

# **Agent-Based Modeling and Simulation of Earthmoving Operations**

**Ahmad Jabri**

A Thesis

In the Department of

Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

The Master of Applied Science (Building Engineering) at

Concordia University

Montreal, Quebec, Canada

July, 2014

© Ahmad Jabri, 2014

**CONCORDIA UNIVERSITY**

**School of Graduate Studies**

This is to certify that the thesis prepared

By: **Ahmad Jabri**

Entitled: **Agent-based Modeling and Simulation of Earthmoving Operations**

and submitted in partial fulfillment of the requirements for the degree of

**Master of Applied Science (Building Engineering)**

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

\_\_\_\_\_  
Chair  
Dr. O. Moselhi, Chair

\_\_\_\_\_  
External Examiner  
Dr. S. Tahar

\_\_\_\_\_  
BCEE Examiner  
Dr. S. Rahaman

\_\_\_\_\_  
BCEE Examiner  
Dr. O. Moselhi

\_\_\_\_\_  
Supervisor  
Dr. T. Zayed

Approved by \_\_\_\_\_  
Chair of Department or Graduate Program Director

\_\_\_\_\_  
Dean of Faculty

Date \_\_\_\_\_

## **ABSTRACT**

### **Agent-Based Modeling and Simulation of Earthmoving Operations**

**Ahmad Jabri**

Simulation has been used in construction modeling for decades, especially in large scale operations, such as earth moving, where heavy and costly equipment is used. Simulation can be used as a planning tool to analyze the time and cost of earthmoving operations. Current methods used in simulating earthmoving operations are based on Discrete-Event Simulation (DES), with recent efforts to introduce System Dynamics (SD) in a hybrid DES-SD approach. However, due to the predetermined nature of Discrete-Event Simulation (DES) models, some inflexibility is experienced when modeling earthmoving operations, which translates into a higher degree of difficulty in regards to model creation and a reduced accuracy of outputs. Although the introduction of System Dynamics (SD) contributed significantly to accounting for qualitative factors and strategic aspects of earthmoving operations, there still exists a need for enhancing the accuracy of capturing the logistics of these operations in a smart and flexible manner.

With the advancement of computational capabilities, Agent-Based Modeling and Simulation (ABMS) is rapidly replacing the conventional simulation techniques. This thesis introduces Agent-Based Modeling and Simulation (ABMS) as an effective tool for modeling earthmoving operations. First of all, it provides a generic methodology introduced for creating Agent-Based models for construction operations, based on a set of rules and criteria. Then, an Agent-Based (AB) model for earthmoving operations consisting of bulldozers, loaders, haulers and spotters is developed. The model in

question governs the process logistics, information sharing, equipment properties as well as activity durations. Finally, a Java-Based software application (*ABSEMO*) is developed as an implementation of the proposed Agent-Based (AB) simulation model. Overall, the desired outcome is to create a smart system that has a flexible logic in addition to a good representation of model operations.

A real-life case study of a riverbed excavation in a dam construction project is simulated using *ABSEMO* and the results are compared with those obtained from Discrete-Event Simulation (DES) models for verification. A percentage difference of 0.42% from the DES results was finally obtained, indicating that the model's logic and flow of resources are indeed accurate. The proposed Agent-Based (AB) methodology and the developed model aim at enhancing current practices of modeling earthmoving operations by looking at these operations from an individual Agent-Based (AB) prospective. This allows the capturing of realistic behaviors, through crafting agents' attributes, roles and interactions. The proposed methodology can be extended to general applications in construction management, where heterogeneity can be accounted for through replicating the different participants of construction projects in Agent-Based (AB) models as well as studying the emergent behavior of their interactions on the system.

## **ACKNOWLEDGEMENT**

Foremost, all praises and thanks are due to Allah for giving me the patience and determination to successfully accomplish my MAsc. program.

Furthermore, my sincerest gratitude goes to my supervisor Professor Tarek Zayed for the valuable comments, remarks and constant engagement throughout the learning process of this master's thesis. I would like to thank him for introducing me to the topic as well as for his appreciated support along the way. His kind gestures, tolerance and valued observations played a crucial role in the fulfillment of this work.

I wish to express my appreciation to my dear friend Karim Orabi for his continuous help and sharp remarks which assisted me in enhancing my model and strengthening its capabilities.

In addition, I would like to thank my loved ones, who have supported me throughout the entire process, by keeping me focused and helping me put all the pieces together. I will be forever grateful for my mother Sausan, my father Amer, my sister Hadoun and my brother Hassan for being there for me whenever I needed them and for being the wonderful family that they are.

# TABLE OF CONTENTS

<b>LIST OF FIGURES .....</b>	<b>ix</b>
<b>LIST OF TABLES .....</b>	<b>xi</b>
<b>LIST OF ABBREVIATIONS .....</b>	<b>xiii</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>1</b>
1.1    General Overview .....	1
1.2    Problem Statement .....	3
1.3    Research Objectives .....	4
1.4    Research Methodology.....	5
1.5    Thesis Organization.....	5
<b>CHAPTER 2: LETIRATURE REVIEW .....</b>	<b>8</b>
2.1    Chapter Overview .....	8
2.2    Simulation Techniques.....	9
2.2.1    Discrete-Event Simulation (DES).....	12
2.2.2    Limitations of the Discrete-Event Simulation Technique .....	14
2.2.3    System Dynamics (SD).....	16
2.2.4    Limitations of the SD Technique .....	26
2.3    Simulation in Construction Management.....	26
2.3.1    Discrete-Event Simulation (DES) Applications .....	27
2.3.2    System Dynamics Applications .....	31
2.3.3    Hybrid Simulation Applications .....	33
2.3.4    Earthmoving Simulation Practices.....	34
2.4    Agent-Based Modeling and Simulation (ABMS) .....	35
2.4.1    ABMS Methodology.....	36
2.4.2    ABMS Applications in Civil Engineering and Construction Management	44
2.5    Summary and Limitations of Literature .....	48
<b>CHAPTER 3: METHODOLOGY .....</b>	<b>50</b>

3.1	Chapter Overview .....	50
3.2	A Procedure for Creating AB Models for CM Applications .....	51
3.2.1	Environment Recognition .....	54
3.2.2	Participant Identification .....	55
3.2.3	Participants' Characteristics Determination .....	56
3.2.4	Process Understanding .....	57
3.2.5	Interaction Mechanism and Stages Identification .....	57
3.2.6	Characteristics Influence on Interaction .....	58
3.2.7	Agent Objects and Non-Agents Objects .....	59
3.2.8	Modeling Tool Determination .....	60
3.3	A Comprehensive AB Model for Earthmoving Operations .....	60
3.3.1	The Scope of the Model .....	61
3.3.2	The Creation of Agents .....	64

**CHAPTER 4: IMPLEMENTATION ..... 82**

4.1	Chapter Overview .....	82
4.2	Main Class .....	83
4.2.1	Earthmoving Environment .....	85
4.2.2	Agent Populations .....	86
4.2.3	Agents' Queues .....	87
4.2.4	Model Run Control .....	88
4.2.5	Results and Analysis .....	89
4.3	Agent Classes .....	91
4.3.1	The Bulldozer Agent .....	92
4.3.2	The Loader Agent .....	94
4.3.3	The Hauler Agent .....	97
4.3.4	The Spotter Agent .....	100
4.4	Graphical User Interface (Java Application) .....	101
4.5	Reporting of Results .....	106
4.6	System Verification .....	109
4.6.1	Case Study Description .....	110

4.6.2	Scope of Work .....	110
4.6.3	Fleet Selection and Configuration .....	112
4.6.4	DES Simulation Results.....	113
4.6.5	ABMS Simulation Results.....	115
4.6.6	Comparison between DES and ABMS Results .....	116
4.7	Superiority of ABMS .....	117
4.8	Implementation of ABMS in Other CM Areas .....	118
<b>CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS.....</b>		<b>120</b>
5.1	Summary and Conclusions.....	120
5.2	Research Contributions .....	122
5.3	Research Limitations.....	122
5.4	Future Work and Recommendations.....	123
<b>REFERENCES.....</b>		<b>125</b>
<b>APPENDIX A .....</b>		<b>132</b>
<b>APPENDIX B .....</b>		<b>137</b>



## LIST OF FIGURES

Figure 1-1: Research methodology.....	7
Figure 2-1: Literature review areas.....	8
Figure 2-2: Simulation of real world problems (Borshchev and Filippov 2004) .....	10
Figure 2-3: Simulation techniques (Alzraiee et al. 2012).....	11
Figure 2-4: DES bank service example (Borshchev and Filippov 2004) .....	13
Figure 2-5: System state update in DES and continuous simulation (Alzraiee 2013).....	16
Figure 2-6: CLD of ‘work to do’ for a typical construction operation (Alzraiee 2013)...	23
Figure 2-7: Stocks and Flow Diagramming (SFD) (Alzraiee 2013).....	24
Figure 2-8: System state computation in SD (Alzraiee 2013) .....	25
Figure 2-9: SFD and associated equations (Alzraiee 2013).....	25
Figure 2-10: A typical agent in AB models (Macal and North 2010) .....	39
Figure 2-11: AB simulation model of Conway’s Game of Life (Chan et al. 2010) .....	42
Figure 2-12: Boids simulation (initial random configuration) (Macal and North 2008) ..	43
Figure 2-13: Boids simulation (after 500 updates) (Macal and North 2008) .....	43
Figure 2-14: Agents in an infrastructure AB model .....	47
Figure 3-1: Methodology breakdown .....	51
Figure 3-2: Procedure for developing AB Models in construction management .....	52
Figure 3-3: Color legend for agents’ state charts.....	64
Figure 3-4: Bulldozer’s state chart.....	67
Figure 3-5: Loader’s state chart .....	72
Figure 3-6: Hauler’s state chart.....	78
Figure 3-7: Spotter’s state chart .....	80
Figure 3-8: Summary of agents’ interactions in the proposed earthmoving AB model ...	81
Figure 4-1: Implementation breakdown.....	83
Figure 4-2: Earthmoving Environment.....	85
Figure 4-3: Bulldozer’s state chart in <i>ABSEMO</i> .....	92
Figure 4-4: Loader’s state chart in <i>ABSEMO</i> .....	94
Figure 4-5: Hauler’s state chart in <i>ABSEMO</i> .....	97
Figure 4-6: Spotter’s state chart in <i>ABSEMO</i> .....	100

Figure 4-7: Welcome page of <i>ABSEMO</i> .....	103
Figure 4-8: Earthmoving environment in <i>ABSEMO</i> .....	103
Figure 4-9: Material input in <i>ABSEMO</i> .....	104
Figure 4-10: Equipment and labor input in <i>ABSEMO</i> .....	104
Figure 4-11: Model run control in <i>ABSEMO</i> .....	105
Figure 4-12: 2-D snapshot of model run in <i>ABSEMO</i> .....	105
Figure 4-13: 3-D snapshot of model run in <i>ABSEMO</i> .....	106
Figure 4-14: Equipment utilization statistics in <i>ABSEMO</i> .....	107
Figure 4-15: Bulldozers detailed utilization statistics in <i>ABSEMO</i> .....	107
Figure 4-16: Loaders detailed utilization statistics in <i>ABSEMO</i> .....	108
Figure 4-17: Haulers detailed utilization statistics in <i>ABSEMO</i> .....	108
Figure 4-18: Spotters detailed utilization statistics in <i>ABSEMO</i> .....	109
Figure 4-19: Location of Sainte-Marguerite-3 dam (Hydro Quebec 2003).....	111
Figure 4-20: Sainte-Marguerite-3 dam (Hydro Quebec 2003) .....	111
Figure 4-21: EZStrobe DES model of riverbed excavation in the Sainte-Marguerite dam construction project (Alzraiee 2013) .....	113

## LIST OF TABLES

Table 2-1: DES chronological list example .....	14
Table 2-2: Denotations for Causal Loop Diagramming (Sterman 2000) .....	21
Table 2-3: CYCLONE modeling elements (Halpin 1976) .....	28
Table 2-4: Literate review on SD applications in construction management .....	32
Table 3-1: Earthmoving operations’ participants and their properties .....	56
Table 3-2: Proposed AB model participants .....	62
Table 3-3: Agents’ attributes and variables in the proposed AB model .....	63
Table 3-4: Bulldozers’ attributes and variables .....	66
Table 3-5: Bulldozer’s state chart transitions .....	68
Table 3-6: Loader’s attributes and variables .....	70
Table 3-7: Loader’s state chart transitions .....	74
Table 3-8: Hauler’s attributes and variables .....	77
Table 3-9: Hauler’s state chart transitions .....	79
Table 3-10: Spotter’s attributes and variables .....	80
Table 3-11: Spotter’s state chart transitions .....	81
Table 4-1: AnyLogic elements used in <i>ABSEMO</i> .....	84
Table 4-2: Agent populations’ elements in <i>ABSEMO</i> .....	86
Table 4-3: Agent queues’ elements in <i>ABSEMO</i> .....	87
Table 4-4: Model run control in <i>ABSEMO</i> .....	88
Table 4-5: Results and Analysis in <i>ABSEMO</i> .....	89
Table 4-6: Bulldozer class elements in <i>ABSEMO</i> .....	92
Table 4-7: Loader class elements in <i>ABSEMO</i> .....	95
Table 4-8: Hauler class elements in <i>ABSEMO</i> .....	98
Table 4-9: Spotter class elements in <i>ABSEMO</i> .....	100
Table 4-10: Scope of work in in the SM-3 dam construction earthmoving operations (Alzraiee 2013) .....	112
Table 4-11: Hauler and loader fleet configurations in the SM-3 dam construction earthmoving operations (Alzraiee 2013) .....	114

Table 4-12: Spread and compact equipment characteristics in the SM-3 dam construction earthmoving operations (Alzraiee 2013) .....	114
Table 4-13: A comparison between DES and ABMS Results for the SM-3 Riverbed Excavation.....	116
Table A-1: Agent populations' elements in <i>ABSEMO</i> (Java code) .....	132
Table A-2: Agent queues' elements in <i>ABSEMO</i> (Java code).....	134
Table A-3: Model run control in <i>ABSEMO</i> (Java code).....	134
Table A-4: Results and Analysis in <i>ABSEMO</i> (Java code).....	135
Table A-5: Bulldozer class elements and transitions in <i>ABSEM</i> (Java code).....	137
Table A-6: Loader class elements and transitions in <i>ABSEMO</i> (Java code) .....	143
Table A-7: Hauler class elements and transitions in <i>ABSEMO</i> (Java code).....	152
Table A-8: Spotter class elements and transitions in <i>ABSEMO</i> (Java code) .....	162

## LIST OF ABBREVIATIONS

AB	Agent-Based
ABMS	Agent-Based Modeling and Simulation
ABSEMO	Agent-Based Simulator for Earthmoving Operations
AI	Artificial Intelligence
CLD	Causal Loop Diagram
CM	Construction Management
CYCLONE	Cyclic Operation Network
DES	Discrete-Event Simulation
DISCO	Dynamic Interface Simulation for Construction Operation
FIFO	First-In-First-Out
GUI	Graphical User Interface
HiSim	Hybrid Simulation
HSM	Hierarchical Simulation Modeling
KEYSTONE	Knowledge Discovery Based Simulation System
RBM	Resource-Based Modeling
SD	System Dynamics
SFD	Stocks and Flows Diagram
SM-3	Sainte Marguerite Three
WBS	Work Breakdown Structure

# CHAPTER 1: INTRODUCTION

## 1.1 General Overview

Primarily, simulation is the attempt to imitate a real-life or hypothetical situation (Banks et al. 2000), which provides a powerful tool to experiment and plan operations before the actual execution. The latter is done in an effort to avoid time delays and cost overruns. Creating a successful simulation model often leads to more realistic planning and more accurate estimation of the required resources. One of the most common applications of simulation in the construction industry is the simulation of earthmoving operations. Since these operations are typically lengthy in duration and fall on the critical path of construction projects, accurate planning is crucial in ensuring the success of such projects. In addition, earthmoving operations are often considered equipment-intensive, utilizing large fleets of trucks, loaders, bulldozers, etc.

Due to time and cost constraints, an efficient use of resources is a vital objective for contractors in projects involving such operations (Moselhi and Alshibani 2009). The cyclic nature of earthmoving operations and the type of work tasks involved in such operations make the simulation process a valid planning tool where good forecasting of productivity and costs are expected (Touran 1990). There are numerous applications of simulation of earthmoving operations available in literature.

The most common simulation techniques are Discrete Event Simulation (DES), System Dynamics (SD) and Agent-Based Modeling and Simulation (ABMS). Although these three types have all been used in construction management applications, the application of simulation is still limited in this field (Marzouk and Moselhi 2003). In

regards to DES, it is by far the simulation technique with the most applications in construction management. This is due to the fact that DES was the first to be introduced to construction operations (Halpin 1977). Furthermore, the nature of DES provides, in most cases, a sufficient solution to modeling most construction operations, especially on the operational and technical level. To address the decision-making aspect of construction management, recent efforts suggested the utilization of SD to address the complex strategic level when simulating construction operations (Alzraiee et al. 2012).

In this thesis, the simulation technique that is going to be used will be of the third type, which is ABMS. The latter is the newest simulation field (Schelling 1971) that offers solutions to overcome many limitations of current simulation practices. ABMS is based on the idea of simulating the interactions of autonomous objects, in order to identify, explain, generate and design emergent behaviors (Chan et al. 2010). Basically, ABMS is a bottom-up modeling approach, where individual participants of the operation are modeled and given attributes and roles to reach an emergent behavior of the whole system (Macal and North 2008). Conveniently, the main advantage of using ABMS is the ability to create a system of smart agents that adapt with varying conditions and act accordingly to capture the real behavior of the system being studied. The latter is done without the need to make assumptions or direct the model in a way which is acceptable by the capacities of the simulation technique. ABMS is gaining the interest of many in multiple fields, especially those related to supply chains and consumer behavior (Garcia 2005). This is due to the fact that the capabilities of ABMS fit quite well the requirements of modeling individuals' behavior in consumer markets. However, the applications of ABMS in construction management are still limited and few in number (Ren and

Anumba 2003, Tah 2004, Bernhardt and McNeil 2008, Min and Bjornsson 2008, El-Adawy and Kandil 2009, Osman 2012).

## **1.2 Problem Statement**

There is a need to incorporate Agent-Based (AB) technologies with significant construction operations such as earthmoving. This must be completed in a smart and flexible paradigm that accepts various types of data, maintains a presentable view of model operation and analysis and produces accurate results. This research aims at rebuilding the methodology of creating earthmoving simulation models, based on the AB approach.

While the application of simulation in earthmoving remains a well-researched area (Touran 1990, Oloufa 1993, Martinez 1998, AbouRizk and Hajjar 1998, Smith et al. 2000), all previous work is based solely on a DES or on a DES-SD hybrid system (Alzraiee et al. 2012). Even presently, simulation models often lack graphical modeling support, which is the key of model definition and manipulation (Hajjar et al. 2002). In addition, building and validating SD models is time consuming, requires modeling skills and is heavily dependent on conceptual models as well as the availability of detailed data (Alzraiee 2013).

Other limitations arise from the fact that the developed models behave in a predetermined manner, mainly due to the nature of DES and SD. Although various efforts succeeded in overcoming some of these limitations, these efforts were scattered in various research works and were never gathered in one complete system. Consequently,



this is where ABMS comes into play as a modeling tool that can deliver a comprehensive system for modeling and simulating earthmoving operations.

