Exchange (HDX, 2023), which provided geospatial data pertaining to Syria and its IDP camps, as well as information on IDP populations and resident populations across Syrian administrative units. Data on conflict events within Syria was sourced from the Armed Conflict Location and Event Data Project (ACLED, 2023).

It's important to acknowledge the challenges faced when sourcing data for the Syrian IDP simulation. Notably, the scarcity of data pertaining to the same reference time period presented a challenge. The newest available IDP camp data is from early 2016, and the oldest IDP population data is from the end of 2016. The only accessible resident population data dates back to 2004. Nevertheless, the conflict data is meticulously detailed, and the IDP camp data is verified through satellite imagery. Moreover, the IDP population data consists of arrival estimates per Syrian governorate and is sourced from a collective of multiple humanitarian actors.

8.2 Data Preparation

Data Preparation for the Turkey Simulation The data preparation for the Turkey simulation involved converting the shapefiles to GEOJSON out of personal preference, as both file types are compatible with the MARS Framework. This conversion was carried out using QGIS.

The conflict data, initially in CSV format, was also converted to GEOJSON using QGIS, so it is compatible with the ConflictLayer, which can only receive georeferenced data as input.

IDP Simulation Data Preparation Preparing data for the Syrian IDP simulation was more intricate:

The IDP population data file was converted from Excel to CSV to meet the MARS Framework's file type requirements. Subsequently, data attributes were extracted and cleaned to remove unwanted characters. This process was facilitated using Jupyter Notebook.

The data on conflict events in Syria was converted to GEOJSON, but prior to that, the day and month attributes were extracted for each conflict event using Jupyter Notebook. This step was taken to facilitate dynamic conflict calculations.

Preparing the Syrian resident population data was the most challenging. This data also underwent format conversion from Excel to CSV. Discrepancies in location names, attributed to variations in English spellings, presented a challenge. To harmonize district names between the Syrian resident population data and Syria geodata, a data join was performed using Jupyter Notebook. Any unmatched data points were rectified by aligning name spellings.

9 Verification And Validation

Having meticulously explored the intricacies of model implementation and the preparation of integrated data sources in preceding chapters, now the model verification and validation processes are presented and discussed.

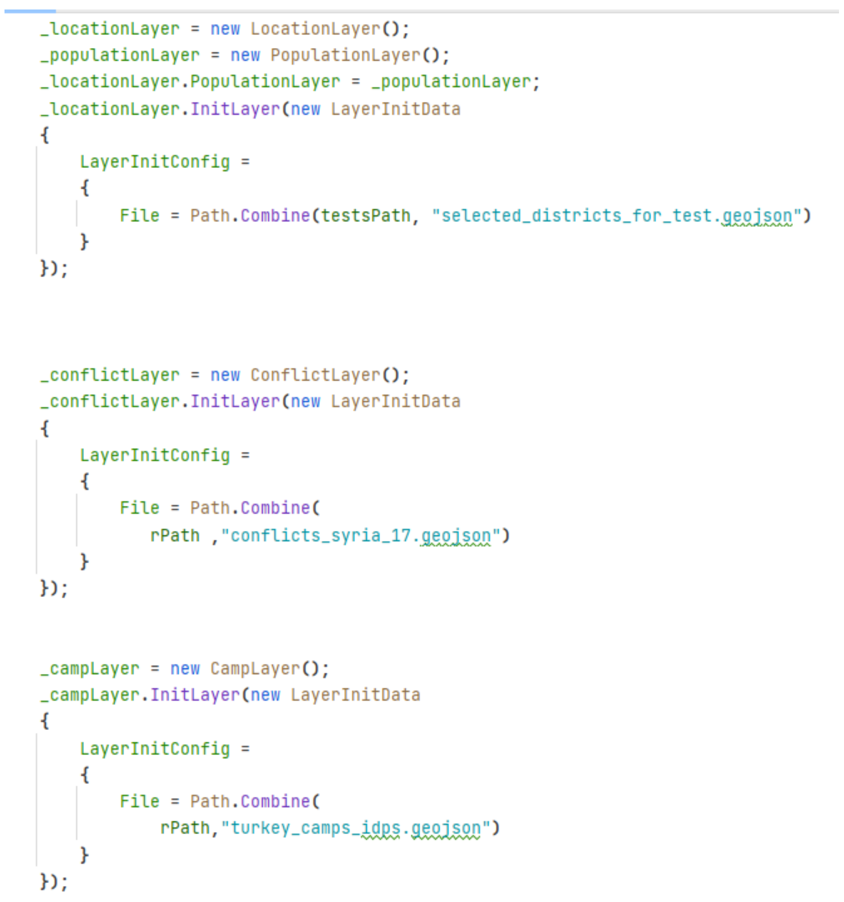
9.1 Verification

Verification, a pivotal step in model development, serves the function of determining the correctness of the model's implementation. It encompasses debugging, meticulous examination of calculations, and testing of model components (Xiaorong Xiang et al., 2005).

Accordingly the verification process for this model consists of repeatedly running the simulation in debug mode to identify and rectify errors, to scrutinize the code for semantic coding issues, and most importantly, of testing the separate functions of the model which come together to produce an emergent behavior.

The tests were conducted using the open-source unit testing tool for the .NET Framework xUnit (xUnit.net, 2023).

Building the Test Environment In order to write the tests described above, a simplified version of the simulation environment had to be built manually, when it is otherwise built by the MARS Framework. Figure 9.1 shows the initialization routine called by the test class.



```
_locationLayer = new LocationLayer();
_populationLayer = new PopulationLayer();
_locationLayer.PopulationLayer = _populationLayer;
_locationLayer.InitLayer(new LayerInitData
{
    LayerInitConfig =
    {
        File = Path.Combine(testsPath, "selected_districts_for_test.geojson")
    }
});



_conflictLayer = new ConflictLayer();
_conflictLayer.InitLayer(new LayerInitData
{
    LayerInitConfig =
    {
        File = Path.Combine(
           rPath ,"conflicts_syria_17.geojson")
    }
});


_campLayer = new CampLayer();
_campLayer.InitLayer(new LayerInitData
{
    LayerInitConfig =
    {
        File = Path.Combine(
            rPath,"turkey_camps_idps.geojson")
    }
});
```

Figure 9.1: Diagram showing the initialization of the simulation environment for testing

The following model functionalities were tested via unit tests:

Environment This test verified the seamless integration of agents into the environment, assessing their ability to traverse it. It also confirmed the model's capacity to retrieve agent information and positions from the environment. For its implementation, two agents were spawned at a selected location and then retrieved by accessing the environment. One of them was then moved to another location in order to check if the position of agents is updated properly in the environment.

Location Initialization Ensuring that all location parameters began with accurate and valid values, this test compared initialization values with manual determinations of the number of camps, conflicts, and neighbors at three selected locations. An additional test confirmed that the resident population of a location, made available by the population data, is correct for a selected location.

Migrant Population Per Location This test validated that the number of migrants at each location aligned with expectations, even with agents entering and departing locations concurrently.

Number of Social Contacts At Location Agents' decision-making relies on factors such as the number of friends and kins at a location. This test scrutinized the system's ability to provide and retrieve this critical information accurately. This was achieved by adding social ties between two agents whose locations are known, and checking if the method called to determine the number of social contacts at a location returns the correct result.

Update of Social Network Social networks evolve through the formation of new friendship ties at camp locations. This test validated the sequence that successfully adds new friend references to an agent's network, ensuring the integrity of this feature.

Challenges of Verification The decision making process of agents depend on pseudo-random numbers, such as the social contacts of agents being picked at random. While this has the benefit of simulating unmeasured variables and random effects, all simulation runs are expected to have different outcomes, making some parts of the simulation difficult to verify.