Drawbacks and limitations of current research can be briefly encapsulated as follows:

- Earthmoving models are often inflexible in accepting different types of data; such as the case when having trucks' capacities that are not multiples of loaders' capacities.
- Often, earthmoving models have a predetermined behavior. This can be clearly observed in the truck loading process, where trucks are always loaded to their full capacities.
- Most earthmoving models are inflexible in modeling equipment capacities and properties. For instance, when having two or more trucks with different capacities in the same earthmoving model, properties cannot be directly established.
- Earthmoving models, similar to other DES-based models, require visualization when implementing. As the earthmoving operation becomes more complicated, its visualization becomes more difficult.
- Hybrid DES-SD models that rely on DES for the operational level will have the same previous technical limitations, in addition to the SD part requiring detailed sets of data, large amount of time and specialized modeling skills.

### **1.3 Research Objectives**

The general objective of this thesis is the introduction of ABMS into earthmoving operations for the purpose of overcoming limitations of current research work and

enhancing current practices. This general objective can be broken down into the following sub-objectives:

- Generate a procedure for creating AB models, which can be used for construction management applications.
- Develop a detailed AB model for earthmoving operations, which captures the properties and interactions of model elements.
- Design a stand-alone ABMS software system for earthmoving operations and verify the model using a real-world case study.

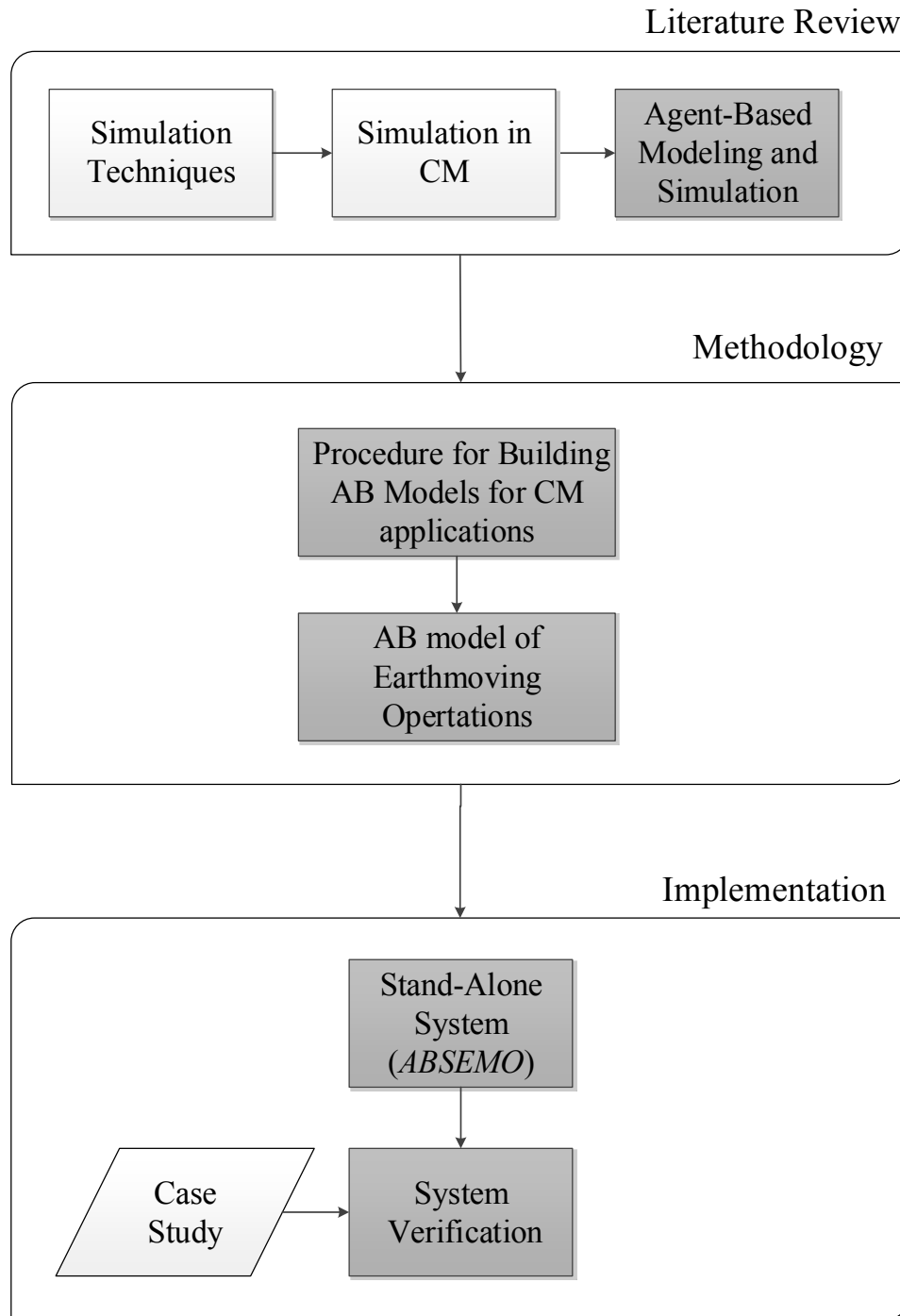
#### **1.4 Research Methodology**

A generic flow chart that summarizes the research methodologies and major tasks is depicted in Figure 1-1. Major components and contributions are highlighted in grey. Primarily, the literature will be reviewed on the basis of different simulation techniques, applications of simulation in construction management and the ABMS technique. Afterwards, the methodology section, which presents a generic procedure for building AB models for construction management applications and a comprehensive AB model for earthmoving operations, will be outlined in detail. Finally, the implementation of the proposed model will be presented and followed by the verification of the implemented system.

#### **1.5 Thesis Organization**

This thesis is comprised of five chapters. Chapter 1 is an introduction to the topic which includes the problem statement, research objectives and the thesis organization.

Chapter 2 summarizes the literature review on the state-of-the-art techniques related to this research work. It discusses areas related to computer simulation, applications of simulation in construction management, current earthmoving simulation techniques and an elaborate review on the AB methodology. In Chapter 3, the methodology of producing AB models is provided, followed by the development of the proposed earthmoving simulation model. Chapter 4 represents the implementation of the proposed model in a stand-alone software application, while explaining the various elements of the application and their uses. Finally, Chapter 5 outlines the conclusions and recommendations of this research work at-hand.

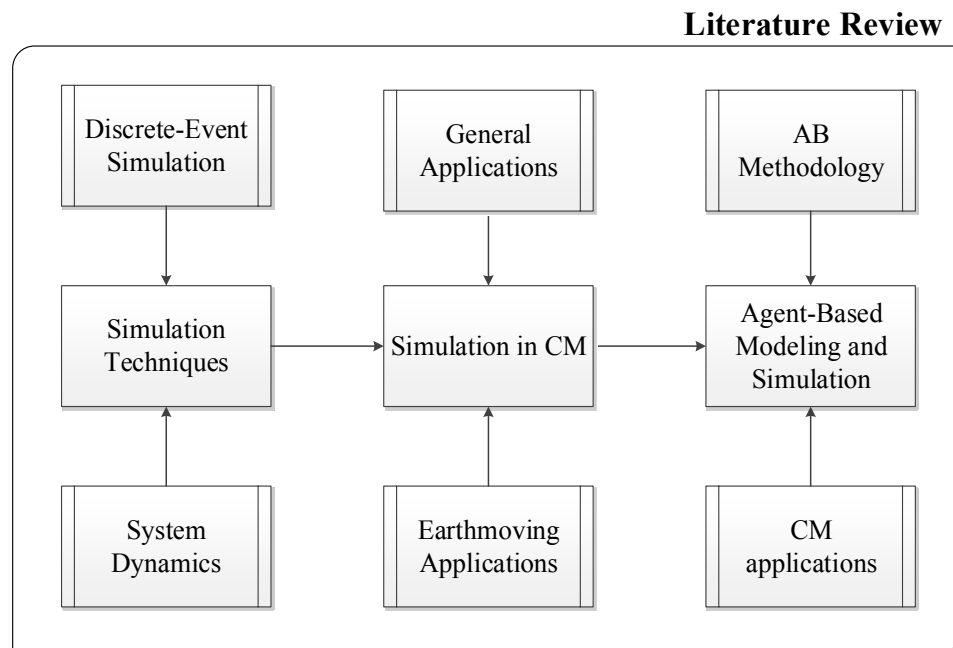


**Figure 1-1:** Research methodology

## CHAPTER 2: LITERATURE REVIEW

### 2.1 Chapter Overview

This chapter is dedicated to summarizing the literature that was visited prior to the model development and implementation. It mainly aims at highlighting different simulation techniques, going over simulation practices used in construction management in general and earthmoving operations in particular and explaining the AB Methodology in detail. Figure 2-1 depicts the different literature review areas of this thesis.



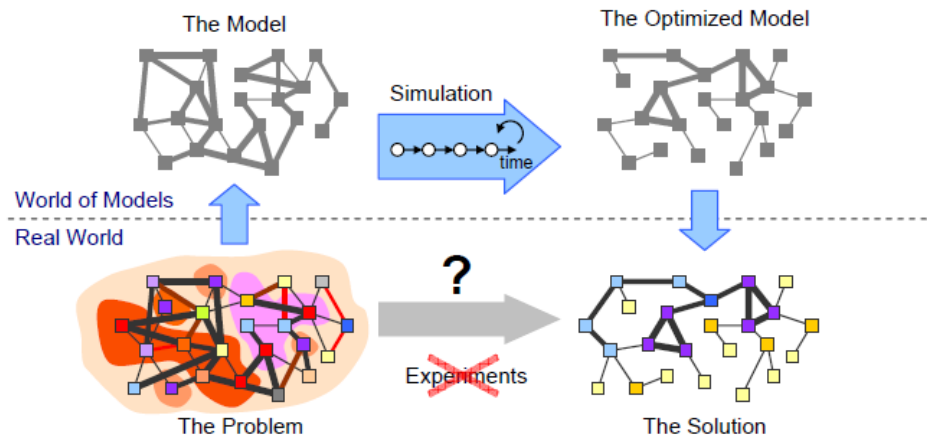
**Figure 2-1:** Literature review areas

Section 2.2 gives an overview about different simulation techniques. It discusses Discrete-Event Simulation (DES) and its limitations, as well as System Dynamics (SD) and its limitations. Furthermore, Section 2.3 demonstrates the application of simulation in construction management. It includes the applications of DES, SD and Hybrid DES-SED

in the construction industry. In addition, this section demonstrates the current practices of simulating earthmoving operations as well as some general and special purpose simulation software. In Section 2.4, the ABMS methodology is thoroughly explained. Major AB components, such as agents, environments, interactions and emergence, are broken down and explained in detail. Also, some ABMS applications, in construction management, are highlighted. Section 2.5 summarizes the literature review and points out some gaps and limitations of current research.

## **2.2 Simulation Techniques**

Different areas have witnessed the application of simulation including the manufacturing, industrial, environmental and construction fields (Banks et al. 2000). Simulation was defined by Shannon (1992) as *“the process of designing a model of a real system and conducting experiments with this model for the purpose of either understanding the behavior of the system and/or evaluating various strategies for the operation of the system.”* Hence, simulation is a preplanning tool to help regulate resources, mitigate risk and forecast durations and costs. In relation, Figure 2-2 demonstrates how simulation can help reach a solution for a real-world problem without the need of experimenting.



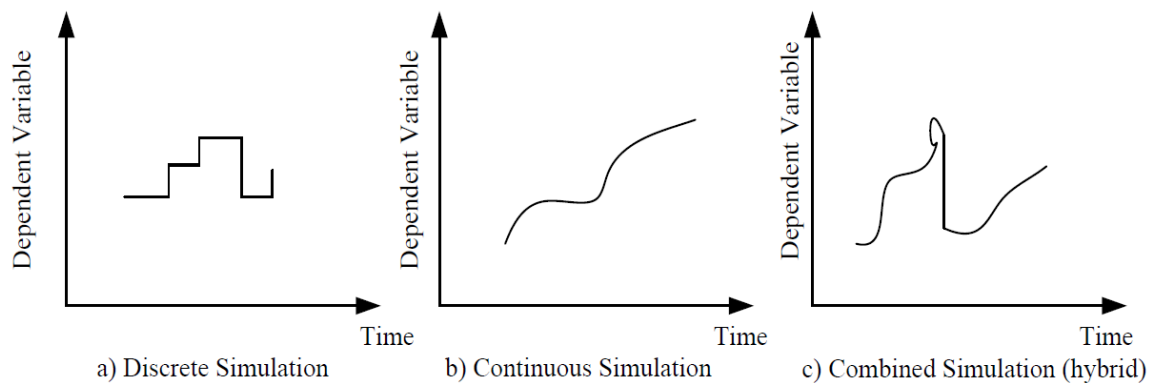
**Figure 2-2:** Simulation of real world problems (Borshchev and Filippov 2004)

Computer simulation refers to simulation models being run on computers to acquire different sets of analysis and results. Simulation models can be classified according to various independent pairs of attributes. The most famous of the latter are:

- Deterministic or stochastic; deterministic models are based on constant variables, while stochastic models include randomness. One of the most crucial uses of simulation is its ability to account for randomness and inaccuracy. This is why complex simulation models mostly fall under the stochastic type.
- Static or dynamic; static models are those that represent static environments, while dynamic models are those characterizing dynamic operations. The simulation of construction operations is clearly an example of dynamic simulation.
- Discrete or continuous; this is the most important classification when it comes to computer simulation models. It is related to the changing nature of variables with respect to time during the simulation run. The change in these variables over the simulation time can be discrete, continuous or a combination of both.

When variables change in a discrete manner, the system is referred to as a Discrete-Event Simulation (DES) System. A DES model depicts the operation of a system as a sequential order of events called the chronological list. In DES, a change in a variable occurs as an event at a specific point in time, marking a change of state in the system (Halpin and Riggs 1992). On the other hand, in continuous simulation models, variables change over a period of time and not instantaneously at specific points in the simulation run time. This change is performed using a set of mathematical equations.

Also, in DES models, time progresses by time steps; while in continuous simulation models, time advances constantly. In combined simulations systems, changes in variables occur instantaneously at certain simulation points in time and continuously at others. Hence, time in such systems advances by steps or constantly depending on the change type of variables. SD is an example of a continuous simulation technique. Figure 2-3 demonstrates the change in the dependent variable with time in discrete, continuous as well as combined simulation.



**Figure 2-3:** Simulation techniques (Alzraiee et al. 2012)



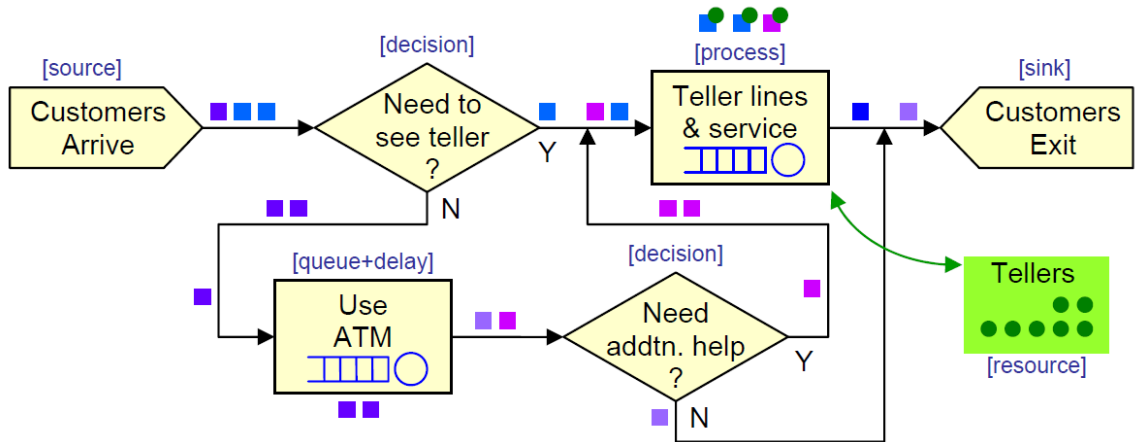
### **2.2.1 Discrete-Event Simulation (DES)**

The DES technique focuses on a list of events which occur instantaneously, changing the state of the system. This same list is updated every time an event occurs. DES models a system as a network or a flow diagram of a collection of queues and processes, where changes in the system occur at distinct points in time (Brailsford and Hilton 2001). Overall, in DES, entities involved in the process being modeled are treated as passive objects. They can represent people, equipment, organizations, documents, tasks, messages, etc.

These entities travel through the DES flowchart where they stay in queues, get processed and then release resources (Borshchev and Filippov 2004). Basically, an entity flows through a DES model, seizing resources to perform different tasks, and releasing these resources once the work task has been completed. But, if these resources are busy and unavailable when the entity requires them to complete the work task, the entity will pause and be delayed in a queue until the required resources become available again. Once an event is performed, an object called the simulation clock steps right to the time when that event was accomplished. Afterwards, this information is stored in a chronological list based on the order of occurrences.

Figure 2-4 illustrates an example of a DES system. The model describes the cycle a customer undergoes, from the time they enter into the bank until the time they leave. It demonstrates how customers pass through the flow diagram based on the needs of their visit. Moreover, tellers here are used as resources which customers can seize when they require a teller service. However, the authors chose not to use ATMs as resources;

instead, ATMs are just implanted in the system in a form of vacant spaces that can be filled with customers. It is also clear in this model how queues are embedded in processes instead of being separated in two different entities. The latter are all conventions of the authors. DES models can be built in different ways as shall be seen later in this chapter.



**Figure 2-4:** DES bank service example (Borshchev and Filippov 2004)

As an example, a chronological list of this bank's operations is depicted in Table 2-1. As can be observed in the table, events are ranked based on their order of completion. During twelve minutes of the bank service, three customers have arrived, one customer has used the ATM, two customers have used the teller service and one customer has exited the bank. This is a very simple example of a chronological list, which is demonstrated for the purpose of understanding how the technique works and how entities travel and seize resources in simulation.

**Table 2-1:** DES chronological list example

<b>Activity</b>	<b>Duration</b>	<b>T<sub>end</sub> (min)</b>
Customers Arrive	0	0
Use ATM	3	3
Teller service	5	8
Customers Arrive	0	8
Customers Exit	1	9
Customers Arrive	0	9
Teller service	4	12

### **2.2.2 Limitations of the Discrete-Event Simulation Technique**

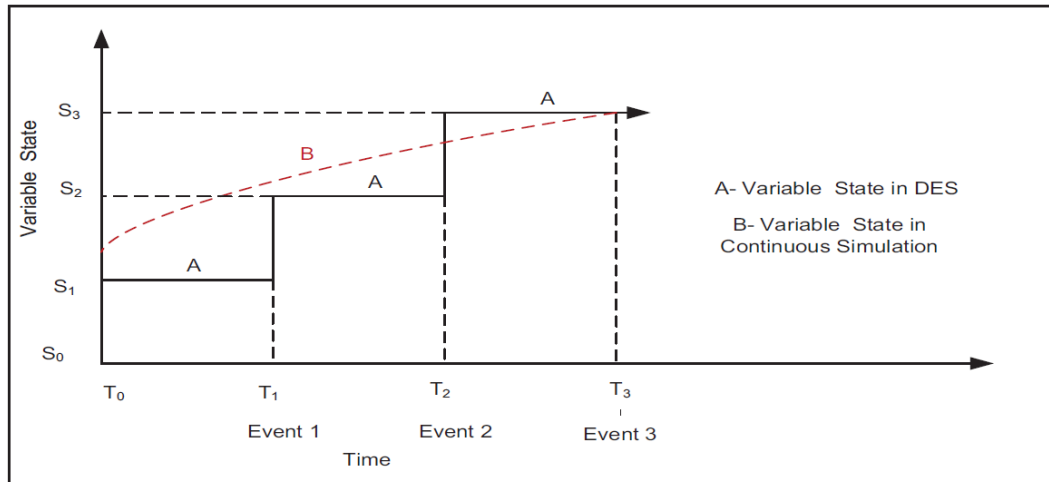
Since DES is a simulation approach that involves the process of creating a computer model and applying the DES methodology to it, limitations of both the technique itself and the developed model are expected. As a simulation technique, DES has some limitations which were previously highlighted by researches. One major limitation of DES is that it cannot quite capture the external factors that influence the operation that is being modeled.