9.2 Validation

Validation is the process of ensuring that the behavior of the model corresponds to the behavior of the target (Gilbert, Troitzsch, 2005). It determines if the simulation outputs are consistent with real-world outputs (Xiaorong Xiang et al., 2005). In this case the target behavior is the real-world movement of Syrian forced migrants.

Relevant Methods of Validation Here are some methods of validation which are relevant in the context of this work:

1. Empirical Validation: The simulation output is compared against real-world data (Gilbert, Troitzsch, 2005).

2. Satisfaction Tests: Determine whether the desired or intended outcome has been achieved to a sufficient degree (Bungartz et al., 2013).

3. Model Comparison: Compare the model's results to the results of another model which has already been verified (Xiaorong Xiang et al., 2005).

4. Face Validity: Examine a graphical representation of the model's behavior and decide if the model behaves reasonably (Xiaorong Xiang et al., 2005).

Challenges of Validation The complex nature of ABMs, the numerous variables and the possible variation in results from tweaking just one variable make validating ABMs a difficult task. The complexity of the system also makes it difficult to be sure whether the behavior of the simulation is representative of the actual system (Niazi et al., 2017).

Furthermore both the simulation and the target behavior which is to be reflected in the simulation, are likely to depend on random factors. Thus it is difficult to estimate whether a model can be trusted based on the difference between model outputs and data. It could also be the case that the model logic is correct but the real-world data is incorrect (Gilbert, Troitzsch, 2005).

9.2.1 Validation of Syrian Refugees in Turkey Simulation

Validation data In order to validate the model, simulation predictions are compared against the outputs of the reference model, which has already been validated. The input data is the same as the reference model's, except that the simulation started with a refugee population of only 2 344 876 instead of the real number 3 607 694. The reason being the bad performance at such a high number of threads. The reference model was tested using the actual number, but it was run on virtual machines with 16 virtual CPU clusters (Richey, 2020), while the MARS model was run on a single dual core processor.

Validation Idea Since no real-world data for validation could be found, a combination of the model comparison and face validity validation methods are employed to ensure that the behavior of Syrian refugees in Turkey are sufficiently representative of the actual system. The only outputs from Richey's model available are maps of Turkey which visualize the distribution of refugees over the districts after the simulation run. In order to compare the results, a map showcasing the ten most common routes undertaken by individual agents is created using the results of the replicated MARS model after a simulation run of the same length (40 days), with the parameter values suggested by Richey. Finally, a comparative analysis of the two maps is conducted. The aim is to assess the extent to which the replicated model's behavior aligns with the reference model's.

The following table shows the suggested parameters:

Parameter Name	Value
kin weight	0.2
friend weight	0.2
camp move probability	0.3
conflict move probability	0.7
other move probability	0.85
population weight	0.75
camp weight	0.75
conflict weight	0.75
location weight	0.75
number new agents to spawn	50

Table 9.1: Table of parameter values for the Turkey simulation run.

Outputs and Discussion Figure 9.2 shows the reference model's output map of Turkey, with regions color-coded to indicate the concentration of Syrian refugees (Richey 2020). Over time, the reference model shows a significant shift in the distribution of refugees, as they transition from a more dispersed pattern to congregating in the south-

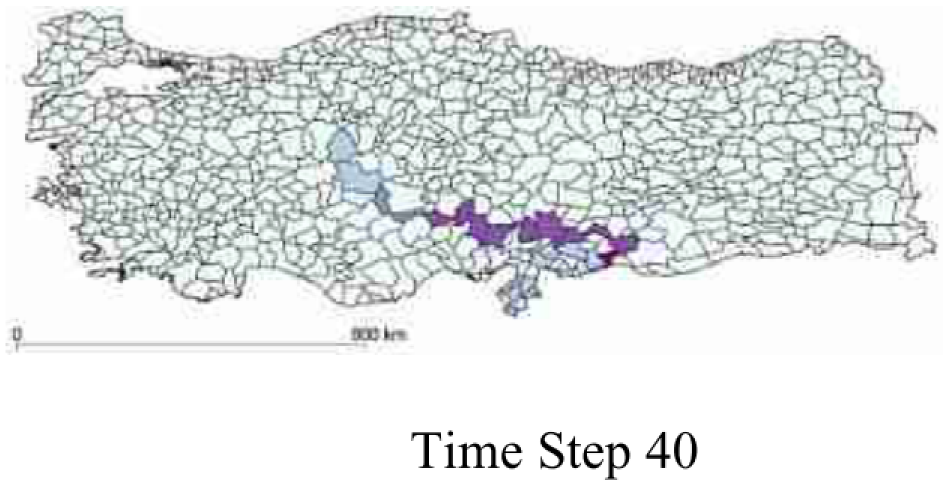


Figure 9.2: Output of the reference model after 40 days

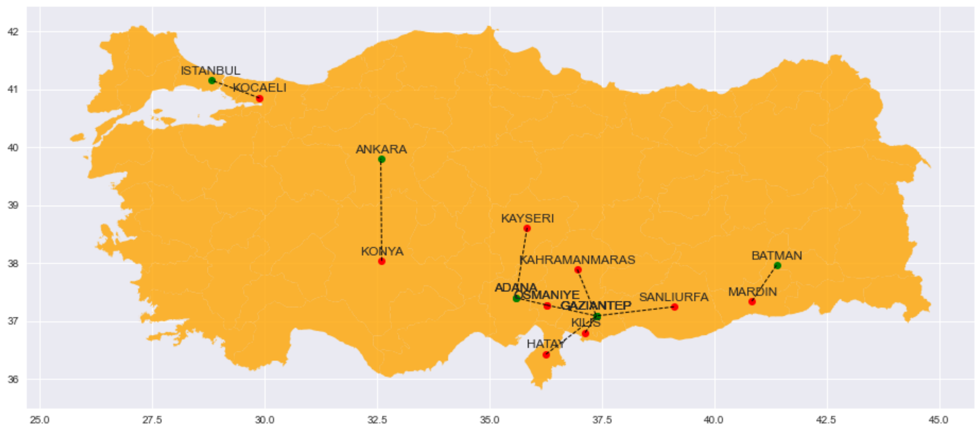


Figure 9.3: Output of the replicated model after 40 days

ern and mid-western regions of the country, with Istanbul in the north-west emerging as a notable hub.

Figure 9.3 shows the output of the replicated model offering a view of refugee agents' movements throughout the simulation period. Each route on the maps represents the journey from an origin location (represented by a red dot) to a destiny location (represented by a green dot).

Analyzing the MARS model's output, it becomes evident that agents have gravitated towards select regions in the southern part of the country, leading to localized pooling. Additionally, a substantial number of agents have migrated to Ankara in the mid-west and Istanbul in the north-west, in line with the reference model's observations. However, the model also reveals a behavior absent in the reference model, with a significant influx of agents moving further eastward.

9.2.2 Validation of IDPs' Simulation

Validation Data In order to validate the model, simulation predictions are compared against real-world estimated data on IDPs' population per Syrian governorate. The validation data used is from January 2017. The simulation is configured using the administrative level 3 of Syria. The output is aggregated by governorate (administrative level 1). The initial number of IDPs per district is reduced for the calibration experiments in order to perform many model runs, which would otherwise take a very long time. Thus the model started with a total of 170560 agents for each experiment instead of the data driven approximated number of 2 048 183.

Validation Idea In order to validate the model with the movement of IDPs as the target, a variation of the empirical validation method is employed. Since the available data on IDPs' population per governorate is based on estimated numbers, the validity of the model would not be accurately reflected by calculating model error based on absolute values. Instead, the proportional distribution of IDPs across Syrian governorates is used for validation. The percentage of total IDPs in a governorate is taken from the actual data and compared against the percentage predictions made by the simulation.

Error calculation The MAPE (Mean Absolute Percentage Error) is a measure of accuracy of a forecast system (StatisticsHowTo, 2022). The absolute percentage error for each governorate is calculated based on the proportional distribution. Having calculated the error for each governorate, the MAPE is calculated by adding up all the error values and dividing the result by the number of governorates. Equation 9.1 shows the MAPE calculation, where N is the total number of governorates.

$$\frac{1}{N}\sum_{i=1}^{N}\left|\frac{\text{actual\_percentage}_i - \text{predicted\_percentage}_i}{\text{actual\_percentage}_i}\right| \times 100 \tag{9.1}$$

Calibration

Using the described validation method, the model parameters are re-calibrated to fit the IDP simulation use case. For each calibration experiment, the model is run three times and the outputs are averaged.

Firstly, the error for the model run using the parameter values suggested by Richey (Richey, 2020) is calculated, representing a baseline to compare against. The baseline value for each parameter is the same as in table 9.1. The exception is the number of new agents to spawn, which is set to 9500. This number is taken from real world average estimations (Mooney, 2023). Moreover, the new agents are dispersed throughout the country and not spawned at a handful of locations. Table 9.2 shows the calculated MAPE across all districts for each calibration experiment. The calculated error for the baseline run is 144.5%.

Simulation Run with calibrated parameters The simulation ran again using the local minima of the parameters determined through calibration. This time with the total initial IDP population of 2 048 183. After 30 days there was a total of 2 347 891 agents in the simulation. The average calculated MAPE for this model run amounts to 66.7%.

Calibration Experiments Discussion The baseline accuracy of over 144% can only be improved slightly by changing the value of parameters, indicating a low sensitivity factor. Notably, assigning a low value to the camp weight improved accuracy the most, diminishing the MAPE by about 14% (Experiment 1.1). A remarkable accuracy improvement is achieved by running the simulation with a substantially higher number of agents, which confirms the observation made by Richey (Richey, 2020) that the accuracy of the model is highly dependent on the number of agents.

Experiment Number	Calibrated Parameter	Parameter Value	MAPE
1.1	camp weight	0.05	130.7%
1.2	camp weight	0.5	144.03%
1.3	camp weight	0.75	142.59%
2.1	conflict weight	0.25	132.57%
2.2	conflict weight	0.75	150.08%
2.3	conflict weight	0.95	142.18%
3.1	social weights	0.0	144.75%
3.2	social weights	0.5	144.9%
3.3	social weights	0.75	144.94%
4.1	population weight	0.5	139.44%
4.2	population weight	0.25	142.17%
4.3	population weight	0.0	196.45%
5.1	location weight	0.5	143.40%
5.2	location weight	0.25	141.21%
5.3	location weight	0.05	141.36%
6.1	camp move prob	0.1	136.53%
6.2	camp move prob	0.5	144.88%
6.3	camp move prob	0.75	142.59%
7.1	conflict move prob	0.5	140.98%
7.2	conflict move prob	0.9	142.36%
8.1	other move prob	0.65	143.90%
8.2	other move prob	0.45	140.42%
8.3	other move prob	0.25	139.40%

Table 9.2: Table of calibration experiments.

10 Scenarios

In the previous chapter the model verification and validation process was described in detail. Notably the model's IDP behavior simulation was validated empirically. In this chapter, satisfaction tests are conducted in the form of prediction scenarios. Based on these scenarios, the question of whether this model holds the potential to be useful for humanitarian organizations in their resource distribution planning can be explored.

10.1 Scenario Description

The following scenarios aim to provide practical insights into the distribution of IDPs at varying administrative levels, thereby aiding in more informed resource allocation decisions. Each scenario involves running the simulation with the calibrated parameter values. Utilizing 2 053 122 agents, and spanning a simulation period of 30 days in the month of March 2017. The initial IDP population per region is taken from real-world data of IDP arrival estimations from the end of 2016 plus the arrivals in January and February. The number is then reduced by 300 000 because of performance issues.

10.1.1 Scenario 1: Administrative Level 1

The first scenario is centered on predicting the distribution of IDPs across the Syrian governorates, which represent administrative units at level 1. The aim is to forecast how IDP populations are likely to be distributed across these larger geographical areas. This prediction has substantial implications for humanitarian organizations at the macro level, providing insights into resource allocation and aid planning on a statewide scale. This scenario is additionally run in the time period of January 2017.

10.1.2 Scenario 2: Administrative Level 2

In the second scenario, the focus is narrowed to the Syrian districts, which are administrative units at level 2. This scenario seeks to predict the distribution of IDPs at a more localized level. Such predictions are invaluable for humanitarian organizations with district-level operations, allowing for more precise resource distribution and response planning.

10.1.3 Scenario 3: Administrative Level 3

The third scenario dives even deeper, focusing on the Syrian sub-districts, which are administrative units at level 3. Predicting the distribution of IDPs at this granular level provides a highly detailed understanding of population movements. This level of prediction serves humanitarian organizations involved in localized and community-specific relief efforts, enhancing their capacity for targeted resource allocation.

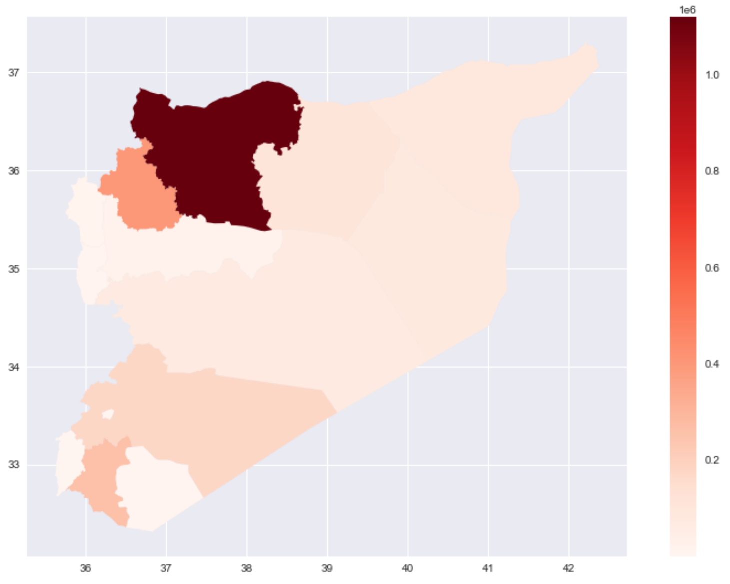
10.1.4 Scenario 4: Routes Prediction

The fourth scenario involves the prediction of the most common IDP routes within the simulation. This scenario offers an opportunity to explore if routes taken by IDPs in Syria can be reproduced in the model.

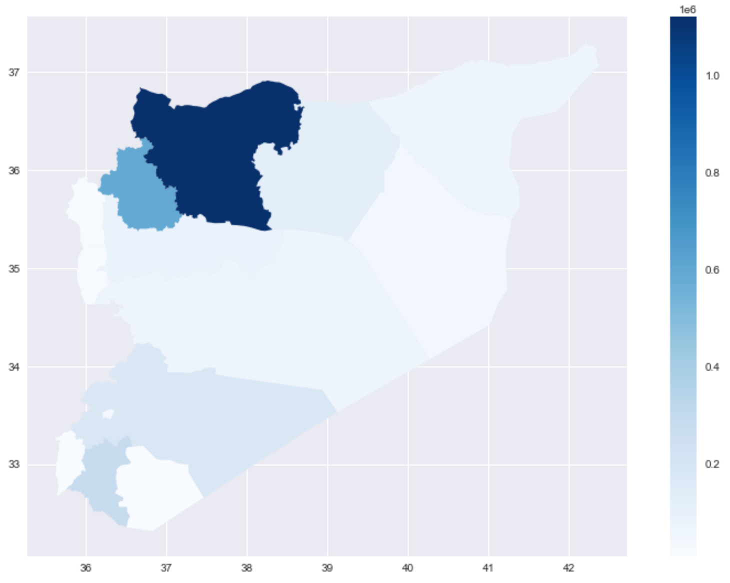
10.2 Results

10.2.1 Results of Scenario 1

The figures 10.1 to 10.4 show the predictions made by the model in the context of the first scenario, compared against real-world data for the same time period and administrative level.