Also, DES does not have the aptitude of determining the modeled system's stability in the surrounding environment. Hence, the performance of the system could, due to its surrounding environment, be driven by hidden causal relationships that could be non-linear (Helal 2008). This is because the strategic aspect of any operation is often qualitative and continuous in nature, in turn creating challenges when applying DES to model such an aspect (Zulch et al. 2002).

Other researchers state that DES encounters a lot of difficulties in accounting for the overall strategy of the project or operation being modeled, due to its time step advancement mechanism (Martin and Raffo 2001). In addition, even on the operational level, DES fails to deliver a clear representation of the state of the system between consecutive events and time steps. Alzraiee (2013) demonstrated the latter through the example shown in Figure 2-5. This figure portrays three consecutive events ( $E_1$ ,  $E_2$  and  $E_3$ ) that occur at three sequential times ( $T_1$ ,  $T_2$  and  $T_3$ ). At the starting point of the simulation, time is at zero ( $T_0$ ) and the state of the system is  $S_0$ . When event  $E_1$  occurs, the simulation clock is advanced to  $T_1$  and the system's state changes to  $S_1$ .

Similarly, the occurrence of events  $E_2$  and  $E_3$  at times  $T_2$  and  $T_3$  changes the system state to  $S_2$  and  $S_3$  consecutively. The purpose of illustrating this figure is to highlight the state of the system in DES between two different but consecutive states. The zero-slope line (A) between events  $E_1$  and  $E_2$  and events  $E_2$  and  $E_3$  demonstrates how the state of the system between two consecutive events in DES remains constant. The state of the system after the occurrence of event  $E_1$  becomes  $S_1$  and stays this way until event  $E_2$  occurs. The latter is also the case between events  $E_2$  and  $E_3$ . So, the state of the system

between times  $T_0$  and  $T_1$ ,  $T_1$  and  $T_2$ , and  $T_2$  and  $T_3$  is not updated. As an alternative, the state of the system is only updated once the next scheduled event occurs.



**Figure 2-5:** System state update in DES and continuous simulation (Alzraiee 2013)

On the other hand, this major limitation of DES, which is associated with the behavior of the technique itself, limits the full understanding and representation of the actual state of the system being modeled and the interaction between the system's entities and parameters. Correspondingly, this limitation can lead to delays in taking corrective measures due to any deviation that may occur based on what is planned to happen in the operation (Alzraiee 2013). Curve B forms an expected representation of the actual state of the system without adhering to the time step principle of DES.

### 2.2.3 System Dynamics (SD)

In continuous simulation, change is governed by a set of differential equations, which can be formulated in a conceptual model that represents the system on an abstract level. Continuous simulation is often deterministic, simple and requires less data in order to develop a working model. However, the more complex the continuous simulation

model becomes, the more complex the differential equations associated to it become (Alzraiee 2013).

The most famous type of continuous simulation is System Dynamics (SD). Unlike DES, variables involved in an SD simulation model change in a continuous fashion. Thus, the state of the system changes in a continuous fashion as well. SD was first introduced by Jay Forrester in 1961 in a book titled “Industrial dynamics” (Forrester 1961). It was then considered as a new modeling and planning approach aiming at solving complex problems in social systems related to the industrial sector. Also, SD is based on the principle that the overall behavior of the system is determined by its structure. For instance, systems requiring holistic consideration as well as feedback loops among their participants have been successfully modeled using SD (Alzraiee 2013). SD has been used in many disciplines, especially those related to social, economic, engineering, environmental, as well as management problems (Wolstenholme 1990). Researchers believe that SD constitutes an elaborate extension on continuous simulation aiming at addressing the complexity of systems and nonlinearity of feedback processes. It is basically an approach utilized to solve problems at top management levels (Sterman 2000, Lyneis 2001).

Involving both “soft” and “hard” data, considering the project as a whole rather than a sum of individual elements, examining non-linear scenarios of element interaction and the failure of the conventional DES tools to address the project management aspect of systems are all motivations behind utilizing SD as a modeling tool for different applications (Rodrigues and Bowers 1996, Sterman 2000).

Sterman (2000) developed a methodology of five steps for modeling a system using SD. Firstly; it begins with understanding the system, where the modeler should break the system into smaller components for a better visualization and understanding of its overall behavior. The second step is conceptualization, where feedback loops are identified and theories related to the behavior of the system at-hand are studied. The third step is the formulation of the simulation model at-hand. The fourth step is testing the model in order to verify it. The fifth and final step is the design and evaluation of the policy, which deals with the environmental conditions and new decision rules and strategies.

The most important terms in SD are Model Boundary, Causal Loops Diagrams (CLDs), which capture the conceptual relationships of variables in the system and finally Stocks and Flows Diagrams (SFDs), which define the movement of entities and its rate from start to end in the model. To understand how SD works, each of these terms is explained individually.

### **Model Boundary:**

The behavior of the system in SD models is generated within a closed boundary of the feedback process. In order to define this boundary in an SD model, it is required to identify the behavior of interest, which can be extracted from the purpose of the model. The components of interaction that are necessary to generate this behavior have to be identified and selected. In order to summarize the scope of the model in SD, key variables should be classified as endogenous, exogenous or excluded (Alzraiee 2013).

On a similar note, endogenous variables are the most important variables in an SD model. The latter usually portray the dynamic inherent in systems. Endogenous variables exist in a casual-effect structure in SD models and their values are determined by the state of other variables in the system. Exogenous variables, on the other hand, are variables external to the model and are not represented by the model's feedback structure. Exogenous variables are also involved in casual-effect structure. However, unlike endogenous variables, the value of exogenous variables is independent from the state of other variables; their values are determined by variables external to the system. In other words, the system's internal interactions will have no influence on exogenous variables.

Excluded variables are variables beyond the scope of the SD model. These variables are not included in the causal-effect feedback of the structure. It is usually better to exclude such variables from the model in order to keep the model simple and comprehensible. For example, in an SD model of a typical construction project, fatigue, overtime required and error are all considered as endogenous variables. A project's planned duration and planned productivity are considered as exogenous variables. Other unrelated variables to the operation, such as the number of site engineers or the traffic condition around the site, are considered as excluded variables (Alzraiee 2013).

### **Causal Loops Diagrams (CLDs):**

The major strength of an SD model lies in its feedback loops, which are the main reason behind the dynamic behavior in a system (Sterman 2000). CLDs, in essence, conceptualize the feedback structure of the system as being understood by the modeler (Richardson and Pugh 1981). In addition, CLD systems can be classified according to



two independent pairs of attributes. The first classification is based upon the outputs' influence on the inputs in the system, while the second classification is related to the tendency of the CLD to stabilize or destabilize the system itself (Sterman 2000).




In the first classification, when the system's outputs are responsive towards their inputs but have no influence on them, it is referred to as an open CLD. On the other hand, outputs in a closed CLD both respond to and influence their inputs. In the other classification, a CLD can be either positive or negative. Positive CLDs consist of a series of causal relationships which signify a self-reinforcing process and amplify results. On the other hand, negative CLDs consist of a series of causal relationships that aim at directing the operation to a specific goal value (Sterman 2000).

Table 2-2 demonstrates how a causal link containing two variables, 'A' and 'B', can be positive or negative. In the first row, it is demonstrated that an increase in the value of 'A' causes an increase in the value of 'B', and a decrease in the value of 'A' causes a decrease in the value of 'B'. In this scenario, 'A' has a positive impact on 'B' and a positive sign is marked at the end of the arrow. In the second row, an increase in the value of 'A' causes a decrease in the value of 'B', and a decrease in the value of 'A' causes an increase in the value of 'B'. In this scenario, 'A' has a negative impact on 'B' and a negative sign is placed at the end of the arrow. What can be retained from this is that changes in variables can be mathematically computed through the integration of variables' rate of change.

The third row of the table is dedicated to explain an important concept in SD, which is *delay*. In SD, *delay* is a term used to describe a process whose outputs lags

behind its inputs. Plus, *delays* represent crucial sources of dynamic behavior in systems and they should be managed in order to avoid creating instability (Sterman 2000). In Table 2-2, delay is represented by parallel lines placed on the causal link between ‘A’ and ‘B’.

**Table 2-2:** Denotations for Causal Loop Diagramming (Sterman 2000)

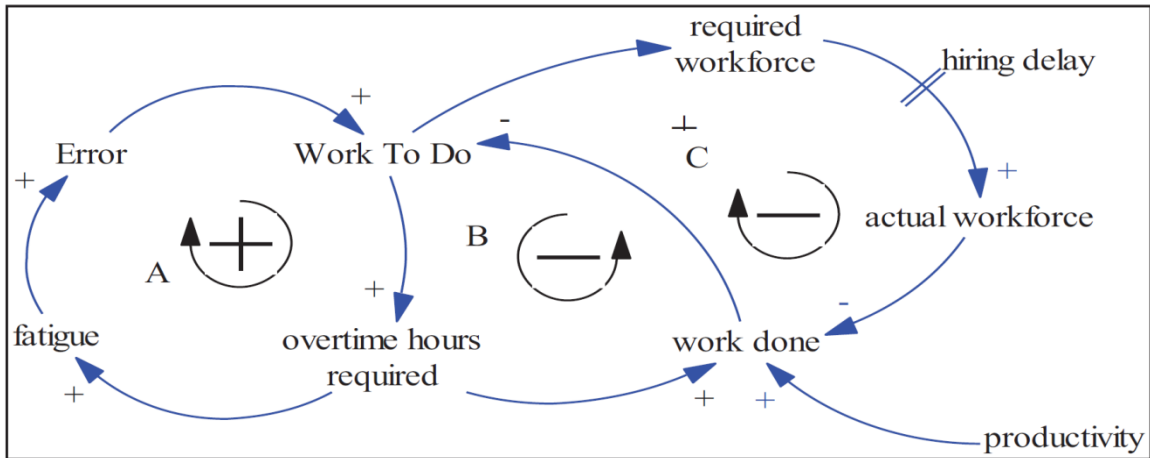
Causal Link	Description	Mathematical Formulation
	All else remaining equal, if variable ‘A’ increases (decreases) then ‘B’ increases (decreases) in variable above (below)	$\partial B / \partial A > 0$ in this case of accumulations, $B = \int_{t_0}^t (A + \dots) ds + B_{t_0}$ (2.1)
	All else remaining equal, if variable ‘A’ increases (decreases) then ‘B’ decreases (increases) in variable below (above)	$\partial B / \partial A < 0$ in this case of accumulations, $B = \int_{t_0}^t (-A + \dots) ds + B_{t_0}$ (2.2)
	Significant time delay is involved in implementing the causal relationship between the variables ‘A’ and ‘B’	

Alzraiee (2013) presented the CLD shown in Figure 2-6 of ‘work to do’ in a typical construction operation. As can be seen in the figure, three causal loops exist in the operation. ‘A’ is a reinforcing (positive) loop, while ‘B’ and ‘C’ are both balancing (negative) loops. The variables contained in loop ‘A’ are ‘work to do’, ‘overtime hours required’, ‘fatigue’ and ‘error’. As the ‘work to do’ variable increases, overtime hours become essential to meet the schedule of this operation and the overall deadline of the project. This increase in ‘overtime hours required’ will cause an increase in ‘fatigue’, which in turn causes an increase in ‘error’. Going back to the starting point of this loop, the increase in ‘error’ will require some rework, which will add to the initial scope of work causing an increase in ‘work to do’. Furthermore, multiplying the signs of all

variables in loop 'A' (+, +, +, +) leads to a positive sign, which represents the overall positive polarity of the loop.

Loop 'B' consists of 'work to do', 'overtime' and 'work done'. In loop 'A', an increase in 'work to do' causes an increase in 'overtime hours required'. Of course, an increase in 'overtime hours required' entails an increase in 'work done'. However, an increase in 'work done' means that parts of the scope of work are accomplished, decreasing 'work to do'. The latter is what causes the negative polarity of loop 'B', as multiplying the signs of its variables (+, +, -) leads to an overall negative sign. In the CLD displayed in Figure 2-6, loops 'A' and 'B' counter influence each other, as they have different polarities. Hence, it is clear that loop 'A' tends to increase the 'work to do' variable, while loop 'B' tends to decrease it. In this case, since it is desirable to have less 'work to do', it is the responsibility of decision makers to decrease the effect of loop 'A' and in turn facilitate the conditions surrounding loop 'B' (Alzraiee 2013).

The previously discussed concept of *delay* in SD models can be observed in loop 'C', which is of a negative polarity similar to loop 'B'. In the situation where the 'required workforce' variable increases due to an increase in 'work to do', the management has to hire workforce to accommodate the extra work packages and recover from schedule slippage. Consequently, this hiring process has to pass through different stages and the new workforce has to be trained. This is when *delay* occurs, as the decision to hire requires time to affect the productivity. The Duration between the decision to hire more workforce and the noticeable effect of that decision is *delay*, represented by the parallel-lines arrow in Figure 2-6.



**Figure 2-6:** CLD of 'work to do' for a typical construction operation (Alzraiee 2013)

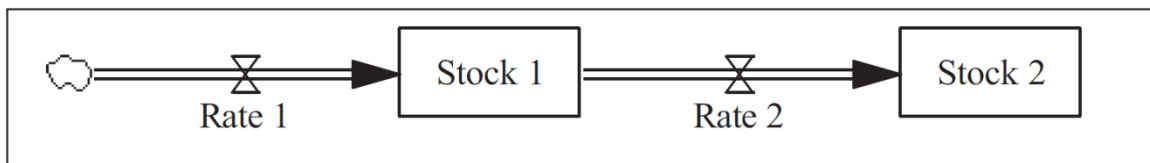
Although CLDs are considered practical in representing variable interdependencies and the feedback process, on their own, they cannot capture the stocks and flows of an SD system, which in turn represents the rate at which variables change in the model (Sterman 2000). This is exactly where the need for Stocks and Flows Diagrams (SFDs) arises.

### **Stocks and Flows Diagrams (SFDs):**

SFDs are generally generated from CLDs (Sterman 2000). SFDs are the reason behind the dynamic behavior of SD models. The term *stock* in SD refers to an accumulation characterizing the state of the system and generating information which is the basis of decisions and actions. The mechanism by which a *stock* can change is through *flows*. A *stock* accumulates the difference between inflows going in the *stock* and outflows leaving the *stock*, and this is how delay is created in the system. This explains why *stocks* are modeled by the mathematical integration of the sum of inflows and outflows (Alzraiee 2013). In SFDs, *flows* influence and control the level of accumulation at *stocks*. A *Flow* is basically a rate that represents the source of disequilibrium in SD

(Sterman 2000). SFDs are added to the CLDs in SD models in order to improve the clarity of the model schema (Alzraiee 2013).

An SFD consists of three elements which are *stocks* represented by rectangles, *flows* represented by pipes with valves pointing at the *stocks* and sources of inflow to or outflow from the model represented by clouds. Clouds are related to the model boundary, as model inputs before the cloud and model outputs after the cloud are considered outside the boundary of the model. Figure 2-7 represents a typical SFD.

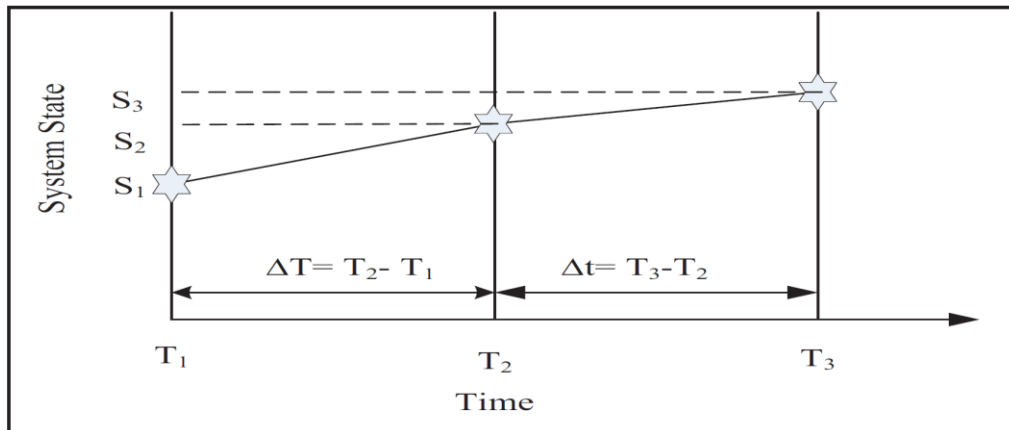


**Figure 2-7:** Stocks and Flow Diagramming (SFD) (Alzraiee 2013)

To control the mechanism of *flows* in SD, the hydraulic principle was utilized. This principle states that the accumulation of water in a reservoir during a specific time period is equal to the quantity of water that flows into the reservoir subtracted from the quantity of water flowing out of the reservoir (Forrester 1961). Applying this principle to SFDs, the net *flow* into a *stock* becomes equal to the rate of change of that same *stock*. This can be applied by using a set of mathematical equations that describe the model. *Stocks* are given initial values which change during the model run. Decision makers monitor the system's performance over time by constantly observing the updated *stocks* and *flows*. Therefore, they can change strategies accordingly (Alzraiee 2013).

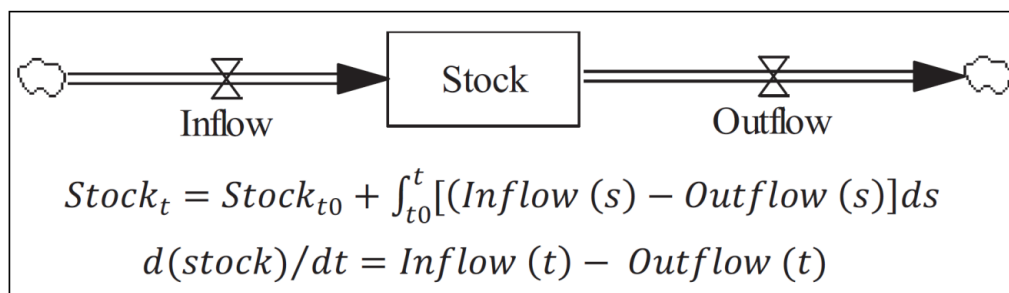
To explain how mathematical equations are used in order to represent SFDs, the mechanism of time elapsing in SD must be understood. Simulation time has to be broken

into small equal time intervals, as displayed in Figure 2-8. At the beginning of the simulation ( $T_1$ ), the system's state is  $S_1$ . At  $T_2$ , the system's state updates to  $S_2$ , which is the summation of the original  $S_1$  state and the net flow between  $T_1$  and  $T_2$ . Likewise, the  $S_3$  state at  $T_3$  is the summation of the  $S_2$  state and the net flow between  $T_2$  and  $T_3$ .