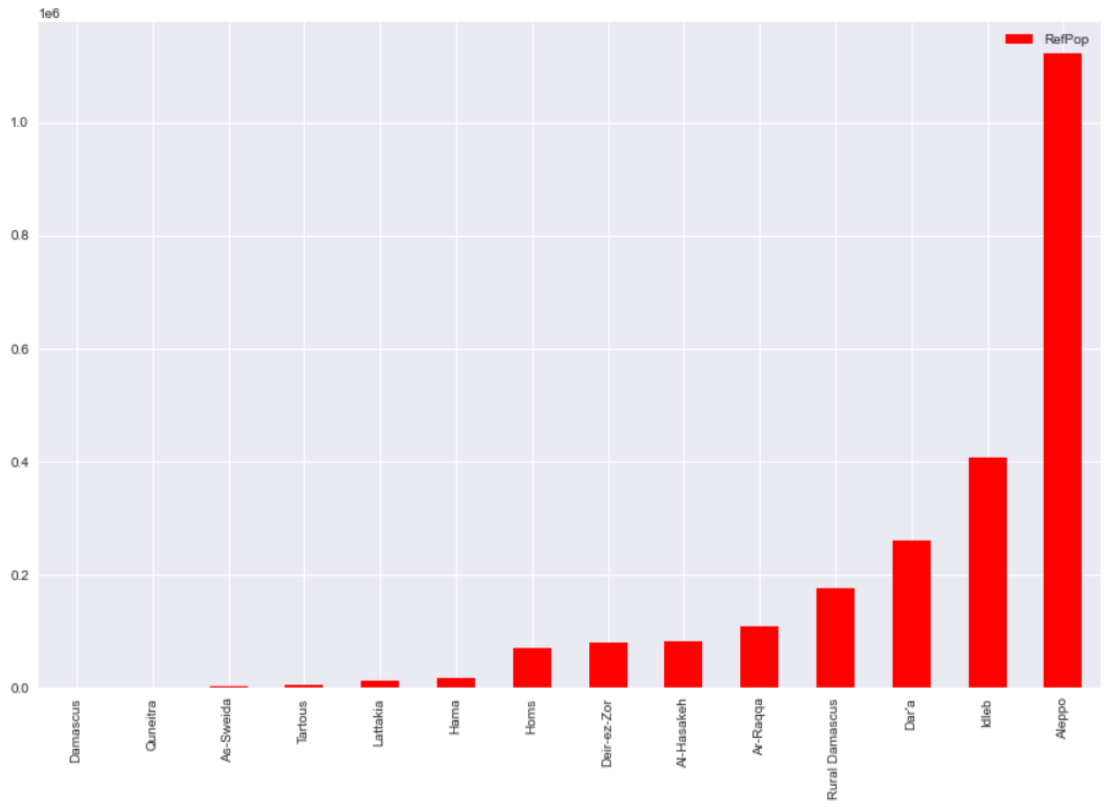


(a) Predicted

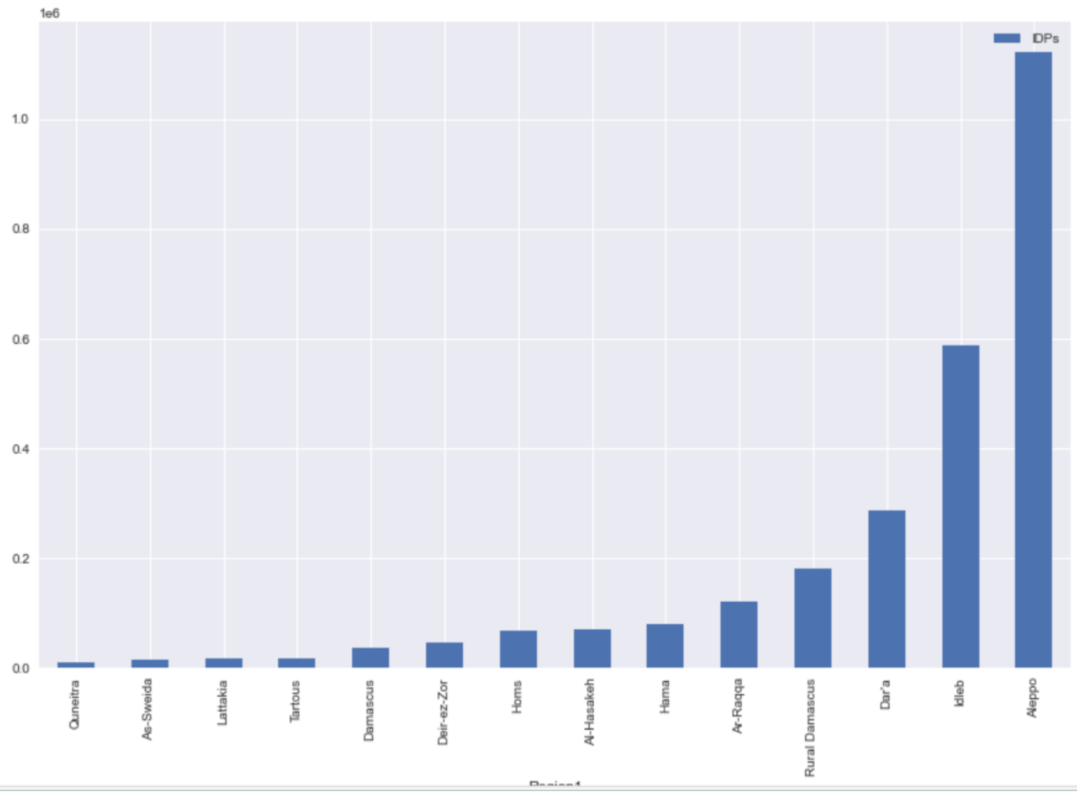


(b) Actual

Figure 10.1: Map of predicted IDP population density compared against population density from real-world estimations in March 2017

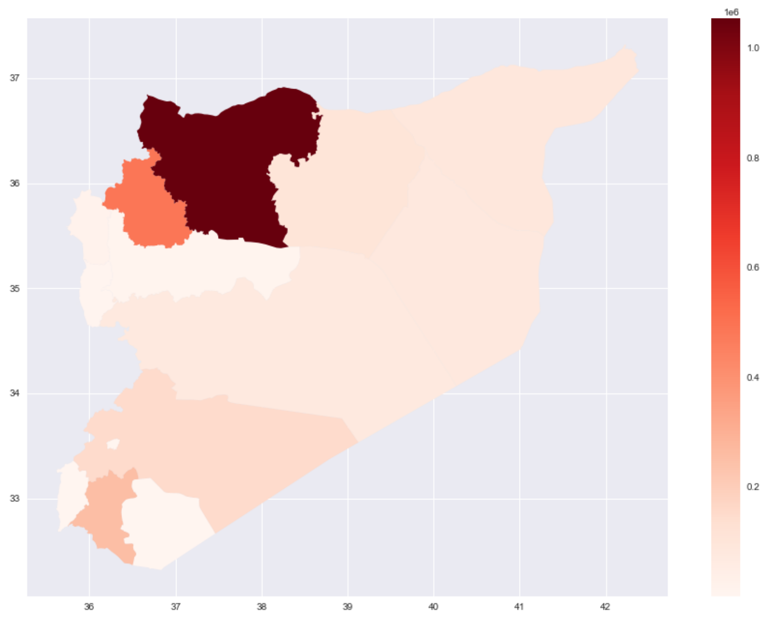


(a) Predicted

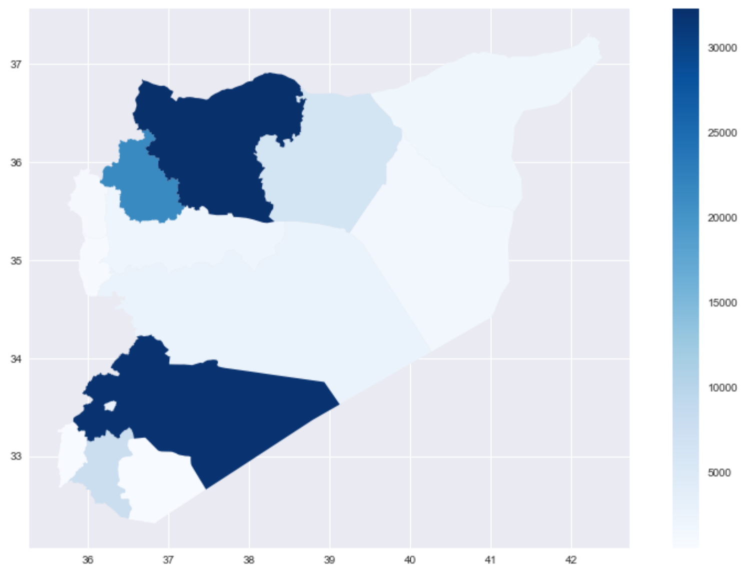


(b) Actual

Figure 10.2: Bar chart of predicted distribution of IDPs across governorates compared against real-world estimations in March 2017

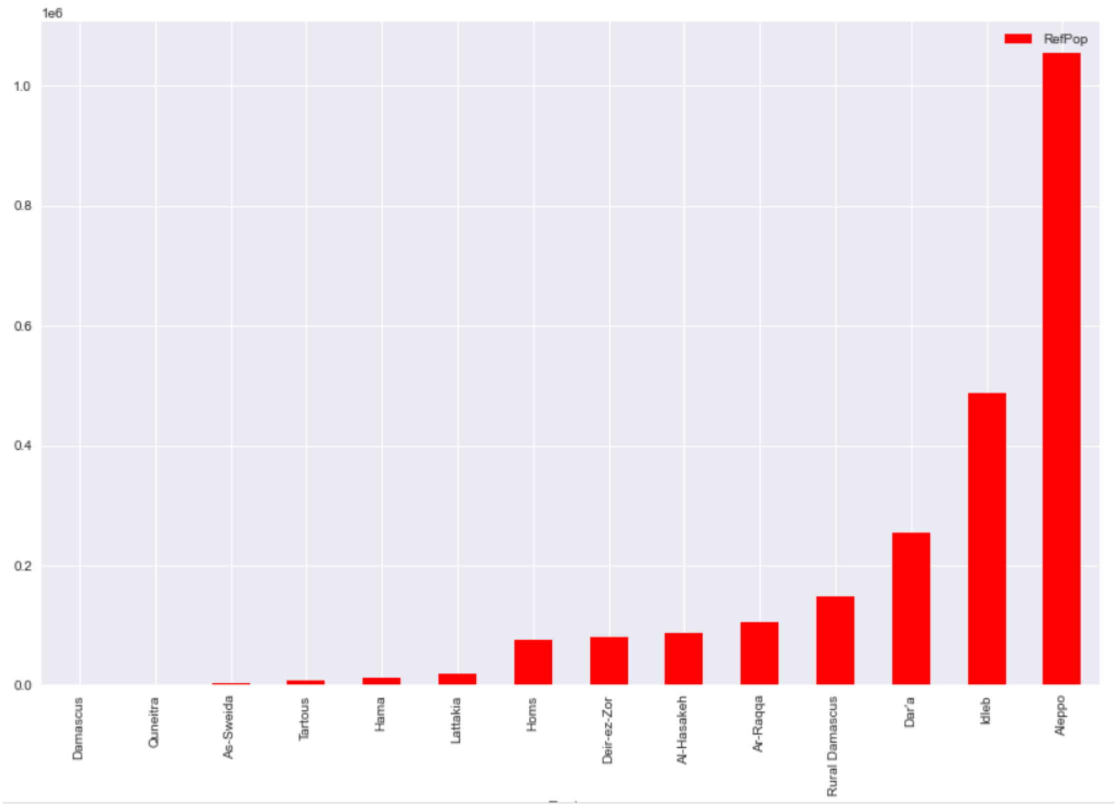


(a) Predicted

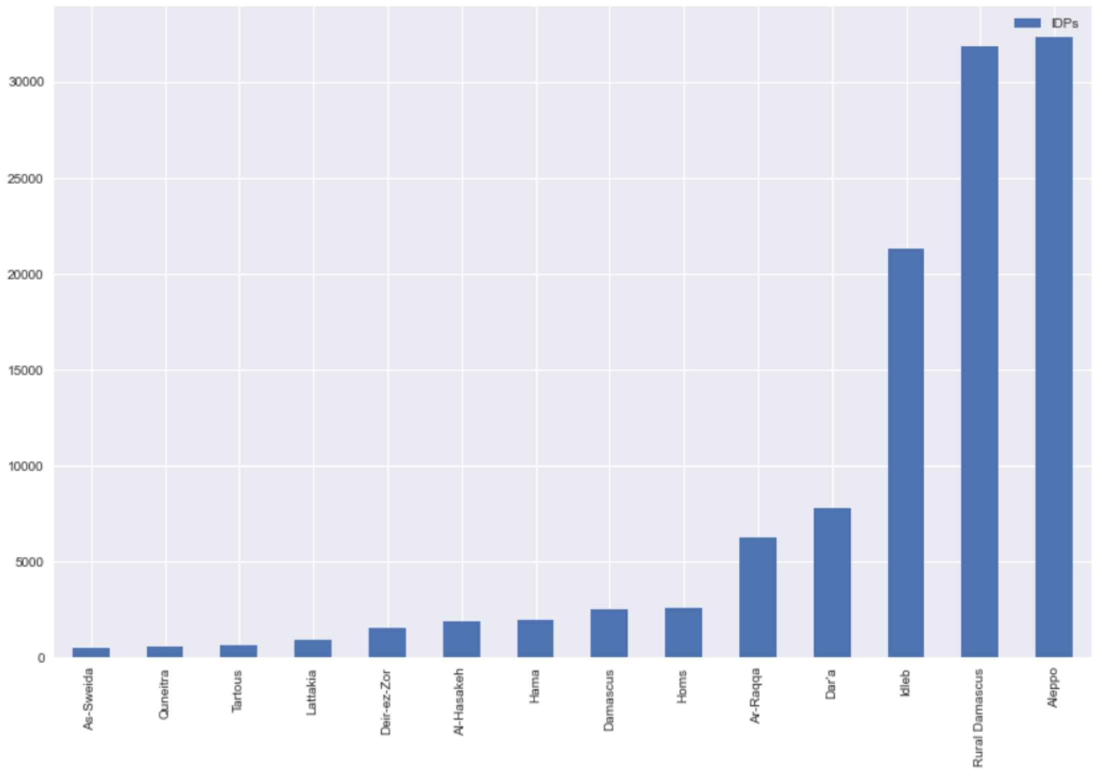


(b) Actual

Figure 10.3: Map of predicted IDP population density compared against population density from real-world estimations in January 2017



(a) Predicted



(b) Actual

Figure 10.4: Bar chart of predicted distribution of IDPs across governorates compared against real-world estimations in January 2017

Both the model output and the real-world data, depicted through maps and bar charts, share common features, such as showcasing the highest IDP population densities in the north-west and south-west regions of Syria. However, a few notable disparities emerge. The most significant of these disparities is evident in Rural Damascus, where the model underestimates the IDP population in January. Furthermore, when analyzing the bar charts, it becomes evident that the model's predictions consistently underestimate the IDP population in Damascus.