**Figure 2-8:** System state computation in SD (Alzraiee 2013)

Figure 2-9 represents a simple *stock* with inflow and outflow and its associated mathematical equations. The *stock* level at any given time ( $t$ ) is equal to the initial *stock* level at the initial time ( $t_0$ ) in addition to the net *flow* from the initial time ( $t_0$ ) up until that time ( $t$ ). This is clearly represented by the first equation in the figure. Similarly, the net rate of change of that *stock* at any given time ( $t$ ) is the derivative of inflows less outflows, which is represented by the second equation.



**Figure 2-9:** SFD and associated equations (Alzraiee 2013)

#### **2.2.4 Limitations of the SD Technique**

As in DES, SD has a number of known limitations that arise from the nature of the method. Most criticisms towards SD are based on the fact that SD, alone, is not able to capture a detailed view of the quantitative operational aspects of the systems being modeled.

SD is usually used to model systems with the purpose of having a holistic view and control over system variables. This is why SD is believed to have a wider focus on the system being modeled, by viewing it as general and abstract when compared to the narrowly focused DES (Lane 2000). Alzraiee (2013) states that SD models are unable to represent in detail the operational aspects of the systems being modeled.

Another drawback of SD is the matter of randomness. SD, in nature, is a more deterministic modeling approach, compared to stochastic modeling techniques such as DES (Meadows 1980). This is due to the simulation mechanism of SD that relies on mathematical differential equations which are deterministic in nature. Finally, developing and validating SD models is a time consuming procedure which relies heavily on the availability of detailed sets of data, while requiring time and specialized modeling skills (Alzraiee 2013).

### **2.3 Simulation in Construction Management**

Several attempts have been made by researchers to utilize simulation in modeling and planning construction operations. Despite the fact that the vast majority of these applications utilize DES as the modeling tool, there have been some efforts to make use

of SD, Hybrid DES-SD and Agent-Based Modeling and Simulation (ABMS) in modeling construction operations. The main objective of current research is to create robust simulation models, which produce accurate results, while being easy to understand and operate by end users. Researchers constantly try to enhance the accuracy of available models by either incorporating factors which were not included before or by introducing new modeling techniques to their existing work.

### **2.3.1 Discrete-Event Simulation (DES) Applications**

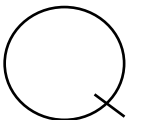
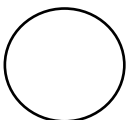
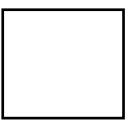
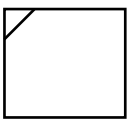
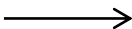
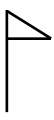
DES was the first simulation technique to be introduced to construction management. Until now, it has by far the majority of applications in this field. This is due to the fact that DES is relatively older than other simulation techniques and has a strong support for operations which are cyclic in nature.

The introduction of simulation in construction management was through CYCLONE, CYCLic Operation Network (Halpin 1977), which is a general purpose simulation language providing users with a systematic way of planning, analyzing and controlling construction operations. CYCLONE is best used for modeling construction operations with a cyclic nature such as highway projects, earthmoving operations, pipe laying, concrete pouring and crane operations. However, due to its generic nature, this language can be used for modeling certain operations outside of the construction industry or for modeling acyclic systems such as projects' schedules. CYCLONE consists of six basic elements, as demonstrated in Table 2-3. 1) *QUEUE*, which resembles a waiting location for resources to be served or used; 2) *FUNCTION*, which is related to a process function, like generating resources from a resource or consolidating a number of



resources into one resource; 3) *NORMAL*, which represents a work task that is unconstrained; 4) *COMBI*, which represents a constrained work task; 5) *ARC*, which indicates the flow logic of resources; 6) *COUNTER*, which controls the mechanism of counting processed resources by the model.

**Table 2-3:** CYCLONE modeling elements (Halpin 1976)

Name	QUEUE	FUNCTION	NORMAL	COMBI	ARC	COUNTER
Symbol						

Various implementations of the CYCLONE modeling language were later developed by researchers including INSIGHT (Paulson 1978), RESQUE (Chang and Carr 1987), UMCYCLONE (Ioannaou 1989), Micro-CYCLONE (Halpin and Riggs 1992) and CIPROS (Tommelein et al. 1994).

Oloufa (1993) developed a simulation tool, MODSIM, which was one the first systems that utilized object-oriented programming in modeling and simulating construction operations. The idea behind MODSIM was to simplify the process of building simulation models, by reducing the required amount of code users must write and using a graphical interface for the application. Objects of diverse classes are created in order to represent the different entities and resources participating in the operation being modeled. These objects have properties, store information and communicate using messages.

Haung and Halpin (1994) developed an application called Dynamic Interface Simulation for Construction Operations (DISCO). This application employs a schematic modeling format to demonstrate the dynamics of the construction operation being modeled. An abstract model diagram of the construction operation is used as a static display. Then, information results are dynamically demonstrated on the screen. At the end of the simulation, the application reports the statistical information in a tabular format.

CIPROS (Tommelein et al. 1994) is an object-oriented system that models construction operations by matching their resource properties to the properties of the design components. Product components and construction resources represent the two types of resources distinguished by the model. Product components refer to the design elements, while construction resources are the equipment, material and labor used during construction.

Sawhney and AbouRizk (1995) developed a hierarchical simulation modeling tool named HSM. It utilizes the principle of Work Breakdown Structure (WBS) and the regular construction process modeling for planning construction projects. The project is broken down into work tasks and the CYCLONE modeling language is employed for the process modeling.

Resource-based Modeling (RBM) (Shi and AbouRizk 1997) is a construction simulation tool that treats processes of active resources as atomic models. These atomic models are stored in the application's library and can be project-specific based on the user's preferences as well as project characteristics. An end user can construct a

simulation model for any construction operation using an atomic model as a base. Then, the required resources are chosen from the resource library and the model is generated by formatting processes into SLAM II network statements (Pritsker 1986).

STROBOSCOPE (Martinez and Ioannou 1999, Martinez 1996) is a very famous general purpose simulation language. It is a programmable simulation system that allows users to model complex construction operations and create special purpose simulation applications. The user can develop models in STROBOSCOPE by writing codes or by using its graphical-based tool, EZStrobe, in which elements similar to those of the CYCLONE language can be dragged and placed in a network form representing the construction operation being modeled. STROBOSCOPE provides users with the ability to access information related to the state of simulation and distinguish involved entities and resources.

McCabe (1997) integrated belief networks and computer simulation in an automated modeling approach. Belief networks are used as a diagnostic tool to assess different performance indices in the project, while computer simulation is used to model the different construction operations. The overall performance of the system is evaluated by belief networks in order to take corrective actions such as modifying the number of servers involved in the model and/or their capacities.

Oluofa et al. (1998) proposed a special purpose simulation system which is a collection of resource-based simulation libraries. The wider the range of defined libraries, the more construction applications it can accommodate. Construction resources are

selected from the libraries and linked together based on the logic of their interaction in the operation.

Simphony (Hajjar and AbouRizk 1999) is a computer application used for developing special purpose simulation tools for construction operations. Special purpose simulation templates can be generated in the Simphony environment by either using the Simphony editor for writing codes or the Simphony graphical designer. Simphony has built-in libraries, which users can utilize to create and run special purpose simulation models. Simphony is the result of accumulating three special purpose simulation tools developed prior to its creation, which are AP2-Earth (Hajjar and AbouRizk 1997) for earthmoving operations, CRUISER (Hajjar and AbouRizk 1998) for aggregate production plants and CSD (Hajjar et al. 1998) for optimizing construction dewatering operations.

Knowledge Discovery Based Simulation System (KEYSTONE) (Elwakil 2011) is a simulation tool that accounts for the subjectivity in modeling construction operations using the Fuzzy logic. KEYSTONE accounts for missing data points and outliers in input data. It uses Fuzzy clustering to model qualitative variables and includes an optimization engine in order to select the optimum combination of resources.

### **2.3.2 System Dynamics Applications**

SD has been applied to construction management for the sake of addressing the issue of dynamic inherent (Alzraiee 2013). In literature, there is a wide range of SD applications that, based on the nature of the problem and the modeler's preference, vary

in different aspects such as the level of details in the model and whether the model is full scale, representing a group of projects or narrow scale, representing one single project.

The previously demonstrated ‘work to do’ cycle shows how an increase in the scope of work in a construction project can affect other variables in its CLD. This work was the basis of many enhanced SD models that proceeded it. Alzraiee (2013) summarized the literature review on the application of SD in construction management. His list is depicted in Table 2-4.

**Table 2-4:** Literate review on SD applications in construction management

<b>Idea</b>	<b>Author/s</b>
Project features such as the development stages of a project	Cooper (1980), Richardson & Pugh (1981)
A quality assurance cycle and a rework cycle	Abel-Hamid (1984)
Nonlinear constraints imposed on work availability and progress	Homer et al. (1993)
Releasing completed work downstream	Ford (1995)
The concurrence constraints limiting the execution of work in parallel	Ford and Sterman (1998)
Managing fund contingency	Ceylan and Ford (2002)
Creating schedule buffer and dynamic planning	Park and Pena-Mora (2003)
Managing iterative errors and change cycles	Lee et al. (2007)

### **2.3.3 Hybrid Simulation Applications**

Alzraiee (2013) states that there exists a need to incorporate DES and SD in hybrid construction simulation models, due to the fact that construction operations are neither completely discrete nor continuous in nature. In addition, operational details are the basis for implementing strategies. Yet, SD has drawbacks in representing the operational level of construction projects in detail. There have been few examples on the application of DES-SD hybrid systems in construction management.

Peña-Mora et al. (2008) used a hybrid DES-SD system in order to analyze the strategic and operational aspects of construction management. The main idea was that the incorporation of the operational details and the strategic viewpoint of construction operations is the backbone of enhancing these operations and increasing performance. The latter is done by allowing managers to identify, assess and respond to improvement areas which traditional modeling techniques would normally miss.

Alzraiee (2013) proposed a framework focused on integrating DES and SD in a single computational platform. The hybrid simulation model was developed by decomposing the construction project into smaller units and developing simulation models based on these units. Three variables, which are the sender, receiver and interface, were defined to achieve the interfacing process among the different simulation models. A synchronization method was established to control the data mapping process between variables and an automated tool, HiSim, was then developed as an implementation of the developed method.

#### **2.3.4 Earthmoving Simulation Practices**

There are several applications on the simulation of earthmoving operations in literature. Earthmoving simulation models mainly aim to account for the uncertainties involved in these operations (Marzouk and Moselhi 2003).

Examples on object-oriented simulation applications for earthmoving operations include MODSIM (Oloufa 1993), which was reviewed in Subsection 2.3.1, and SimEarth (Marzouk and Moselhi 2003). SimEarth provides users with a tool to select a near-optimum fleet configuration that minimizes the total cost and duration of earthmoving operations. The technique utilizes object-oriented modeling and DES to simulate earthmoving operations. It also uses a generic algorithm for optimization purposes. In addition, SimEarth uses a fuzzy-based approach to measure more accurately the haul and return durations of trucks.

Resource-based simulation applications for earthmoving operations include RBM-earth (Shi and AbouRizk 1998), in which different activities involved in the earthmoving operation are linked by the user to formalize the model.

Special purpose simulation applications for earthmoving operations include AP2-Earth (Hajjar and AbouRizk 1997), which is a part of Symphony (Hajjar and AbouRizk 1999). AP2-Earth focuses on providing contractors with an automated planning tool for earthmoving operations directed at delivering accurate estimates on haul and return durations of trucks. In addition, Martinez (1998) developed a special purpose simulation tool for earthmoving operations named EarthMover. Different inputs to the model can be

defined using the graphical interface of the application. Inputs include loading and hauling equipment types as well as road segments' length and characteristics.

## **2.4 Agent-Based Modeling and Simulation (ABMS)**

Agent-Based Modeling and Simulation (ABMS) is the third and most advanced simulation technique being used in scientific research. ABMS has different names in literature including Agent-Based Simulation (ABS), Agent-Based Modeling (ABM), Multi-Agent Simulation (MAS) and Individual-Based Modeling (IBM). Being an evolving computational simulation method, ABMS has been recognized as a suitable instrument for capturing complexity in different systems (North and Macal 2007).

Grimm and Railsback (2013) define an agent-based model as a *“class of computational models for simulating the actions and interactions of autonomous agents, both individual or collective entities such as organizations or groups, with a view to assessing their effects on the system as a whole.”* Agents are self-contained entities that have the ability to control their own actions based on their perception of other agents and their operating environments (Gilbert and Troitzsch 2005). Agents may represent various entities such as people, vehicles, equipment units, projects, ideas, organizations, products, etc.

Unlike DES and SD, which are considered as top-down modeling techniques, ABMS is a bottom-up modeling approach, in which model elements are built before the process is studied as whole. Such an approach allows for the investigation of the fundamentals of model dynamics and leads to realistic conclusions (Unsal 2010). Plus, ABMS has no specific convention on time progression during the model run; it can be



discrete, continuous, or a hybrid of both (Chan et al. 2010). ABMS has the potential to have extensive effects on the way researchers use laboratories in order to support their research and businesses use computers to support decision-making (North and Macal 2007).

#### **2.4.1 ABMS Methodology**

ABMS studies the interaction among objects and their relationship with their surrounding environment (Bonabeau 2002). The main idea behind ABMS is the reproduction of the system being modeled by replicating its different entities and their properties. This is done in an effort to forecast the overall behavior of the system. The goal of an AB model is to peruse explanatory insight into the collective behavior of agents interacting in a certain environment and obeying simple rules (Niazi and Hussain 2011).

The first AB simulation work is believed to be the Dynamic Models of Segregation (Schelling 1971). Schelling modeled individuals in households as agents and gave them characteristics, among which was race. Schelling's model applied cellular automata for the purpose of studying housing segregation patterns. It demonstrated that ghettos can develop spontaneously, even if individuals were colorblind, meaning they have no preference regarding the race of their neighbors. Since then, ABMS has had many applications in different fields, including military, biology, social science, economics and business. The ABMS community has grown very largely and ABMS has become a multidisciplinary subject integrating computer science, cognitive and social sciences as well as simulation (Chan et al. 2010).

Researchers do not agree on a precise definition of agents in AB models. On one hand, some researchers believe that any type of independent entities can be modeled as agents (Bonabeau 2001). However, on the other hand, the majority believe that entities should be adaptive in behavior to be considered as agents. Agents should learn from each other and from their environment and change their behavior in response to their experience (Macal and North 2008). Furthermore, in AB models, agents are given rules to guide their adaptation and responsiveness. Casti (1997) believes that agents in AB models should have base-level rules to guide their behavior and respond to the environment as well as higher-level rules to guide how they can adapt by changing their rules (rules to change the rules).

In relation, Macal and North (2008) argue that the most fundamental characteristic of agents is their independence in making decisions, which requires agents to be active members that respond and interact rather than being purely passive in the model. They list the following common features of agents in AB models, which should not necessarily all exist in every agent:

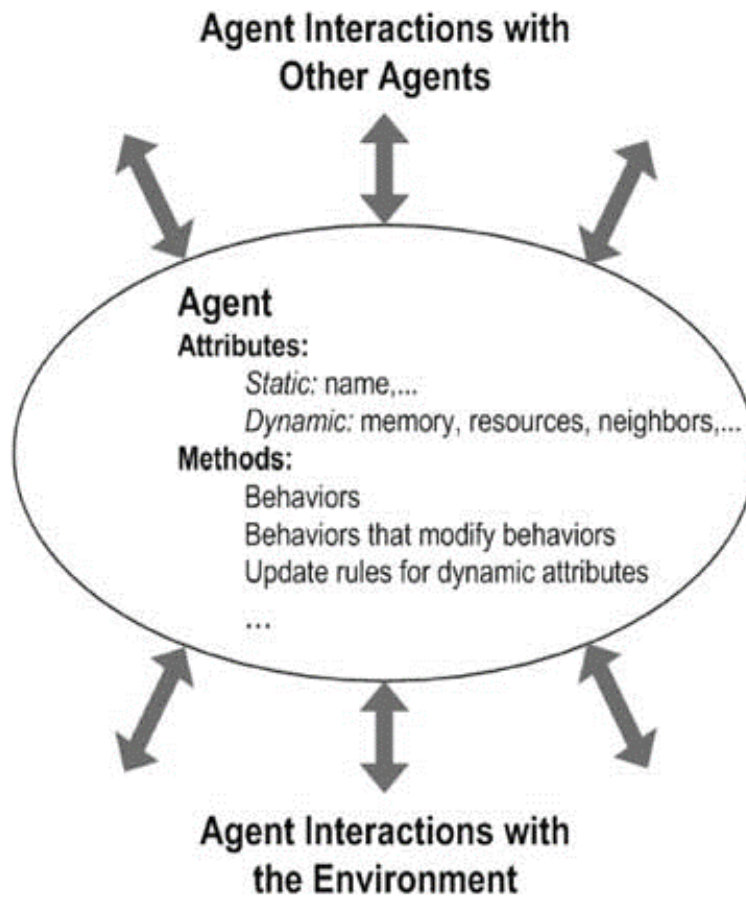
- Agents are identifiable individual components; they have a set of characteristics and rules to guide their behavior and decision-making capacities.
- Agents are autonomous and self-directed; they function independently in their environment and in their interactions with other agents.
- Agents are social; they are able to recognize other agents, distinguish their traits and interact with them based on protocols for interaction and communication.

- Agents are situated; they exist in an external environment with which they interact.
- Agents are goal-directed; they have tasks to achieve and a mechanism to compare the outcome of their behavior to their initial goals.
- Agents are flexible and adaptive; they learn, gain experience, store information and change their behavior accordingly.

Wooldridge and Jennings (1995) gave more detailed specifications on the properties that agents in AB model should possess. They summarized those properties in four specific terminologies as follows:

- Autonomy; agents operate independently, without other agents having direct control over their actions and internal states.
- Social ability; agents interact with other agents in the environment through a specific type of 'language' (a computer language).
- Reactivity; agents are able to perceive and respond to their environment, whether it is physical or virtual.
- Proactivity; agents are able to take the initiative to engage in a goal-directed behavior.

Figure 2-10 demonstrates a typical agent in an AB model. Each agent has different characteristics and methods to govern its behavior and adaption. In addition, agents interact with other agents as well as with their surrounding environment.



**Figure 2-10:** A typical agent in AB models (Macal and North 2010)

The environment in which agents interact can take many forms in AB models. It can be a 2-D or a 3-D space, in the form of a ring or lattice, in the form of a random network, or based on a map such as Geographic Information System (GIS) maps. In AB models, agents have the ability to move freely in their environments, which promotes ABMS as an effective tool for modeling and visualizing complex behaviors in physical systems such as evacuation models, traffic simulations and mechanical systems (Chan et al. 2010).

Furthermore, what makes ABMS powerful is the simulation of the interaction of agents, which creates opportunities to better understand the nature of complex systems.