In contrast, the model performs well in identifying the top 5 governorates with the highest IDP populations—Aleppo, Rural Damascus, Idleb, Dar'a, and Ar-Raqqa—correctly predicting their prominence. Additionally, it accurately anticipates three out of the five governorates—As-Sweida, Quneitra, and Tartous—with the smallest IDP populations in January. Furthermore, the model predicts four out of the five with the smallest IDP population with Damascus being included.

10.2.2 Results of Scenario 2

Also visualized through color-coded maps, the model's output map, displayed in red, is juxtaposed with the real-world data map, represented in blue. Additionally, a third map, colored in green (Figure 10.5), illustrates the initial distribution of the IDP population over the districts, offering a reference point.

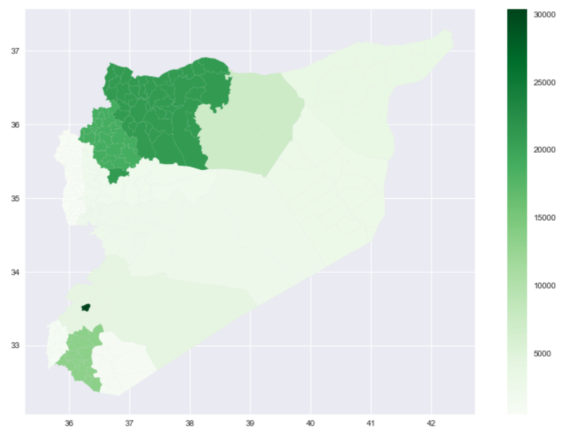
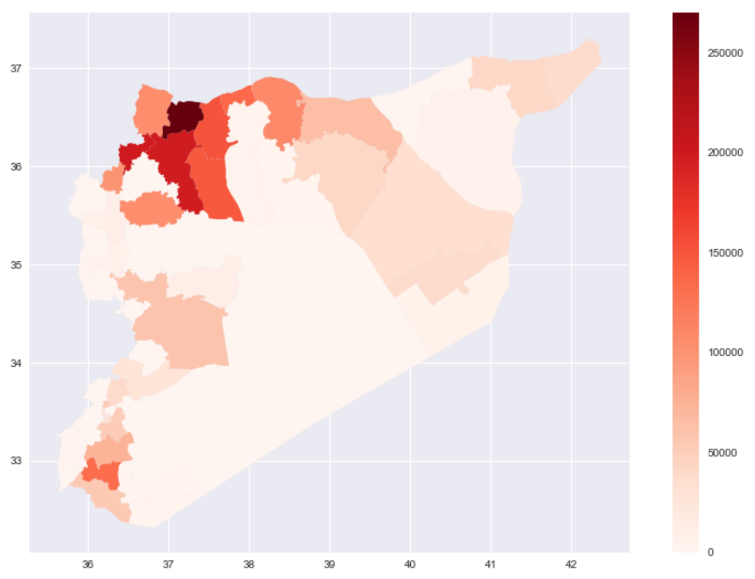
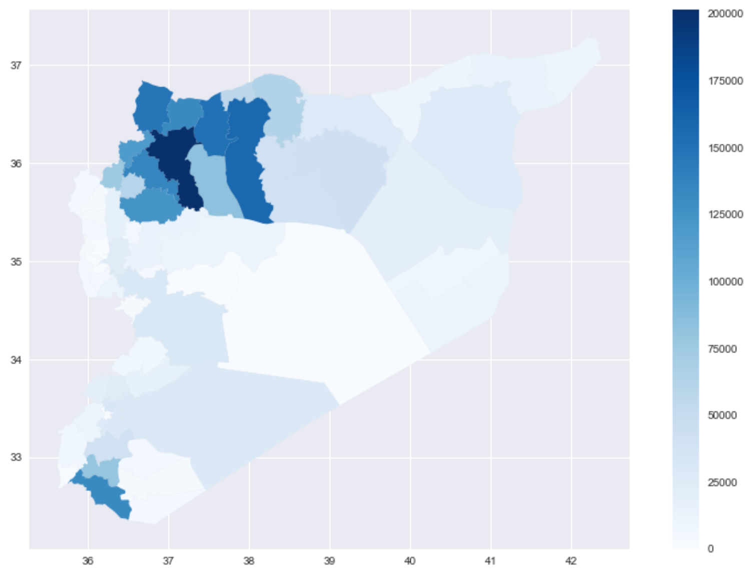


Figure 10.5: Initial distribution of IDP population for March predictions



(a) Predicted

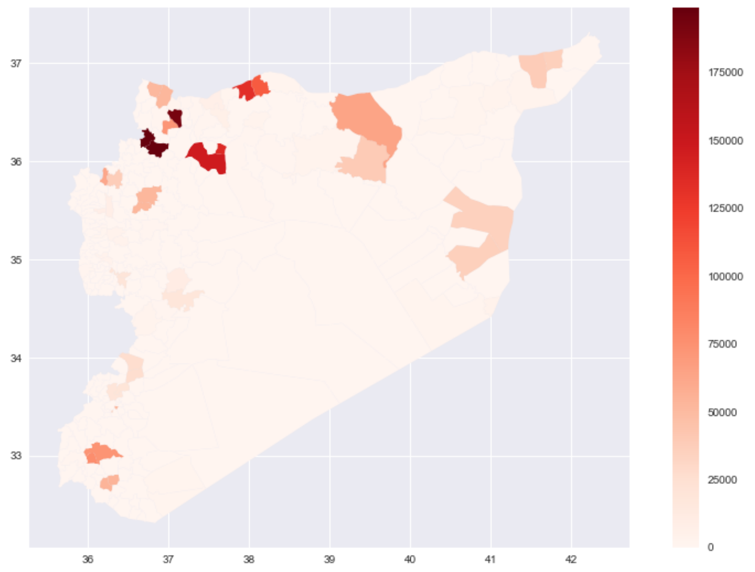


(b) Actual

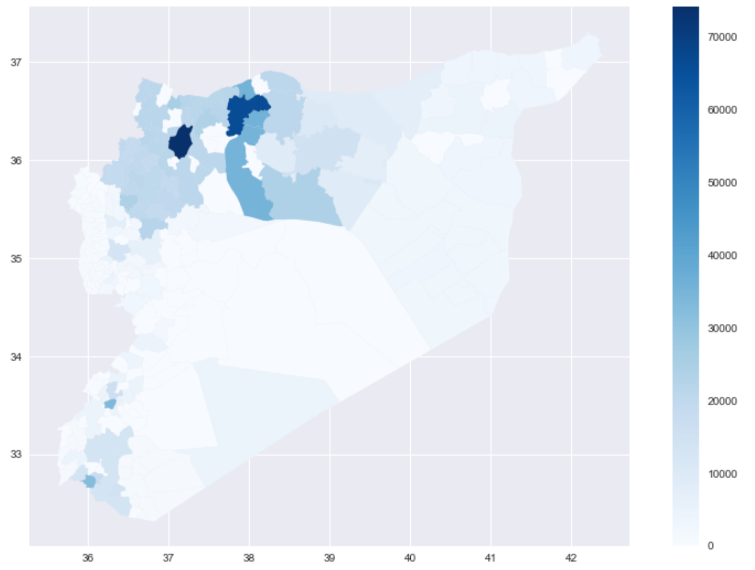
Figure 10.6: Map of predicted IDP population density per administrative level 2 compared against real-world estimations in March 2017

Both the model output map and the real-world data map exhibit notable IDP concentrations within specific districts. Regions in the north-west and south-east consistently stand out, demonstrating a degree of alignment between the model output and real-world data. However, the broader comparison reveals divergences. The real-world map, in particular, showcases concentrations in distinct districts that differ from those highlighted on the model output map. However, these discrepancies are tempered by the observation that the IDP concentrations in the output map tend to exhibit close proximity to those seen on the real-world data map.

10.2.3 Results of Scenario 3



(a) Predicted



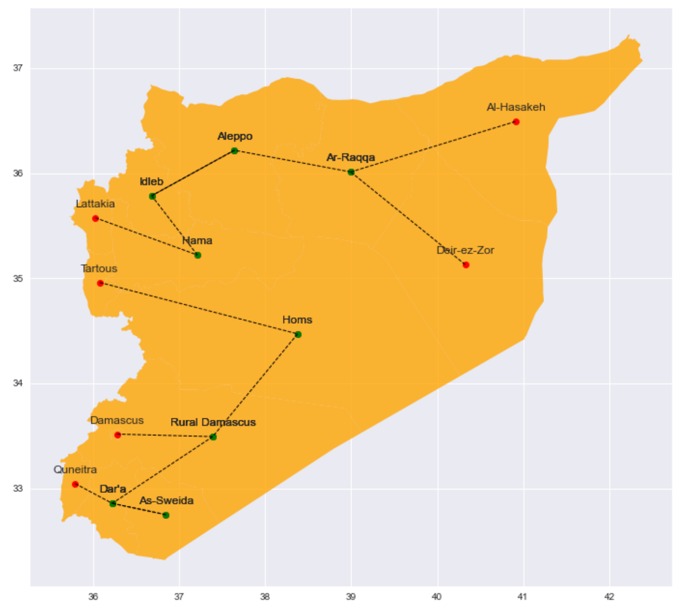
(b) Actual

Figure 10.7: Map of predicted IDP population density per administrative level 3 compared against real-world estimations in March 2017

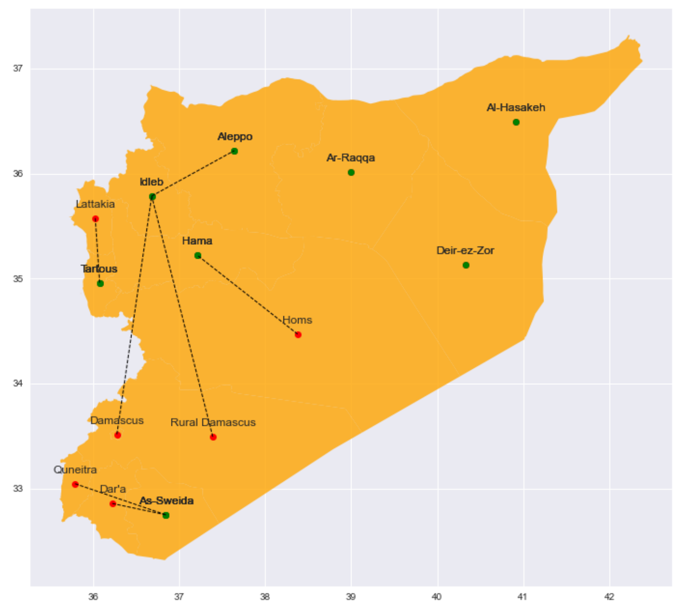
An analysis of the maps at the administrative level 3 reveals even denser concentrations of IDPs within certain sub-districts. Both the model output map and the real-world data map shine a spotlight on regions where IDPs have notably congregated. In the model output map, this is often manifested as the emergence of specific districts marked by a pronounced pooling of IDPs. Conversely, the real-world data map exhibits a more widespread pattern of IDP presence across sub-districts.

10.2.4 Results of Scenario 4

Figure 10.8 shows the most common routes of the agents in the model and the most common routes of IDPs according to real-world estimations. Each common route, representing the most frequently chosen destination from each of the 14 governorates, is illustrated through two dots connected by a line. The red dot signifies the origin, while the green dot denotes the destination, creating a visual representation of these migration paths. Green dots not connected by a line represent return movements.



(a) Predicted



(b) Actual

Figure 10.8: Map of predicted IDP population density per administrative level 2 compared against real-world estimations in March 2017

Upon analyzing these maps, you can see that the only route depicted in both the model output and real-world data maps is the one between Idleb and Aleppo. While the real-world data map presents a trend of movement towards the north-western regions of Syria, the model output not only reflects the north-western migration trend but also demonstrates an additional pattern of movement toward the center of the country.

11 Discussion

This chapter is dedicated to a discussion of the results obtained from the four simulation scenarios conducted previously. We will delve into the potential utility of the model for humanitarian actors operating in Syria and analyze the sources of error in the model, shedding light on areas where improvements may be necessary to enhance its accuracy.

11.1 Analysis of Model Utility for Humanitarian Actors

Based on the findings from scenario 1, the model holds potential value for humanitarian organizations operating at the statewide level. The model's ability to predict the top 5 governorates with the highest IDP populations can inform resource allocation strategies, helping organizations prioritize aid where it is most needed.

However, as scenarios 2 and 3 reveal, the model's usefulness for humanitarian actors with district- and subdistrict-level operations is limited by prediction inaccuracies at these finer-grained levels. This limitation can impact the precision of resource distribution and response planning. Nonetheless, it's worth noting that the proximity of IDP concentrations on the model output map to the real-world data map partially mitigates this restriction, providing some utility at district and subdistrict levels.

The model's inability to accurately predict migration routes, as demonstrated in Scenario 4, significantly restricts its utility in this regard. The model introduces patterns of movement that do not align with real-world migration trends, limiting its practicality for predicting and planning for IDP movements.

In conclusion, the model shows promise in providing valuable insights for humanitarian actors, particularly at the statewide level. However, its limitations at finer-grained administrative levels and in predicting migration routes underscore the need for further research and improvements to increase its practicality for real-world decision-making and planning in the context of internal displacement.

11.2 Sources of Error in the Model

The following sources of error warrant consideration and potential improvements:

1. **Agent Scaling:** The model error was significantly reduced by running the simulation with more agents. This observation aligns with the idea that migration is an emergent behavior, and a larger scale is essential to accurately replicate real-world behavior (Hinsch, Bijak, 2019). Increasing the number of agents is an avenue to explore for refining the model.

2. **Infrastructure Data:** Harrison's research highlighted the significance of infrastructure in the decision-making process of IDPs in Syria (Harrison, 2016). This implies that the absence of data related to the state of infrastructure across Syrian regions hampers the model's accuracy. Incorporating such data, if available, could potentially improve the model's performance.

3. **Rudimentary Social Network Functionality:** The social network functionality, replicated from the reference model, remains rudimentary and does not significantly contribute to the accuracy of IDP simulations. Enhancing this aspect of the model may lead to more realistic results.

4. **Initial IDP Population Distribution:** The model's initial distribution of IDPs is a substantial factor in the decision-making mechanism. However, relying on evenly distributing the population of a governorate across its administrative units did not reflect real-world patterns. Acquiring more precise initial IDP population data could address this discrepancy.

5. **Outdated Resident Population Data:** The use of resident population data from 2004 to spawn new agents may introduce inaccuracies due to its age. Updating this data to reflect current numbers could lead to more accurate simulations.