The strong point of ABMS is that it allows for the simulation of cascading effects arising from minor local interactions, the experimental examination of tipping points, the identification and explanation of beneficial or malicious emergent behaviors and, most importantly, the learning of design mechanisms to grow beneficial behaviors and discard malicious ones (Chan et al. 2010).

The goal of any AB model is to achieve behavior as a result of the interaction of agents. This behavior is often referred to in the AB terminology as emergence or emergent behavior. Macal and North (2010) discuss the issue of emergence in AB models by stating that the bottom-up approach of building AB models agent-by-agent and interaction-by-interaction leads to a state of self-organization. For example, patterns, behaviors and structures emerge from AB models without being explicitly programmed into these models. They state that the focus of ABMS on representing the heterogeneity of agents across their populations and the emergence of self-organization are the main distinguishing features of ABMS when compared to DES and SD.

The principle of emergence can be observed in many existing AB models, even those which are simple and have neither a complex agent architecture nor sophisticated interaction guidelines (Chan et al. 2010). Two simple ABMS examples will be demonstrated in order to give a better idea of agents, environments, interactions and emergence.

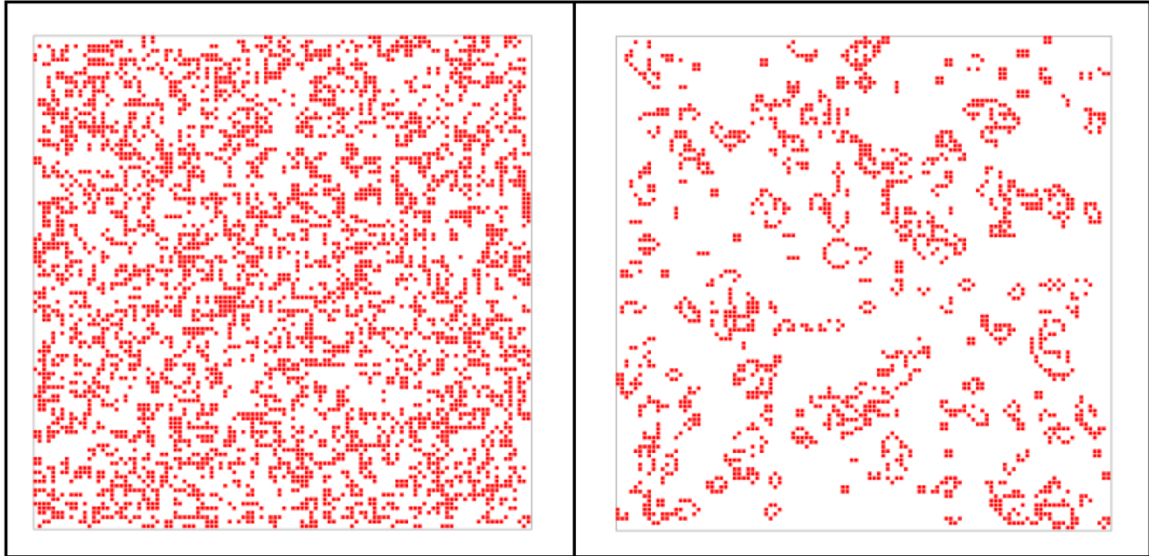
The first example is the application of John Conway's game of life (Berlekamp et al. 2004). The environment is a two dimensional grid that consists of small cells which are considered as agents that align horizontally to make rows and vertically to make

columns. Each cell is surrounded by eight neighbors (right, left, up, down and four diagonals). In addition, cells cannot move in the grid and each cell has two states: alive or dead. Live cells are represented with red dots, while dead cells are represented with white dots. The state of the cells is randomly assigned in the grid and at every time step, there is a possibility for the cell to change its state based on the following three simple rules (Chan et al. 2010):

1. A live cell will remain alive in the next time step, if it has exactly two or three live neighbors.
2. A dead cell will come to life in the next time step, if it has exactly three live neighbors.
3. Other than condition (1) or (2), the cell will die.

The left panel of Figure 2-11 demonstrates the initial state of the system. The simulation is stopped after 100 time steps (100 iterations of executing the AB algorithm) and the state of the system is shown in the right panel of Figure 2-11. The emergent behavior can be clearly noticed in the formation of patterns after the 100 iterations.

What makes the observations of the Game of Life AB model interesting is the fact that the rules are simple and only use local information. The reached patterns are not planned goals which were programmed into the system. Each cell had a set of rules that depended only on its state and the state of its immediately neighboring cells (Chan et al. 2010). In other words, simple rules that rely only on local information can lead to sustainable emergent patterns that are sensitive to those rules and to the agents' initial conditions.

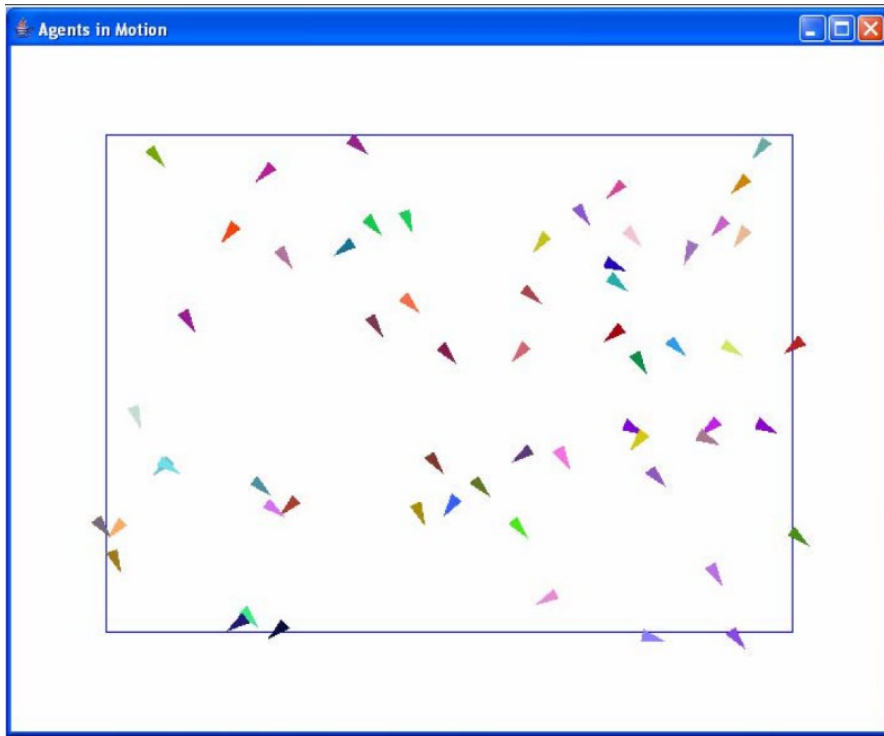


**Figure 2-11:** AB simulation model of Conway's Game of Life (Chan et al. 2010)

Another simple example of AB models is the Boids simulation (Reynolds 1999). The flocking behavior of birds or fish is studied in an AB paradigm. The movement of each agent in the flock is governed by the following three simple rules enumerated below:

1. Separation: agents steer to avoid crowding local 'flockmates'.
2. Alignment: agents steer towards the average heading of local 'flockmates'.
3. Cohesion: agents steer towards the average position of its nearby 'flockmates'.

A fourth rule was added to insure that, during simulation, agents remain close to their initial locations. Figure 2-12 shows the initial configuration of the system, where the number, locations and orientations of agents were randomly assigned. As the simulation started, the behavior of agents started to emerge and a coordinated leaderless flock began to develop. Figure 2-13 demonstrates the state of the system after 500 updates, where clear patterns of organized movement are observed.



**Figure 2-12:** Boids simulation (initial random configuration) (Macal and North 2008)



**Figure 2-13:** Boids simulation (after 500 updates) (Macal and North 2008)



The same conclusions drawn from the Game of Life simulation can be drawn from the Boids simulation as well. Simple rules that depend on local information and agents' initial conditions are enough to create emergent patterns which are not programmed into the model (Macal and North 2008). This is the major strength of the ABMS methodology and the key reason behind this technique replacing current simulations methods such as DES and SD. Hence, lessons learned from applying the Game of Life and the Boids simulations can be extended and applied on a larger scale to represent complex operations in different fields, especially since the ABMS methodology is relatively simple and easy to learn as well as implement.

#### **2.4.2 ABMS Applications in Civil Engineering and Construction Management**

Since ABMS is a relatively new technique, it is expected to have very few applications in fields that do not heavily rely on Artificial Intelligence (AI) such as civil engineering and construction management. The majority of ABMS applications in these fields are focused on the following areas: 1) supply chain management (Tah 2005, Min and Bjornsson 2008); 2) construction claims management (Ren and Anumba 2002, El-Adaway and Kandil 2009); 3) infrastructure management (Sanford Bernhardt and McNeil 2008, Osman 2012).

Supply chain management involves highly complex chains of interacting entities. Sharing information about stocks, costs, quantities and schedules is vital to assure successful supply chain operations. Likewise, construction claims management involves interaction among project participants such as contractors and consultants. It involves discussions, sharing of information and organizing work tasks. Infrastructure

management using ABMS is a promising topic, in which components of an infrastructure system are treated as interacting agents. Governments, infrastructure management agencies, infrastructural assets and users are all modeled as intelligent agents with attributes and goals. This can help anticipate performance, plan maintenance and manage budgets.

Tah (2005) suggested a methodology to develop a modeling platform that can provide inexpensive and risk-free AB environments for organizations to experiment with evolving supply chain management practices prior to execution. Hence, a prototype system was developed using the ZUES tool kit (Ndumu et al. 1999) in an effort to explore the potential of using such an approach to model and simulate collaborative project supply chain networks.

Min and Bjornsson (2008) developed a construction supply chain simulator (CS<sup>2</sup>) that utilized the AB technique in modeling a virtual supply chain for construction projects. The work introduces two types of simulations, which are human-to-human and computer-to-human interactive simulations. In the developed models, groups of people play different roles, as a resemblance to the current practice in construction. The main idea was to test and verify the significance of real-time information sharing in construction.

Ren and Anumba (2002) developed an AB model that utilizes agents' interaction and communication capabilities in AB models to facilitate the claim negotiations among different participants in construction projects. Different project entities are modeled as agents which have different attributes, roles and communication mechanisms. Multi-

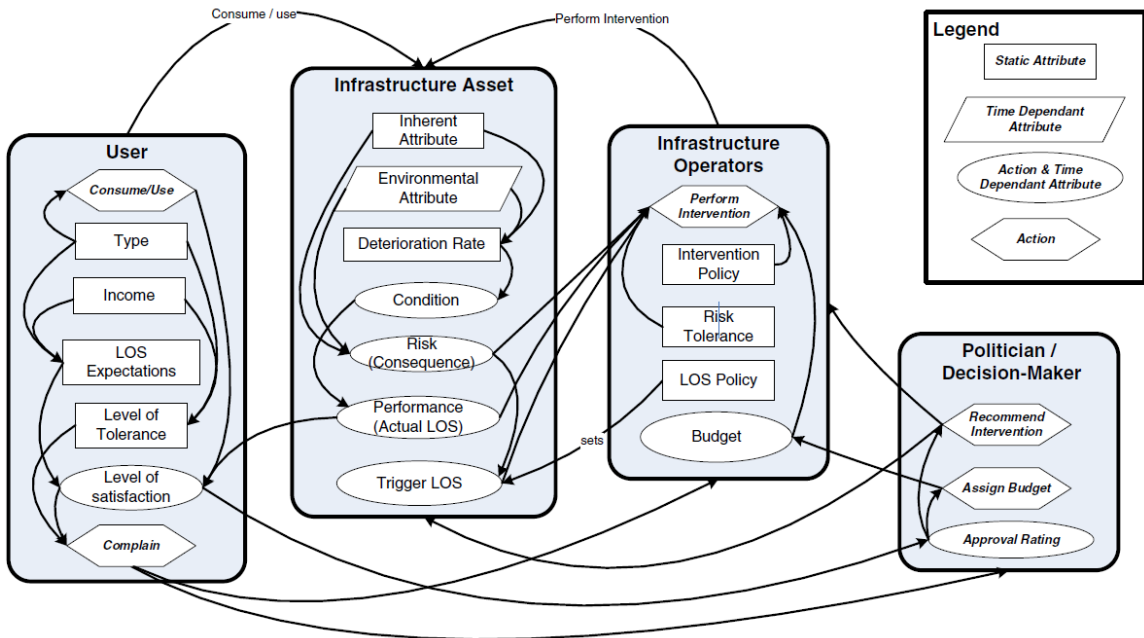
Agent System for Construction Claims Negotiation (MASCOT) was created, with the core objective of enhancing the negotiation power and speeding up the rate of convergence between agents in a construction project.

El-Adaway and Kandil (2009) proposed an AB model for generating legal arguments based on precedent construction disputes. The authors created an AB system for construction dispute resolution (MAS-COR) that automates a previously developed algorithm by the authors. Thirty previously arbitrated construction disputes were used to test and validate MAS-COR. The results of this validation process demonstrated that MAS-COR was capable of originating significant legal arguments for construction claim and dispute professionals, which can help them save time and effort while preparing the defense of their respective positions.

Sanford Bernhardt and McNeil (2008) presented an AB model as a paradigm to improve infrastructure decision-making. The model defines four types of agents: 1) infrastructure segments; 2) users which operate the infrastructure segments; 3) maintenance personnel; 4) politicians/decision makers. Each agent has several predefined characteristics. The model aims at studying the interaction among these agents in order to explore the emergent behavior of the modeled infrastructure system, while aiding in decision-making and budget allocation.

Osman (2012) presented an AB framework to capture the complex interactions occurring in urban infrastructure management. A generic AB model, shown in Figure 2-14, is constructed with four agents: 1) assets; 2) users; 3) operators; 4) politicians. Each of these agents has a set of generic attributes, actions and behaviors. The service quality

domain, which represents customer perceptions and actions that are related to the infrastructure asset level of service, was used as a basis for a detailed behavioral model. A descriptive example of twenty assets and fifty users is simulated to demonstrate emergent behaviors. The simulation showed how changing the social and psychological behavior of users influences their response to consuming municipal infrastructure services. Hence, the results of the simulation highlighted how socio-technical aspects can be incorporated with complex decision-making of urban infrastructure management.



**Figure 2-14:** Agents in an infrastructure AB model

There are a few other ABMS applications on construction management areas including procurement (Dzeng and Lin 2004), construction site safety (Palaniappan et al. 2007) and construction workers' behavior (Ahn et al. 2013).

## **2.5 Summary and Limitations of Literature**

This chapter gave a thorough insight on the different simulation techniques being used in science, and reviewed their applications in construction management in general and earthmoving operations in particular. Firstly, the structure of DES and SD as well as their limitations were presented. Then, different applications of DES, SD and hybrid DES-SD in the construction industry were demonstrated, followed by a discussion on different earthmoving simulation practices. After that, a separate section was dedicated to thoroughly explain the AB methodology and the techniques behind the ABMS realm. Finally, some applications of ABMS in civil engineering and construction management were illustrated.

As mentioned earlier in this chapter, drawbacks of current simulation models of different construction operations in general and earthmoving in particular arise from limitations in both the simulation technique being used as well as the implemented simulation tool itself. DES is considered as a very effective technique for modeling quantitative aspects of operations. However, it is believed that DES cannot capture the external factors that influence the performance of the operation being modeled, which in turn can be driven by hidden causal relationships. In addition, DES is incapable of accounting for the overall strategy of the project or operation being modeled, due to its time step advancement mechanism.

On the other hand, SD is believed to be unable to capture a detailed view of the quantitative operational aspects of modeled systems. SD has a wider and more abstract perspective of the system being studied, compared to the narrowly focused DES. In

addition, SD is a deterministic modeling approach which does not support the representation of operations with stochastic data. Finally, validating SD models is a time consuming process, depends on the availability of detailed data sets and requires specialized modeling skills.

In regards to earthmoving simulation practices, most developed models lack good graphical modeling support, behave in an inflexible and predetermined manner, require visualization when implementing and are rigid in accepting various sets of data. Despite the fact that some researchers were able to overcome limitations of earthmoving simulation practices using DES, SD or a hybrid of both, there is a strong need to introduce the relatively new and promising simulation technique, which is ABMS, to the study and planning of earthmoving operations in one comprehensive simulation system.

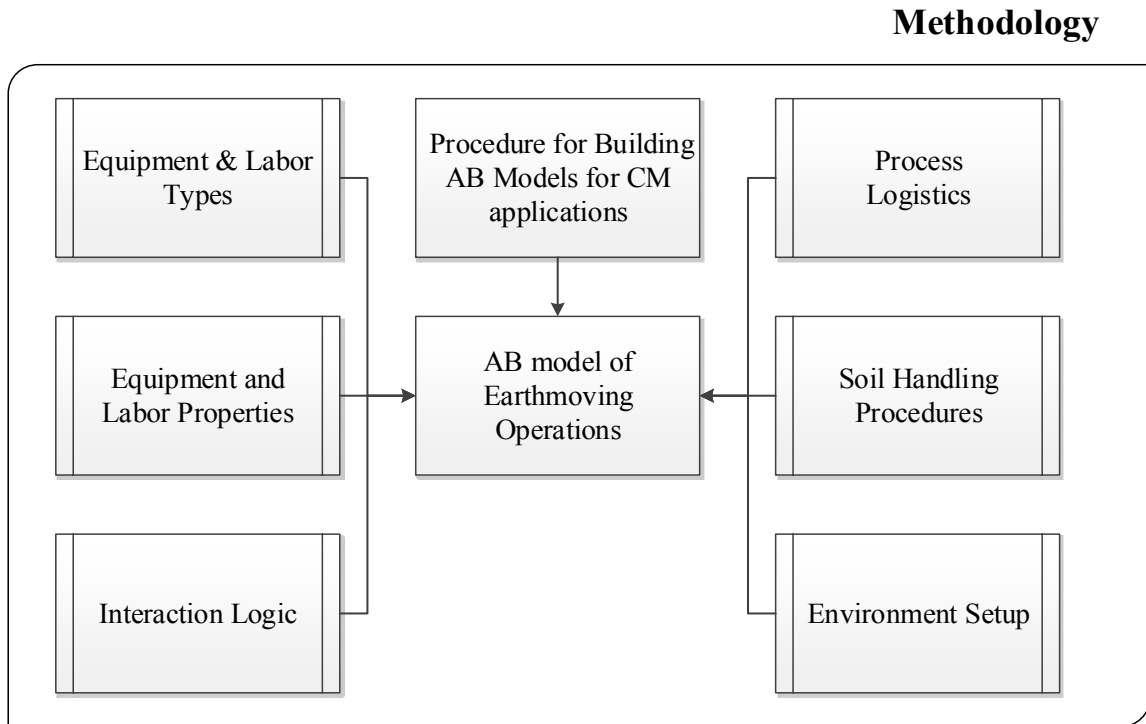
## **CHAPTER 3: METHODOLOGY**

### **3.1 Chapter Overview**

This chapter is dedicated to providing a detailed explanation of the research methodology. The main idea of the proposed framework is the development of a conceptual AB model for earthmoving operations which will later be implemented in an automated simulation tool. In order to achieve this goal, a clear procedure on how to create AB models for construction operations should be first outlined. Figure 3-1 demonstrates the breakdown of the proposed methodology.

In Section 3.2, a general procedure for creating AB models for different construction management applications is presented. This procedure was developed based on the author's knowledge about the AB methodology, which in turn was obtained from reviewing the literature on applications of ABMS in different disciplines. The presented procedure is intended to be generic and simple. Examples will be given in this section to demonstrate how this procedure can be implemented to develop AB models for real-world applications in construction management.

Section 3.3 is devoted to demonstrating the development of a comprehensive AB model for earthmoving operations. Each component of the model is outlined, including agents and their environment. Agents' types, attributes and roles are defined in detail. In addition, the interaction logic and the guidelines for the implementation of the earthmoving AB model are discussed in this section.

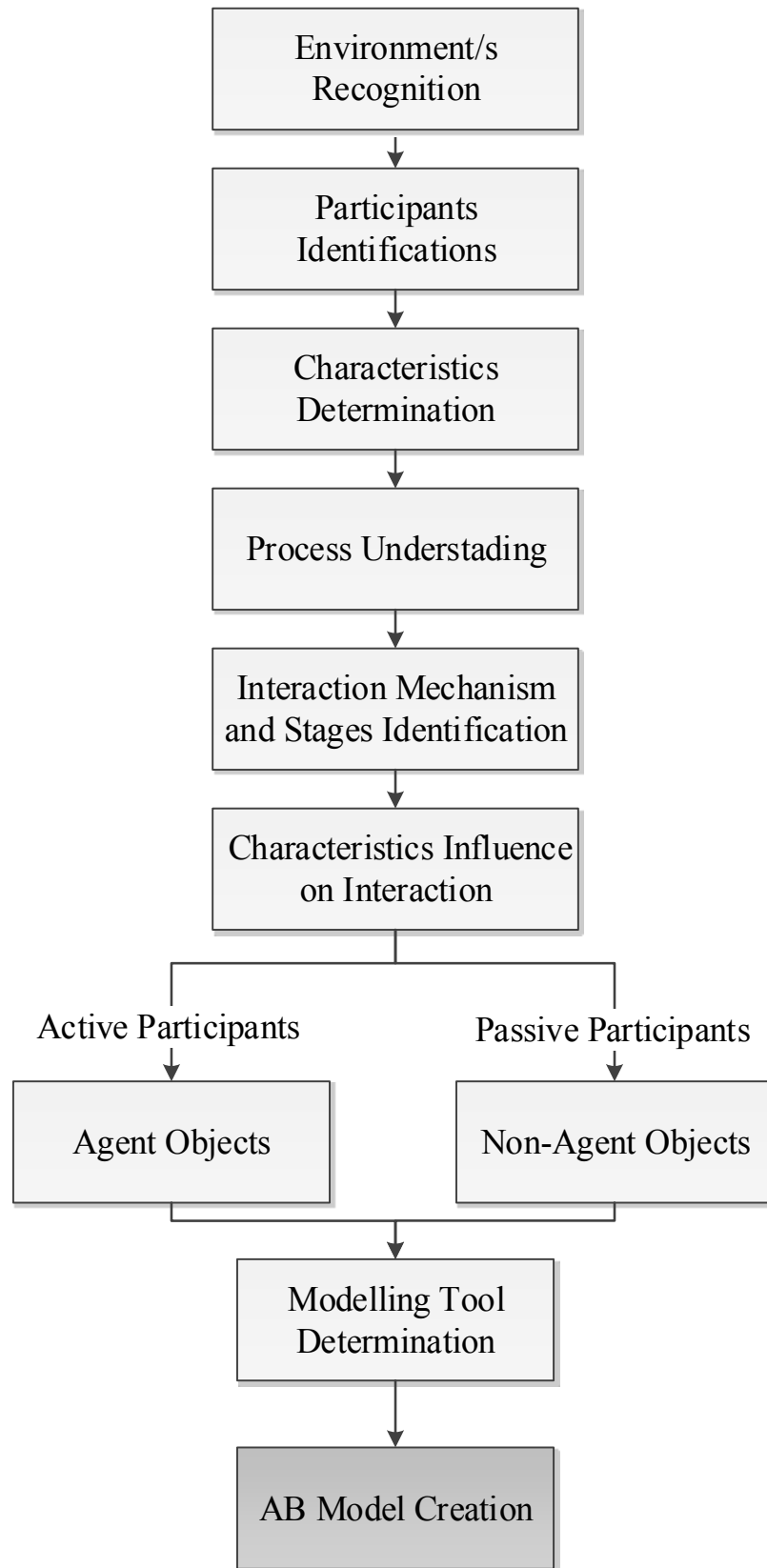


**Figure 3-1:** Methodology breakdown

### **3.2 A Procedure for Creating AB Models for CM Applications**

ABMS is a relatively new technique that is gaining an increasing interest in different fields. As mentioned in Subsection 2.4.2, the applications of ABMS in civil engineering and construction management are very few in number. Consequently, researchers who are interested in developing AB models for systems in these areas have to go through an extensive literature review on the AB methodology and applications in order to familiarise themselves with the method and understand the logic behind it. This is why it was decided to summarize the techniques of building AB models in the simple step-by-step procedure shown in Figure 3-2.





**Figure 3-2:** Procedure for developing AB Models in construction management

The procedure presented by the flow chart in Figure 3-2 emphasizes that ABMS is a bottom-up approach. It shows how several steps have to be carried out to insure a successful transition of ideas into a working and realistic model. Prior to the implementation of any AB model, the following nine-step procedure should be carried out:

- 1) Recognise the environment/s in which the operation takes place.
- 2) Identify all participants in the real-world operation.
- 3) Identify all possible characteristics and properties, both quantitative and qualitative, of each participant.
- 4) Understand in detail the process which is going to be simulated.
- 5) Determine the stages of the operation at which participants would directly or indirectly interact.
- 6) Identify how the different characteristics of each participant would affect the interaction.
- 7) Consider any participant, whose characteristics and states change during the operation and which directly interacts with other participants, as an agent.
- 8) Consider other participants in the operation as passive objects that are imbedded in the environment (considering such participants as agents is not wrong; however, it overcrowds the model and increases the required memory in computer programs).
- 9) Determine the best tool for modeling the system after viewing and studying the properties and capabilities of the available modeling tools.

After all these tasks have been fulfilled, the model can then be created. In ABMS, each agent population is formed separately. After finalizing the agents' properties, the modeler can then move on to reflect on the process as a whole. Agents are firstly linked to the environment, and then the interaction among them at certain stages of the operation is established. During model construction, modifications to the structure of the model can be made if needed. The following subsections demonstrate how this procedure is applied to create an AB model for construction operations

### **3.2.1 Environment Recognition**

The first step in creating AB models is defining the environment in which the system or the operation being modeled exists. An environment in AB terms refers to a platform in which agents live and interact. Having a realistic view of the environment helps the modeler understand and anticipate effects that this environment would have on the model in general and agents in particular. It is important to note that in AB models, more than one environment can exist. An agent, for example, can interact with an agent in a certain environment and with another agent in a different environment.

When modeling most construction operations, the environment would be the construction site. However, in cases where the operation being modeled is specific to one area of the construction site, the modeler could choose that area to be the environment of the AB model. Environments in AB models are not always physical. For example, if an AB model is created to represent different parties working in a construction project (contractors, consultants, construction managers and owners) for conflict resolution purposes, then it would make no sense for the modeler to use the construction site as the

environment of the AB model. Hence, the choice of the environment in that case would heavily depend on the agents' interaction mechanism. A reasonable choice would be a computer database containing the entire project's contracts, schedules, change orders, BOQ's, etc.

When developing an AB model for earthmoving operations, the environment should be the physical area in which the operation takes place. The latter includes the excavation area, loading area, hauling road, dumping area, returning road and any other physical space which the earthmoving operation occupies or affects through any of its equipment or labor at any of its stages.

### **3.2.2 Participant Identification**

Prior to deciding on agent types and characteristics in AB models, it is crucial for the modeler to identify all participants in the system being modeled. Even if the modeler believes that a certain object or entity does not significantly contribute to the operation, it is better at this stage to consider it as a candidate for becoming an agent. All individuals, equipment and physical objects that are parts of the system should be put in a list at this point. To reiterate, any entity in the system that has characteristics and attributes which are likely to influence the operation of that system in any direct or indirect way, is considered as a participant.

Participants in a simple earthmoving operation can be soil, dozers, loaders, haulers (trucks), dump spotters, transportation roads and weather conditions. Later in Subsection 3.2.7, it will be decided which of these participants will be an agent and which will not.

### 3.2.3 Participants' Characteristics Determination

The emergent behaviour of AB models occurs mainly due to the attributes of agents and their interactions. This is why the more accurate the representation of agents' attributes and interactions in an AB model is, the more realistic the emergent behaviour becomes.

Table 3-1 iterates some major properties of the aforementioned participants of an AB model for earthmoving operations. It is crucial to note that selecting these participants and their properties was based on the author's experience and perception of the operation. Different modelers can come up with lists containing different properties than those shown in Table 3-1.

**Table 3-1:** Earthmoving operations' participants and their properties

<b>Participant</b>	<b>Properties</b>
Soil	Type, Quantity, Moisture Content, Density
Dozer/s	Weight, Condition, Blade Dimensions, Speed
Loader/s	Condition, Excavation Rate, Bucket Capacity, Loading Rate, Speed
Truck/s	Weight, Condition, Load Capacity, Dumping Rate, Hauling Speed, Empty Speed
Spotter/s	Experience, Age, Walking Speed
Weather Conditions	Temperature, Precipitation, Humidity, Wind Speed
Transportation Road	Terrain, Length, Inclination, Traffic, Stops

### **3.2.4 Process Understanding**

After recognizing the environment of the system, identifying its participants and determining their characteristics, it is time to take a look at the operation of the system as a whole and understand its mechanism in total depth. Contrary to other simulation techniques, the process understanding in creating AB models is the fourth step and not the first. The latter is due to the bottom-up modeling style of ABMS, which builds the model from its agents up to a full representation of the process as a whole. In addition, if modelers thoroughly investigate systems before identifying participants and their properties, it would affect their judgement on these participants and properties. When creating AB models, modelers might disregard certain agents or attributes of agents thinking that they do not contribute to the operation or that their contribution is negligible. This is a risky issue in ABMS, as very simple attributes of agents can lead to complex emergent behaviours.

In an earthmoving AB model, the process is fairly simple. The operation of an earthmoving system with the aforementioned participants would begin by dozers excavating a specific area. Then, loaders would carry the excavated earth and lay it in trucks. Then, trucks would haul the carried earth to a certain location, dump it and return to the initial site in order to carry and haul more earth. This goes on until the intended quantity of earth is excavated and dumped.

### **3.2.5 Interaction Mechanism and Stages Identification**

The interaction of agents is the backbone of ABMS. The interaction mechanism, location and time are all very important in carving the emergent behaviour of the system

being modeled. Although the interaction mechanism and stages can be inferred by studying the process being modeled in general (process understanding step), an in-depth focus on the interaction procedure is vital, as it will be the core of the implementation stage later on in the process.

An example of the interaction of participants in an earthmoving AB model is a loader loading a truck numerous times until that truck is full and ready to go. How the loader loads the truck, the distance between the two equipment units while working together and the communication between the operators of the two equipment units are all examples of the mechanism of interaction between the loader and truck agents in the AB model. An elaborate discussion on this issue will be presented in Section 3.3.

### **3.2.6 Characteristics Influence on Interaction**

The interaction between agents in AB models heavily relies on the characteristics and states of these agents. This principle is the source of agent intelligence in ABMS. Since agents are proactive and adaptive, their interactions with other agents will take into consideration their attributes and states. This should be studied by modelers prior to the implementation of AB models, as it will help them design agents' attributes, roles and communication methods in a more realistic manner.

Going back to the truck-loading activity example, when the loader arrives at the truck location its operator observes two things: 1) the capacity of the truck, which is an attribute; 2) the level of earth in the truck, which is a variable (state). Truck capacity is fixed, but the quantity of carried earth increases after each load. Truck capacity and carried earth (or available space) determines the duration of interaction between the

loader and the truck, as the loader will stop loading the truck when it is full. More on that will be discussed in Section 3.3.

### **3.2.7 Agent Objects and Non-Agents Objects**

After recognizing the model's environment, defining its participants and their attributes, understanding the process, identifying the interaction stages and figuring out the influence of model participants on the interactions, the model is almost ready for implementation. The last major step prior to model implementation is to select which model participants qualify for being agents and which do not. In AB models, the logic is not affected if all participants are considered to be agents. However, it is a common practice for agents in AB models to be of those participants which are active, have multiple attributes and engage in diverse interactions. Other participants that have limited properties, fixed goals and minimal interaction with other model components are considered as passive non-agent objects. The reason behind this classification is to avoid overcrowding the model, saving memory and reducing run time in implemented computer programs.

For the earthmoving operation participants listed in Table 3-1, a modeler can choose the equipment units (dozers, loaders and trucks) and spotters to be agents. Soil, weather conditions and transportation roads can be embedded in the model's environment. However, this choice relies heavily on the purpose of the model in question and its required level of complexity.



### **3.2.8 Modeling Tool Determination**

Implementing AB models is heavily dependent on computer programming. The nature of ABMS makes object-oriented programming the most suitable implementation tool for this technique. Object-oriented programs are developed through a bottom-up approach in which independent objects are created and given attributes and roles. The majority of AB models in literature were developed using object-oriented programming.

Modelers can choose to use common programming languages including C#, C++, Java, etc. to create an AB model and simulate with it. However, there are some ready AB modeling applications that were created in order to facilitate modeling for users, especially those who do not have a strong programming background. These applications are created using object-oriented programming and have built-in AB libraries, so that users can create AB models in a fast manner without the need of writing many lines of code. Also, flexibility in representing agents, accommodating variables, defining roles, executing actions, producing results and performing analysis differs from one tool to another. The nature of the system being modeled, the required complexity of the model and the targeted type of results and analysis are the major criteria in choosing the appropriate AB modeling tool.

### **3.3 A Comprehensive AB Model for Earthmoving Operations**

This section is dedicated for applying the methodology presented in Section 3.2 in developing a comprehensive AB model for earthmoving operations. The term ‘earthmoving’ refers to the process of excavating and transporting quantities of soil from one location to another, mainly prior to the construction of facilities. Thus, earthmoving

operations utilize different equipment based on the construction project's scope of work and available resources. The most common earthmoving operation represents the excavation of earth from a certain location and the transportation of that earth to another location where it will be dumped. However, the mechanism of the excavation, transportation and dumping activities differ from one project to another, leading to different combinations of equipment types and interactions.

### **3.3.1 The Scope of the Model**

The AB model developed in this research is of an earthmoving operation that includes bulldozers, loaders, haulers (trucks) and dump spotters. Accordingly, the operation would be as follows: 1) bulldozers excavate earth in a certain area and push it in stockpiles; 2) loaders fill trucks with earth from the stockpiles; 3) trucks transport the carried earth to a certain dumping location; 4) with the help of dump spotters, trucks dump their loads in stock piles; 5) trucks return to the loading area to carry and transport more earth.

The model should also accommodate earthmoving operations which do not include bulldozers. This would be the case if the operation was simply the transportation of earth from a stock pile into a dumping area, or if the loaders, or any type of excavators, are performing both the excavation and truck loading activities.

The aforementioned equipment units and dump spotters will all be considered as agents. The structure of the model indicating the types of participants and whether they are agents or not is illustrated by Table 3-2. As can be noted from the table, weather conditions were neglected and soil as well as transportation roads were considered as

passive objects that are embedded in the model's environment. The soil and the transportation road components are represented by the model, but in a passive manner in which agents are in control of them.

It is important to stress the fact that an AB model's accuracy in representing a real-world operation is heavily dependent on the efforts put forth while preparing the model. Highly complex AB models, in which all agents' attributes are captured in detail, the interaction logic replicates the one in real-life operations, all surrounding factors are taken into consideration to the best possible ability, are expected to deliver outstandingly accurate and realistic behaviors and results. The developed AB model captures the operational aspects of earthmoving very accurately, as will be demonstrated in Subsection 3.33.3.2. However, there is still much room for improvement in accounting for factors surrounding the operation. This will be summarized when discussing future work recommendations.

**Table 3-2:** Proposed AB model participants

<b>Participant</b>	<b>Agent or Non-Agent</b>
Soil	Environment-Embedded Object
Dozer/s	Agent Object
Loader/s	Agent Object
Truck/s	Agent Object
Spotter/s	Agent Object
Weather Conditions	Neglected Participant
Transportation Road	Environment-Embedded Object

Two types of properties are defined for each agent in the proposed AB model: 1) attributes, which refer to fixed properties of agents that do not change at any time and are unaffected by interactions with the environment or other agents; 2) variables, which refer to properties of agents that can change due to interactions with the environment or other agents. Table 3-3 lists agents along with their attributes and variables in the proposed AB model. These attributes and variables are explained in detail in Section 3.3, by illustrating their definition and demonstrating how they influence agent interactions.

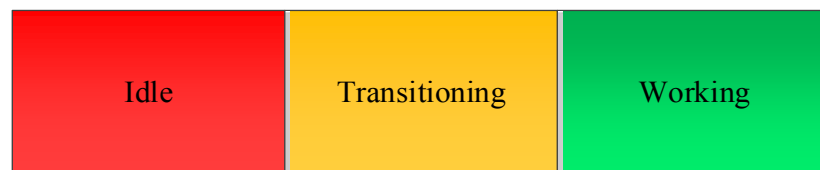
**Table 3-3:** Agents’ attributes and variables in the proposed AB model

<b>Agent</b>	<b>Attributes</b>	<b>Variables</b>
Bulldozer/s	<ul style="list-style-type: none"> <li>• Push Quantity (PQ)</li> <li>• Push Duration (PD)</li> <li>• Time to Adjust Position (TAP)</li> <li>• Return Duration (RD)</li> </ul>	<ul style="list-style-type: none"> <li>• Actual Push Quantity (APQ)</li> <li>• Actual Push Duration (APD)</li> </ul>
Loader/s	<ul style="list-style-type: none"> <li>• Bucket Capacity (BC)</li> <li>• Time to Load Full Bucket (TLFB)</li> <li>• Time to Adjust Position While Full (TAPF)</li> <li>• Time to Unload Full Bucket (TUFB)</li> <li>• Time to Adjust Position While Empty (TAPE)</li> </ul>	<ul style="list-style-type: none"> <li>• Carried Earth (CE)</li> <li>• Time to Load Bucket (TLB)</li> <li>• Unloading Quantity (UQ)</li> <li>• Time to Unload Bucket (TUB)</li> </ul>
Hauler/s	<ul style="list-style-type: none"> <li>• Capacity (C)</li> <li>• Time to Get in Load Position (TGLP)</li> <li>• Hauling Duration (HD)</li> <li>• Time to Get in Dump Position (TGDP)</li> <li>• Dumping Duration (DD)</li> <li>• Returning Duration (RD)</li> </ul>	<ul style="list-style-type: none"> <li>• Carried Earth (HCE)</li> <li>• Available Space (AS)</li> </ul>
Spotter/s	<ul style="list-style-type: none"> <li>• Time to Adjust Position (STAP)</li> </ul>	None

### 3.3.2 The Creation of Agents

In AB models, diagrams referred to as state charts are often used to describe the different stages (states) that agents pass through while performing their roles. In turn, state charts are basically flow charts that consist of blocks connected by arrows. Each block forms a state and each arrow represents a transition from a state to another. Thus, state charts are used to construct agents and govern their roles as well as their communication mechanism. The overall behavior of the AB model is generated from the interaction of its agents' state charts. After creating state charts, they are implemented in computer programs for simulation purposes to obtain the emergent behavior of the system.