12 Conclusion and Next Steps

The thesis followed two primary research objectives. The first was to redesign an existing agent-based model originally developed by Richey (2020) to predict the distribution of Syrian refugees across Turkey. This involved adapting Richey's model to operate within the MARS framework, a decision driven by the framework's scalability and intuitive design. The second research objective was to extend this model to simulate IDPs in Syria, evaluating whether the modified model, inclusive of IDPs, could offer valuable insights for humanitarian organizations in the context of resource distribution planning.

The efforts to achieve these objectives have yielded significant achievements and insights into the field of agent-based modeling in humanitarian settings.

The utilization of the MARS framework for the redesign and extension of the model provided the necessary tools to adapt and enhance the model effectively. By incorporating the complexities of internal displacement in Syria, a realm largely uncharted in the domain of empirically validated ABMs of forced migration was entered.

The analysis of four scenarios highlights the model's potential utility. In scenario 1, it provides valuable insights at the statewide level, having the potential to aid in resource allocation. The model accurately predicts top governorates with high IDP populations. However, inaccuracies in predicting IDP distributions at finer administrative levels and migration routes pose limitations.

Further limitations include the lacking resources to scale the simulation further. Greater scalability would provide insights into the model's evolving accuracy. Data accessibility, particularly up-to-date population data, impacts accuracy calculations and fidelity to real-world conditions.

In order to further advance the model's utility for humanitarian organizations, future research should prioritize scaling up the number of agents in the simulation and the tackling of the error sources in the model described in the previous chapter. Another intriguing avenue for future work is to apply the model to other conflict zones, testing its adaptability and effectiveness in diverse humanitarian contexts.

In conclusion, the model represents a significant step in humanitarian simulation engineering. It demonstrates the potential for providing valuable insights at a statewide level and offers a solid foundation for future work. While challenges remain, these limitations should not deter further exploration but rather inspire continued efforts to refine and expand the model's utility.

Bibliography

ACLED . Data Export Tool. 2023. Available online at https://acleddata.com/data-export-tool/, checked on 10/12/2023.

Bungartz Hans-Joachim, Zimmer Stefan, Buchholz Martin, Pflüger Dirk. Modellbildung und Simulation - Eine anwendungsorientierte Einführung. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. Second Edition.

Clemen Thomas, Ahmady-Moghaddam Nima, Lenfers Ulfia A., Ocker Florian, Osterholz Daniel, Ströbele Jonathan, Glake Daniel. Multi-Agent Systems and Digital Twins for Smarter Cities // Proceedings of the 2021 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. 2021a. 45–55. (ACM Digital Library).

Clemen Thomas, Lenfers Ulfia A., Dybulla Janus, Ferreira Sam M., Kiker Greg A., Martens Carola, Scheiter Simon. A cross-scale modeling framework for decision support on elephant management in Kruger National Park, South Africa // Ecological Informatics. 2021b. 62. 101266.

Doocy Shannon, Lyles Emily, Delbiso Tefera D., Robinson Courtland W. Internal displacement and the Syrian crisis: an analysis of trends from 2011-2014 // Conflict and health. 2015. 9. 33.

Epstein Joshua M. Agent Zero: Toward Neurocognitive Foundations for Generative Social Science. 25. Princeton: Princeton University Press, 2014. 1. (Princeton Studies in Complexity).

Gilbert Nigel, Troitzsch Klaus G. Simulation for the social scientist. Berkshire: Open University Press, 2005. Second Edition.

Guizzardi Giancarlo. Ontological Foundations for Structural Concept Models. Enschede, 2005.

Guizzardi Giancarlo, Wagner Gerd. Tutorial: Conceptual simulation modeling with Onto-UML // 2012 Winter Simulation Conference. 2012. 1–15.

Gulden Timothy, Harrison F. Joseph, Crooks T. Andrew. Modeling Cities and Displacement through an Agent-based Spatial Interaction Model // The Computational Social Science Society of America Conference. 2011.

Güngör Özlem, Günnec Dilek, Salman F. Sibel. Prediction of Migration Paths Using Agent-Based Simulation Modeling: The Case of Syria // Proceedings of the International Conference on Industrial Engineering and Operations Management, Istanbul, Turkey, March 7-10, 2022. Southfield, Michigan, USA: IEOM Society International, 2022.

HDX . The Humanitarian Data Exchange. 2023. Available online at https://data.humdata.org/, checked on 10/12/2023.

Harrison Ethan. Modeling Syrian Internally Displaced Person Movements: A Case Study of Conflict, Travel, Accessibility, and Resource Availability. 2016. (Student Writing).

Hébert Guillaume Arnoux, Perez Liliana, Harati Saeed. An Agent-Based Model to Identify Migration Pathways of Refugees: The Case of Syria // Agent-Based Models and Complexity Science in the Age of Geospatial Big Data. 2018. 45–58.

Hinsch Martin, Bijak Jakub. Rumours lead to self-organized migration routes // The 2019 Conference on Artificial Life: How Can Artificial Life Help Solve Societal Challenges? 2019.

Hüning Christian, Adebahr Mitja, Clemen Thomas, Dalski Jan, Clemen Ulfia A., Grundmann Lukas, Dybulla Janus, Kiker Greg A. Modeling & Simulation as a Service with the Massive Multi-Agent System MARS // Agent-Directed Simulation Symposium (ADS 2016). 2016. (Simulation series).

Klabunde Anna, Willekens Frans. Decision-Making in Agent-Based Models of Migration: State of the Art and Challenges // European journal of population = Revue europeenne de demographie. 2016. 32, 1. 73–97.

MARS-Group . Hello from MARS. 2023. Available online at https://www.mars-group.org/, checked on 10/06/2023.

Mooney Erin. The inside story: internal displacement in Syria. 2023. Available online at https://www.fmreview.org/syria/mooney, checked on 10/13/2023.

Niazi Muaz A., Hussain Amir, Kolberg Mario. Verification & Validation of Agent Based Simulations using the VOMAS (Virtual Overlay Multi-agent System) approach. 2017.

Padgham . Developing Intelligent Agent Systems. Chicester: John Wiley and Sons, 2004.

Project Jupyter . About Us. 2023. Available online at https://jupyter.org/about, checked on 09/20/2023.

Rauch Gedeon. Was ist Jupyter Notebook? // Dev-Insider. 2022.

Richey Melonie K. Scalable Agent-Based Modeling of Forced Migration. Fairfax, 2020.

Modeling population displacement in the Syrian city of Aleppo. // . 2014. 252–263.

Specification O. M.G. Adopted. United Modeling Language 2.0 Proposal. 2003. Available online at https://sparxsystems.com/bin/UML2SuperStructure.pdf, checked on 09/30/2023.

StatisticsHowTo . Mean Absolute Percentage Error (MAPE). 2022. Available online at https://www.statisticshowto.com/mean-absolute-percentage-error-mape/, checked on 10/13/2023.

Suleimenova Diana, Bell David, Groen Derek. A generalized simulation development approach for predicting refugee destinations // Scientific reports. 2017. 7, 1. 13377.

Thibos Cameron. Half a Country Displaced: the Syrian Refugee and IDP Crisis // IEMed (ed.), IEMed Mediterranean Yearbook 2014, Barcelona : IEMed, 2014. [Migration Policy Centre]. 2014. 54–60.

UNHCR . Syria Refugee Crisis Explained. 2023. Available online at https://www.unrefugees.org/news/syria-refugee-crisis-explained/, checked on 09/19/2023.

Xiaorong Xiang , *Ryan Kennedy* , *Gregory Madey* , *Steve Cabaniss* . Verification and Validation of Agent-based Scientific Simulation Models // Agent-directed simulation conference. 47. San Diego, 2005.

xUnit.net . About xUnit.net. 2023. Available online at https://xunit.net/, checked on 10/13/2023.

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original