For the proposed earthmoving model, four state charts are created; one for each agent. Figure 3-3 demonstrates a color legend for the agents' state charts, which is going to be used throughout the model construction and implementation. The red color refers to an idle state of the agent. It signifies that the agent is waiting for a certain condition to be fulfilled or a certain type of interaction to be carried out in order for it to move on to the next state. Furthermore, a green color represents a working state of the agent. It indicates that the agent is currently performing its main role. Finally, a yellow color is used to describe a transitional state of the agent. It indicates that the agent is working, but on a minor task that is mostly a complement of the main task.



**Figure 3-3:** Color legend for agents' state charts

Each agent will be explained separately. A table containing the agent's attributes and variables will be presented and subsequently, its components will be illustrated. Then, the agent's state chart will be demonstrated along with a description on how the agent progresses through the different states of that state chart. Finally, a table that includes a description on the state chart's transitions will be shown. The condition/s and action/s of each transition are listed in that table.

Prior to getting into details, it is important to define three variables which are crucial to understanding the agents' breakdown:

- 1) *Soil Ready for Excavation (SRE)*: refers to the quantity of soil available to be excavated by bulldozers.
- 2) *Soil Ready for Loading (SRL)*: refers to the quantity of excavated soil that is ready to be loaded in trucks (or to the quantity of soil available to be excavated by loaders or other excavators, if the operation does not include bulldozers).
- 3) *Dumped Soil (DS)*: refers to the quantity of soil that has already been transported and dumped at the dumping location.

### **The Bulldozer Agent:**

The attributes and variables of the bulldozer agent are shown in Table 3-4. *Push Quantity* refers to the quantity of excavated soil by the bulldozer at the end of each pass. *Push Duration* is the time it takes for the bulldozer to excavate that quantity of soil (to make one pass). *Time to Adjust Position* refers to the time it takes for the bulldozer to turn and rotate to the opposite orientation. *Return Duration* is the time it takes for the bulldozer to return back to the location where it can begin another excavation pass.

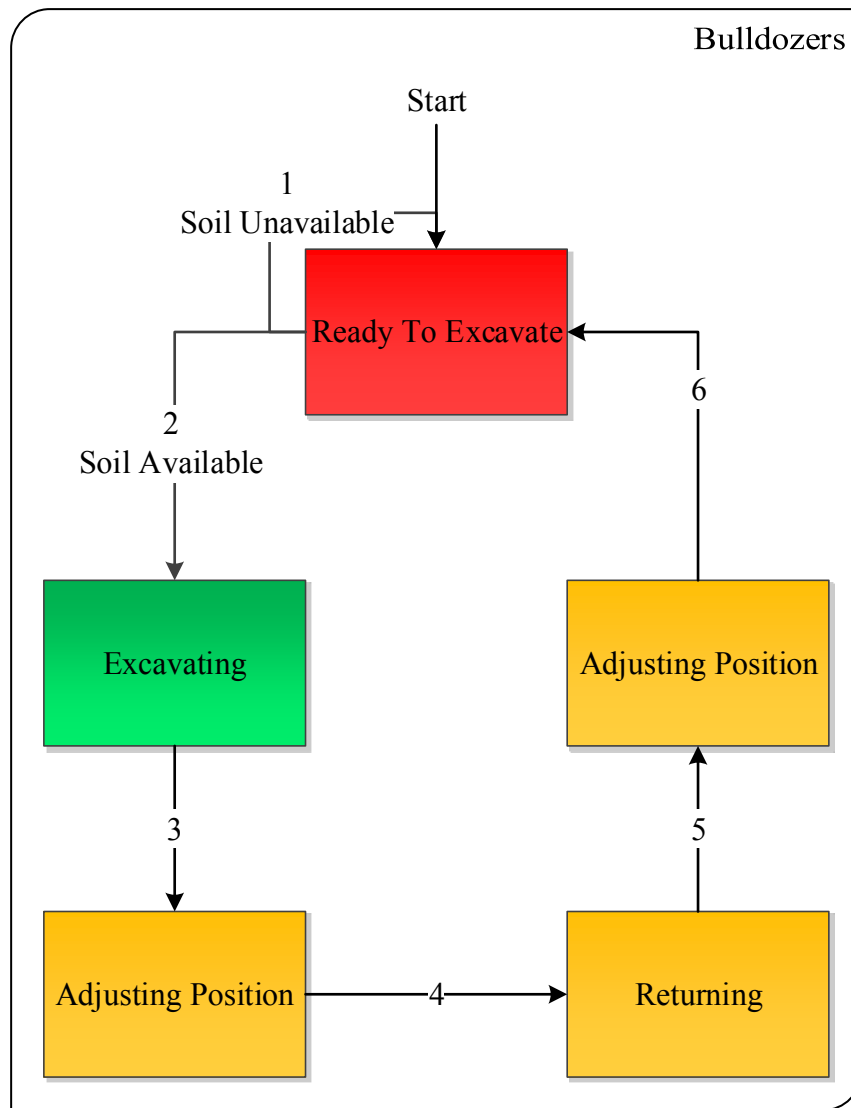
In addition, *Actual Push Quantity* and *Actual Push Duration* are designed for special cases when the quantity of soil available for excavation is less than the push quantity of the bulldozer. In such cases, the *Actual Push Quantity* would be whatever quantity is available and the *Actual Push Duration* would be an interpolated fraction of the regular *Push Duration* based on the ratio of the *Actual Push Quantity* to the regular *Push Quantity*. Regularly, *Actual Push Quantity* and *Actual Push Duration* are equal to *Push Quantity* and *Push Duration*.

**Table 3-4:** Bulldozers' attributes and variables

Agent	Attributes	Variables
Bulldozer/s	<ul style="list-style-type: none"> <li>• Push Quantity (PQ)</li> <li>• Push Duration (PD)</li> <li>• Time to Adjust Position (TAP)</li> <li>• Return Duration (RD)</li> </ul>	<ul style="list-style-type: none"> <li>• Actual Push Quantity (APQ)</li> <li>• Actual Push Duration (APD)</li> </ul>

The state chart of the bulldozer agent is depicted in Figure 3-4. The state chart commences by the bulldozer being at the starting point, ready to begin excavating. If there is no *Soil Ready for Excavation*, the bulldozer remains idle. However, if there is *Soil Ready for Excavation*, the bulldozer starts moving and excavating that soil, either to the regular *Push Quantity* or to whatever quantity is available. After the bulldozer reaches the end of its excavation pass, it adjusts its position and rotates to be able to move in the opposite direction. Subsequently, the bulldozer starts returning and heading for the starting point of the excavation pass, where it adjusts its position and rotates again to be in the correct form, ready to perform another excavation run. In view of that, the bulldozer agent in the proposed AB model has no interactions with other agents.

However, considering the bulldozer participant as an agent is necessary since it is an equipment unit that is actively participating in the system. In addition, it is better to keep it as a flexible entity in case its mechanism changes when upgrading the model in the future.



**Figure 3-4:** Bulldozer’s state chart

Table 3-5 provides a detailed elaboration on the bulldozer’s state chart transitions. The actions of transition 2 were explained earlier when studying the bulldozer agent’s



state chart. The actions of transition 3 include reducing the quantity of *Soil Ready for Excavation* by the *Actual Push Quantity*. Aside from this state, all other states and transitions are straightforward.

**Table 3-5:** Bulldozer’s state chart transitions

Transition	Condition/s	Action/s
1	$SRE = 0$	No action
2A	$SRE > 0$ & $SRE > PQ$	$APQ = PQ$ $APD = PD$ Start Excavating
2B	$SRE > 0$ & $SRE \leq PQ$	$APQ = SRE$ $APD = PD \times \frac{APQ}{PQ}$ Start Excavating
3	Excavation pass completed	$SRE$ reduced by $APQ$ $SRL$ increased by $APQ$ Start adjusting position
4	Position adjustment completed	Start returning
5	Return completed	Start adjusting position
6	Position adjustment completed	No action

**The Loader Agent:**

Table 3-6 illustrates the attributes and variables of the loader agent. In turn, *Bucket Capacity* refers to the quantity of soil that the loader can carry in its bucket. *Time to Load Full Bucket* is the time it takes for the loader to fill its bucket with that quantity

of soil (*Bucket Capacity*). *Time to Adjust Position While Full* refers to the time it takes for the loader, when its bucket is full, to get in an appropriate position for loading trucks (haulers). *Time to Unload Full Bucket* is the time it takes for the loader to dump its full bucket load (*Bucket Capacity*) of soil in the truck. *Time to Adjust Position While Empty* refers to the time it takes for the loader, when its bucket is empty, to get in a position for loading its bucket after dumping its load in the truck.

Moreover, *Carried Earth* represents the actual quantity of soil in the loader's bucket. This variable is added to accommodate different bucket quantity scenarios, determined by the availability of soil. *Time to Load Bucket* refers to the actual time it takes to load the bucket based on *Carried Earth*. Thus, it is an interpolated fraction of the regular *Time to Load Full Bucket* based on the ratio of the *Carried Earth* to the *Bucket Capacity*. *Unloading Quantity* represents the quantity of soil that the loader chooses to dump in the truck it is serving, which depends on a variable of the hauler agent called *Available Space*. To elaborate, if the *Available Space* of the truck is less than the loader's *Carried Earth*, the loader dumps only a quantity of soil equal to the truck's *Available Space*. Accordingly, the *Time to Unload Bucket* is an interpolated fraction of the regular *Time to Unload Full Bucket* based on the ratio of the *Unloading Quantity* to the *Bucket Capacity*. This can be further understood by examining the loader's state chart.

Since the model is designed in a way that allows for the loader agent to perform the excavation activity when no bulldozers are included, all attributes and variables in Table 3-6 are made to be generic and can accommodate both scenarios. *Time to Load Bucket* can refer to either the time it takes for the loader to fill its bucket from an excavated soil in stock piles or the time it takes for the loader, or other types of

excavators, to perform the excavation activity and fill their buckets with soil. If needed, other attributes and variables change accordingly as well. This is an example of flexibility in AB models.

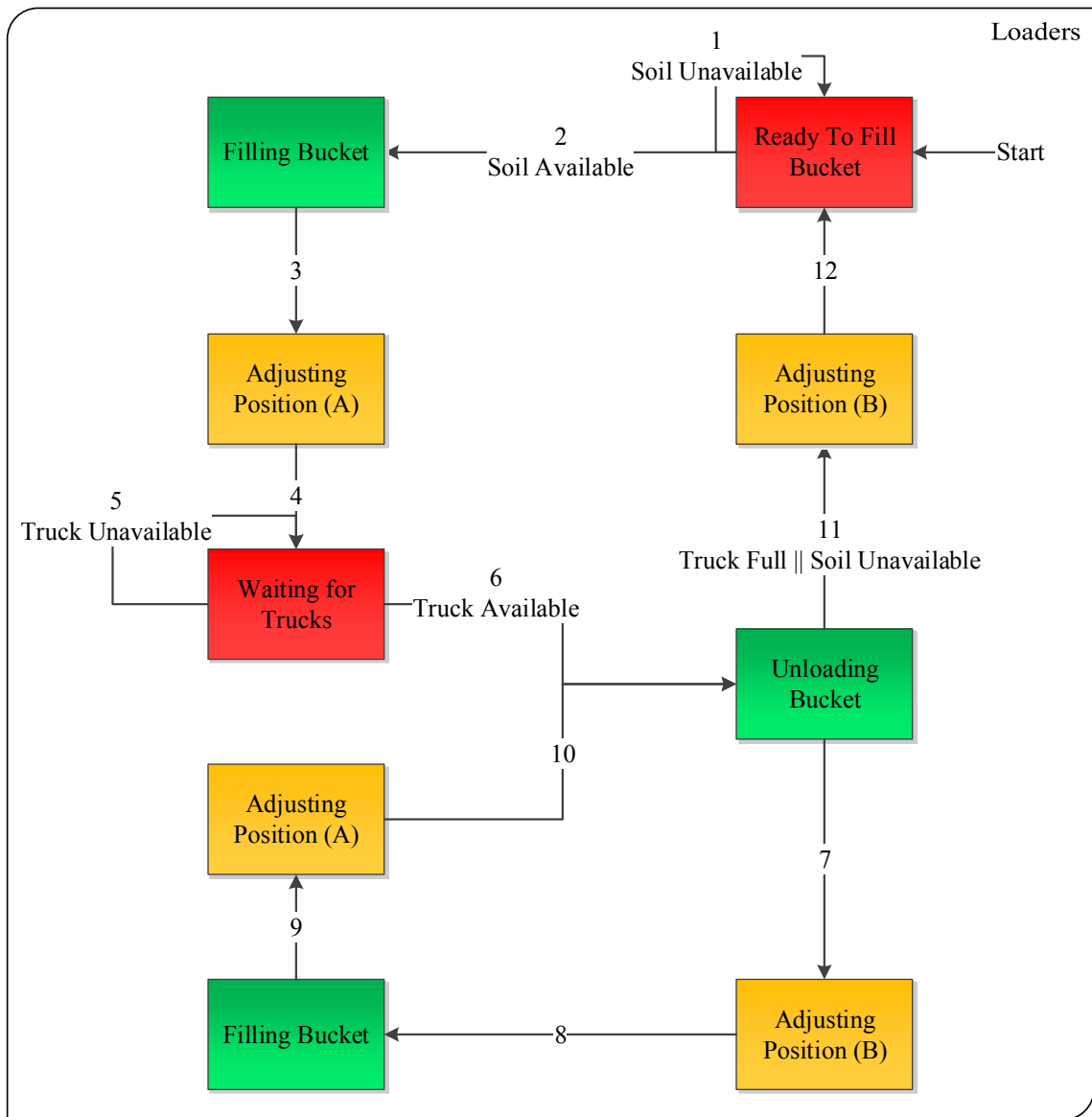
**Table 3-6:** Loader’s attributes and variables

Agent	Attributes	Variables
Loader/s	<ul style="list-style-type: none"> <li>• Bucket Capacity (BC)</li> <li>• Time to Load Full Bucket (TLFB)</li> <li>• Time to Adjust Position While Full (TAPF)</li> <li>• Time to Unload Full Bucket (TUFB)</li> <li>• Time to Adjust Position While Empty (TAPE)</li> </ul>	<ul style="list-style-type: none"> <li>• Carried Earth (CE)</li> <li>• Time to Load Bucket (TLB)</li> <li>• Unloading Quantity (UC)</li> <li>• Time to Unload Bucket (TUB)</li> </ul>

Figure 3-5 represents the state chart of the loader agent. Accordingly, the initial state is the loader being ready to begin loading its bucket by carrying soil from an excavated stockpile (or performing the excavation). If there is no *Soil Ready for Loading*, the loader remains idle. If, however, there is *Soil Ready for Loading*, the loader starts filling its bucket either to its maximum capacity (*Bucket Capacity*) or to the available quantity of soil if it is less than that capacity. After the loader’s bucket is filled, the loader adjusts its position to be ready for loading the arriving truck. At that point, the loader checks the truck’s queue. If there are no trucks waiting to be loaded, the loader remains idle. However, if trucks are available, the loader signals the first truck in the queue to move to the loading area. Then, the loader determines the *Available Space* in the truck. If that *Available Space* is larger than or equal to the loader’s *Carried Earth*, the loader dumps its entire bucket load in the truck. Otherwise, the loader dumps a quantity equal to

the truck's *Available Space*. Regularly, when trucks arrive for loading, their *Available Space* will be equal to their *Capacity*. After unloading the bucket's load in the truck, the loader checks the *Available Space* in the truck again and determines whether the truck as reached its *Capacity* or not. In the event that the truck is full or if there is no *Soil Ready for Loading* at that point, the loader signals the truck that the interaction between the two equipment units is over and that the truck should start hauling to the dumping location. The loader then has to adjust its position to get in the bucket loading setup again and waits for extra quantities of soil to become available. On the other hand, if the loader determines that the truck has not yet reached its full *Capacity* and that there is still *Soil Available for Loading*, the loader adjusts its position, fills its bucket and loads the truck again. This cycle is repeated until the truck reaches its *Capacity* or the *Soil Ready for Loading* is fully consumed.

In light of that, the loader in the proposed AB model is a good example of a dynamic agent that adapts actively with the changes of model conditions and other agents' properties. During different cycles of the loader's operation, the same variable can have different values that depend on the cycle's conditions and interactions. This demonstrates the strength of the ABMS technique and its promising capabilities for future enhancements.



**Figure 3-5:** Loader’s state chart

Table 3-7 presents a detailed breakdown of the loader’s state chart transitions.

Consequently, these few points need to be clarified about the table:

- There are two types of position adjustments made by the loader agent; position adjustment A is performed when the bucket is filled with a quantity of soil, while position adjustment B is performed when the bucket is empty. This distinction

was made to make possible the option of using a longer duration for the position adjustment when the bucket contains soil.

- For the 6<sup>th</sup> transition, since the truck has just arrived for loading, it may appear that the only valid option is the one in the second row (Truck's *Available Space* > *Carried Earth*). However, designing the state chart to accommodate the other alternative (Truck's *Available Space* ≤ *Carried Earth*) does not affect the logic and is more generic.
- For the second alternative of the 6<sup>th</sup> and 10<sup>th</sup> transitions (Truck's *Available Space* > *Carried Earth*), a quantity of (*Carried Earth* - *Unloading Quantity*) is added back to the *Soil Ready for Loading* to indicate that the extra quantity of carried soil, which could not be added to the truck because it was full, is still in the operation and will be used for upcoming loading activities of other trucks.
- The condition (*Soil Ready for Loading* = 0) was added to the conditions of the 11<sup>th</sup> transition to guide the interaction between the loader and hauler agents in a way that when there is no soil available for the loader to use in filling the truck, the loading activity is completed and the truck should be able to leave the loading area.
- It can be observed in the table that the loader is controlling its interaction with the hauler. Hence, the loader is using some of the hauler's properties in its state chart's conditions and actions. Although this interaction depends on the properties of both equipment units, the hauler cannot tell how much soil it is being filled with or when it reaches its full capacity. So, the loader has the overall authority over the loading activity, and the hauler acts only based on commands from the

loader. This will be further explained when discussing the hauler's state chart and transitions in Figure 3-6 and Table 3-9.

**Table 3-7:** Loader's state chart transitions

Transition	Condition/s	Action/s
1	$SRL = 0$	No action
2A	$SRL > 0$ & $SRL > BC$	$CE = BC$ $TLB = TLFB$ Start filling bucket
2B	$SRL > 0$ & $SRL \leq BC$	$CE = SRL$ $TLB = TLFB \times \frac{CE}{BC}$ Start filling bucket
3	Bucket filling completed	$SRL$ reduced by $CE$ Start adjusting position (A)
4	Position adjustment (A) completed	Start waiting for Truck
5	Truck unavailable	No action
6A	Truck available & Truck's $AS > CE$	$UQ = CE$ $TUB = TUFB$ Start dumping $UQ$ in truck
6B	Truck available & Truck's $AS \leq CE$	$UQ = \text{Truck's } AS$ $TUB = TUFB \times \frac{UQ}{BC}$ Start dumping $UQ$ in truck $SRL$ increased by $(CE - UQ)$

Transition	Condition/s	Action/s
7	$UQ$ dumped in truck & Truck's $AS > 0$ & $SRL > 0$	Truck's $AS$ reduced by $UQ$ Truck's $HCE$ increased by $UQ$ Start adjusting position (B)
8A	Position adjustment (B) completed & $SRL > BC$	$CE = BC$ $TLB = TLFB$ Start filling bucket
8B	Position adjustment (B) completed & $SRL \leq BC$	$CE = SRL$ $TLB = TLFB \times \frac{CE}{BC}$ Start filling bucket
9	Bucket filling completed	$SRL$ reduced by $CE$ Start adjusting position (A)
10A	Position adjustment (A) completed & Truck's $AS > CE$	$UQ = CE$ $TUB = TUF B$ Start dumping $UQ$ in truck
10B	Position adjustment (A) completed & Truck's $AS \leq CE$	$UQ = \text{Truck's } AS$ $TUB = TUF B \times \frac{CE}{BC}$ Start dumping $UQ$ in truck $SRL$ increased by $(CE - UQ)$
11	$UQ$ dumped in truck & Truck's $AS = 0$ or $SRL = 0$	$SRL$ reduced by $CE$ Truck's $AS$ reduced by $UQ$ Truck's $HCE$ increased by $UQ$ Start adjusting position (B)
12	Position adjustment (B) completed	No action



### **The Hauler Agent:**

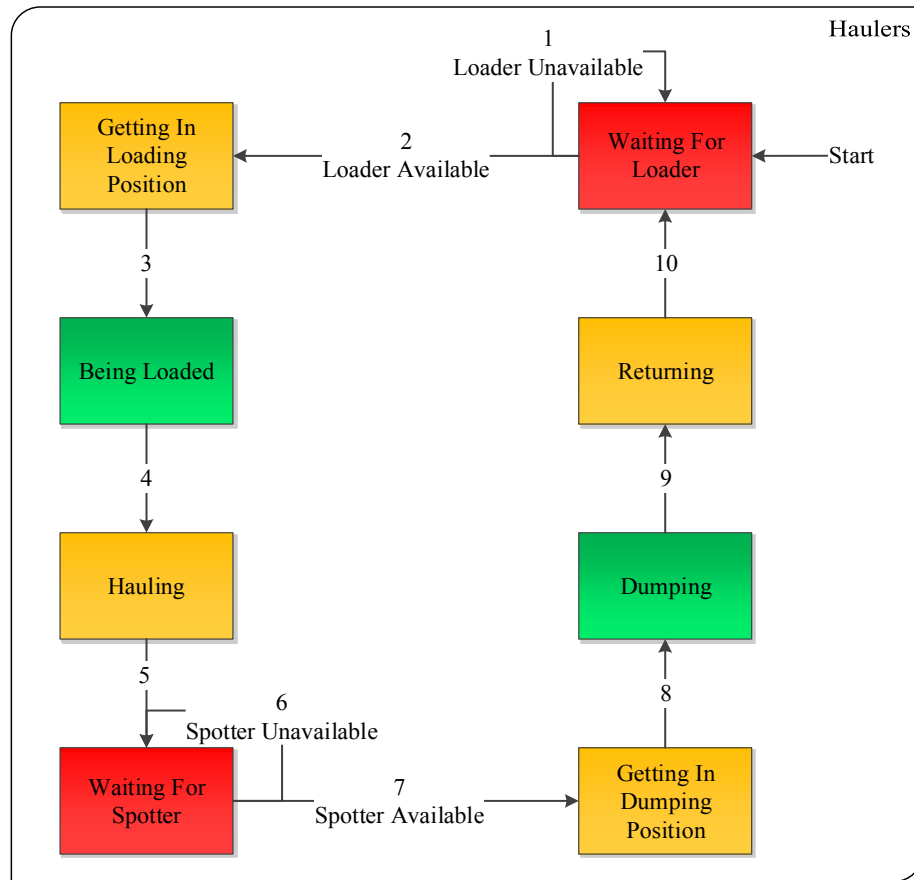
The attributes and variables of the hauler agent are demonstrated in Table 3-8. *Capacity* refers to the maximum quantity of soil that the hauler can carry. *Time to Get in Load Position* is the time it takes for the hauler to leave the queue and arrive at the loading area. *Hauling Duration* refers to the duration of the hauler's trip from the loading area to the dumping site (truck dumping queue). Similarly, *Returning Duration* refers to the duration of the hauler's trip from the dumping area to the loading area (truck loading queue). Accordingly, it is important to note that these two attributes depend on the properties of both the hauler itself and the hauling and returning roads. However, in the proposed AB model, the trip durations are assumed to be attributes of the hauler agent. *Time to Get in Dump Position* is the time it takes for the hauler to leave the queue and arrive at the dumping area. *Dumping Duration* refers to the time it takes for the truck to dump its carried load in a stockpile. In addition, an important observation on the hauler's attributes is that unlike *Dumping Duration*, the truck's loading duration is not an attribute. This is because the hauler agent does not govern its loading activity, which is under the control of the loader agent.

Moreover, *Carried Earth* represents the actual quantity of soil carried by the hauler. This variable is added as an updating mechanism of the quantity of soil present in the hauler at different stages of its cycle. *Available Space* is a variable dependent on *Carried Earth*. Always, the hauler's *Available Space* is equal to its *Capacity* less its *Carried Earth*. *Available Space* is a very important variable for the loader's agent due to the influence it has on the transition's conditions and actions of its state charts, as proven by Table 3-7.

**Table 3-8:** Hauler’s attributes and variables

Agent	Attributes	Variables
Hauler/s	<ul style="list-style-type: none"> <li>• Capacity (C)</li> <li>• Time to Get in Load Position (TGLP)</li> <li>• Hauling Duration (HD)</li> <li>• Time to Get in Dump Position (TGDP)</li> <li>• Dumping Duration (DD)</li> <li>• Returning Duration (RD)</li> </ul>	<ul style="list-style-type: none"> <li>• Carried Earth (HCE)</li> <li>• Available Space (AS)</li> </ul>

The state chart of the hauler agent is illustrated by Figure 3-6. The initial state represents the hauler waiting in the queue for loaders to become idle. Once a loader becomes idle, it signals the hauler to move to the loading area. The hauler then moves and gets in the loading position. At this point, the loading activity begins and the loading cycles described in the loader’s state chart start to be executed. As indicated when describing the loader’s state chart transitions in Table 3-7, when the hauler reaches its *Capacity* or when there is no *Soil Ready for Loading*, the interaction between the loader and hauler agents is finalized. Subsequently, the hauler starts hauling to the dumping location, where it stops first at a queue of haulers waiting for a spotter to become idle. Once a spotter becomes idle, it signals the hauler to move to the dumping area. The hauler then moves and gets in the dumping position. At that point, the hauler starts dumping its load with the help of the spotter, and once its load is fully dumped, the spotter signals the truck to leave, ending the interaction between the two agents. The hauler then heads back to the loading area, where it stops first at a queue waiting for loaders to become idle and starts repeating the cycle again.



**Figure 3-6:** Hauler's state chart

Table 3-9 demonstrates the transitions and states of the hauler agent's state chart. The state chart is simple, owing to the fact that the hauler is not in control of its loading and dumping activities. As mentioned earlier, the loader agent performs all the actions in the loading activity. Similarly, the hauler needs the permission of the spotter before coming to or leaving the dumping area. However, unlike the loading activity, the dumping activity relies solely on properties of the hauler agent. The spotter only acts as a regulator for the dumping activity.

**Table 3-9:** Hauler’s state chart transitions

<b>Transition</b>	<b>Condition</b>	<b>Action</b>
1	Loader unavailable	No action
2	Loader available	Start getting in loading position
3	Permission from loader	No action
4	Permission from loader	Start hauling
5	Arriving at dumping queue	No action
6	Spotter unavailable	No action
7	Spotter available	Start getting in dumping position
8	Permission from spotter	Start dumping load
9	Permission from spotter	$HCE = 0$ $DS$ increased by $HCE$ Start returning
10	Arriving at loading queue	No action

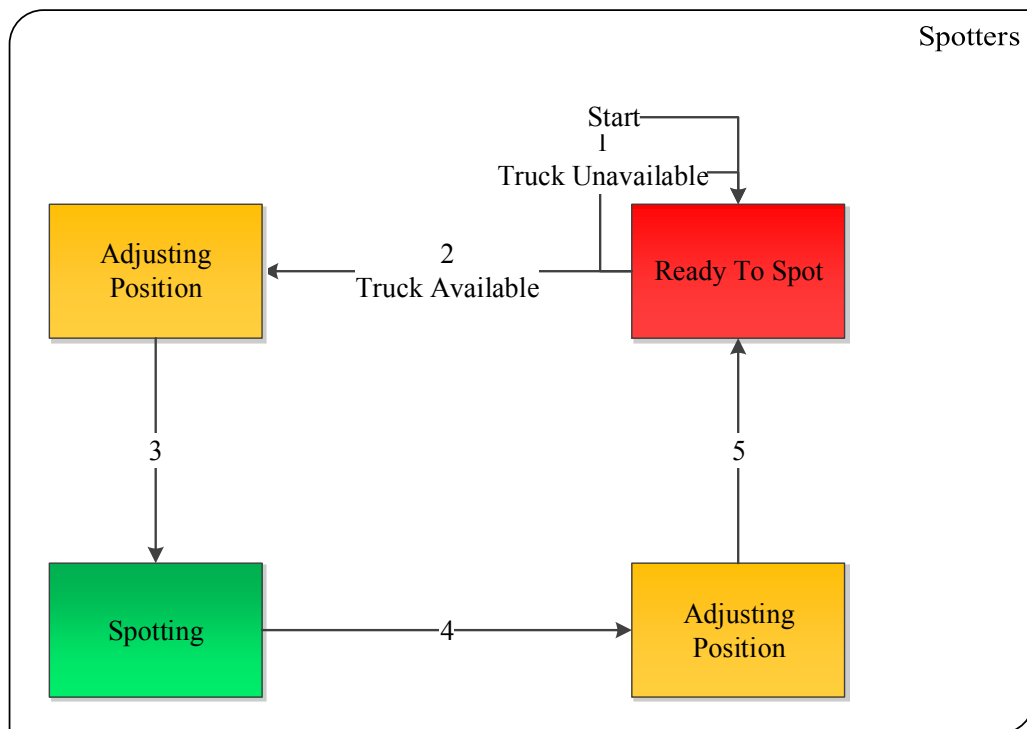
**The Spotter Agent:**

The spotter agent is the most basic agent in the proposed AB model. Its only attribute, *Time to Adjust Position*, is depicted in Table 3-10. This attribute refers to the time it takes for the spotter agent to adjust its position before and after a dumping activity to accommodate arriving and departing haulers.

**Table 3-10:** Spotter's attributes and variables

Agent	Attributes	Variables
Spotter/s	Time to Adjust Position (STAP)	None

The state chart of the spotter agent is demonstrated in Figure 3-7. The initial state represents the spotter waiting for a truck ready to dump its load. The spotter checks the truck's queue, and if there are no trucks waiting to dump, it remains idle. If, however, trucks are available, the hauler agent signals the first truck in the queue to move to the dumping area, while adjusting its position to engage with the arriving truck. When the truck dumps its load, the spotter agent gives it the permission to leave the dumping area and return back to be loaded again. After signaling the truck to depart, the spotter agent adjusts its position again and waits for the next truck to arrive.



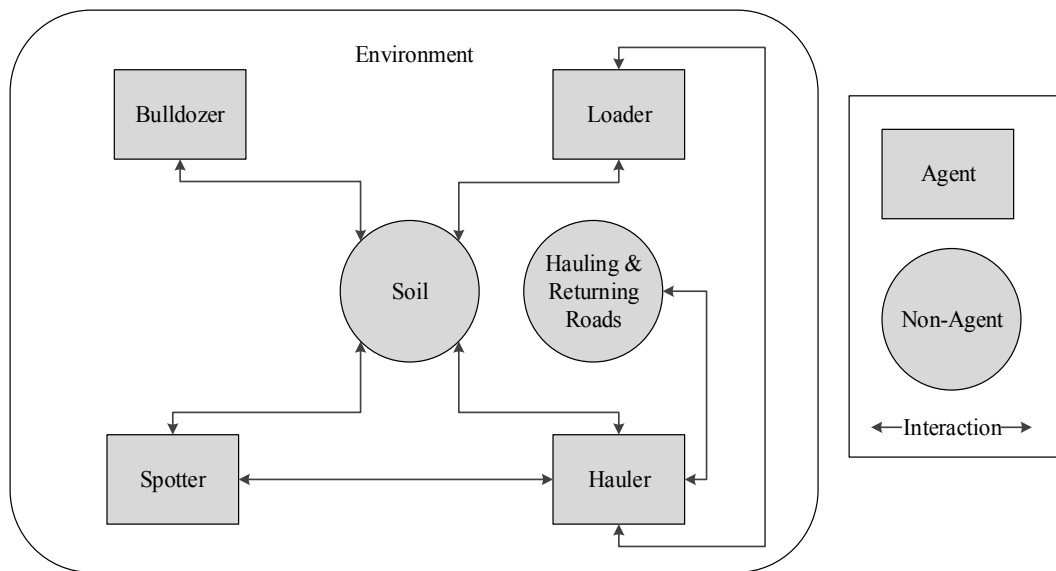
**Figure 3-7:** Spotter's state chart

Table 3-11 represents the transitions' conditions and actions of the spotter agent's state chart. The spotter in the proposed AB model has limited interactions and responsibilities, which can be clearly verified by its simple transitions displayed in Table 3-11.

**Table 3-11:** Spotter's state chart transitions

Transition	Condition	Action
1	Truck unavailable	No action
2	Truck available	Start adjusting position
3	Position adjustment completed	Start spotting
4	Hauler dumping completed	Start adjusting position
5	Position adjustment completed	No action

Figure 3-8 summarizes the interaction between different participants, both agents and non-agents, in the proposed earthmoving AB model.



**Figure 3-8:** Summary of agents' interactions in the proposed earthmoving AB model

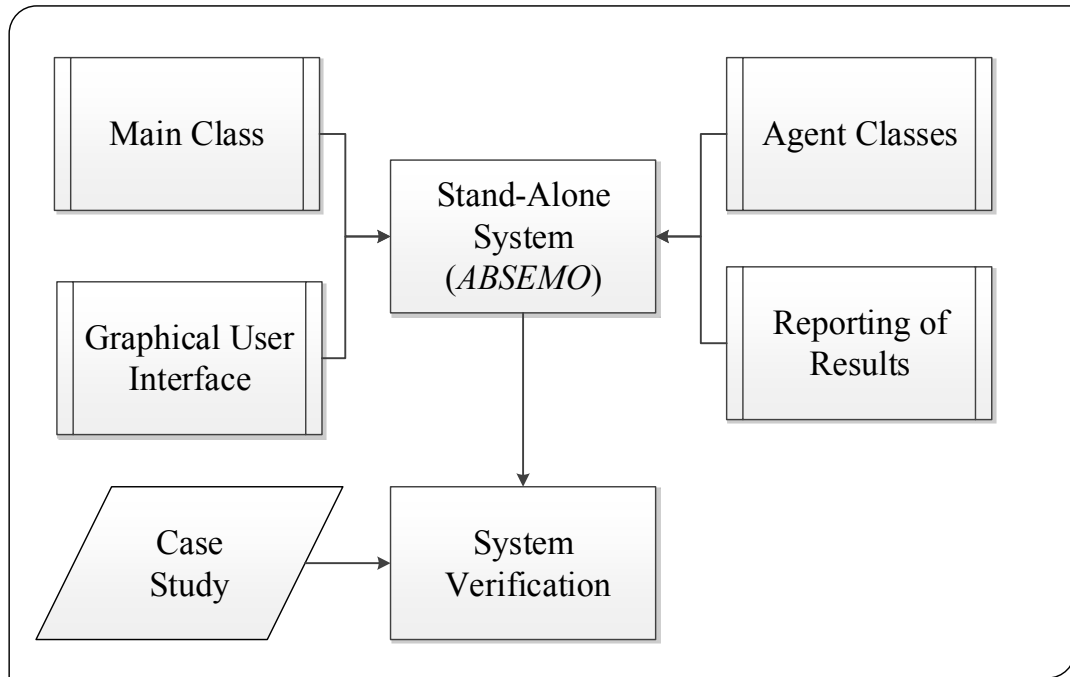
## CHAPTER 4: IMPLEMENTATION

### 4.1 Chapter Overview

This chapter presents the implemented system and describes its components. Accordingly, it demonstrates the modeling aspects of the simulation process utilizing ABMS and object-oriented programming. Also, it primarily focuses on the development of the stand-alone earthmoving simulation program named Agent-Based Simulator for Earthmoving Operations (*ABSEMO*). Figure 4-1 demonstrates the breakdown of the implementation process. The developed simulation application has some interesting aspects: 1) it is the first ever AB simulation tool to be developed for planning earthmoving operations; 2) it can model different types of equipment units performing the same activity; 3) it can accept stochastic data for the characteristics of equipment units as well as for activity durations. 4) it requires neither knowledge in programming nor simulation from end-users to operate; 5) it is a stand-alone system that can be easily shared and run on different platforms.

AnyLogic 7 was utilized in the development of *ABSEMO*. In light of that, AnyLogic is a Java-based modeling tool that includes libraries for DES, SD and ABMS. Thus, users can use the graphical modeling language of Anylogic to prepare simulation models, with the option of extending the model with additional Java code. The Java nature of AnyLogic allows for custom model extensions via Java coding as well as the creation of Java applications, which can be a basis for decision support tools (Wartha et al. 2002). Those applications can be easily shared and run on any standard browser. AnyLogic elements used in the creation of *ABSEMO* are demonstrated in Table 4-1.

## Implementation



**Figure 4-1:** Implementation breakdown

### 4.2 Main Class

The main class in the developed system represents the simulation engine that integrates all model components, manages the interaction between agents and their environment, performs major simulation actions and generates results and analysis. In view of that, the main class duties can be summarized in five point: 1) setting up the earthmoving environment; 2) creating and governing agent populations; 3) managing agent queues; 4) controlling model run-time conditions; 5) performing analysis and producing results. These tasks are separately explained in subsections 4.2.1 ~ 4.2.5.



**Table 4-1:** AnyLogic elements used in *ABSEMO*

Element	Symbol
Agent population	
Parameter (attribute)	
Variable	
Java function	
Event (scheduled function)	
Collection (linked list)	
Pointer	
Statistics	
Data set	
Excel file	
Bulldozer agent	
Loader agent	
Hauler agent	
Spotter agent	
Link to agent/s	
Link to main class	
Time-triggered transition	
Message-triggered transition	
State-end-triggered transition	

### 4.2.1 Earthmoving Environment

The earthmoving environment of the proposed AB model is portrayed in Figure 4-2. It represents the physical space in which the different activities involved in the earthmoving operation take place. Hence, the excavation area, hauler load queue, loading area, haul road, hauler dump queue, dumping area and return road are illustrated by Figure 4-2. Other components including a residential area, a commercial area and a body of water were added for aesthetic purposes. Pointers are placed at different locations in the environment to guide the movement of agents during different activities.

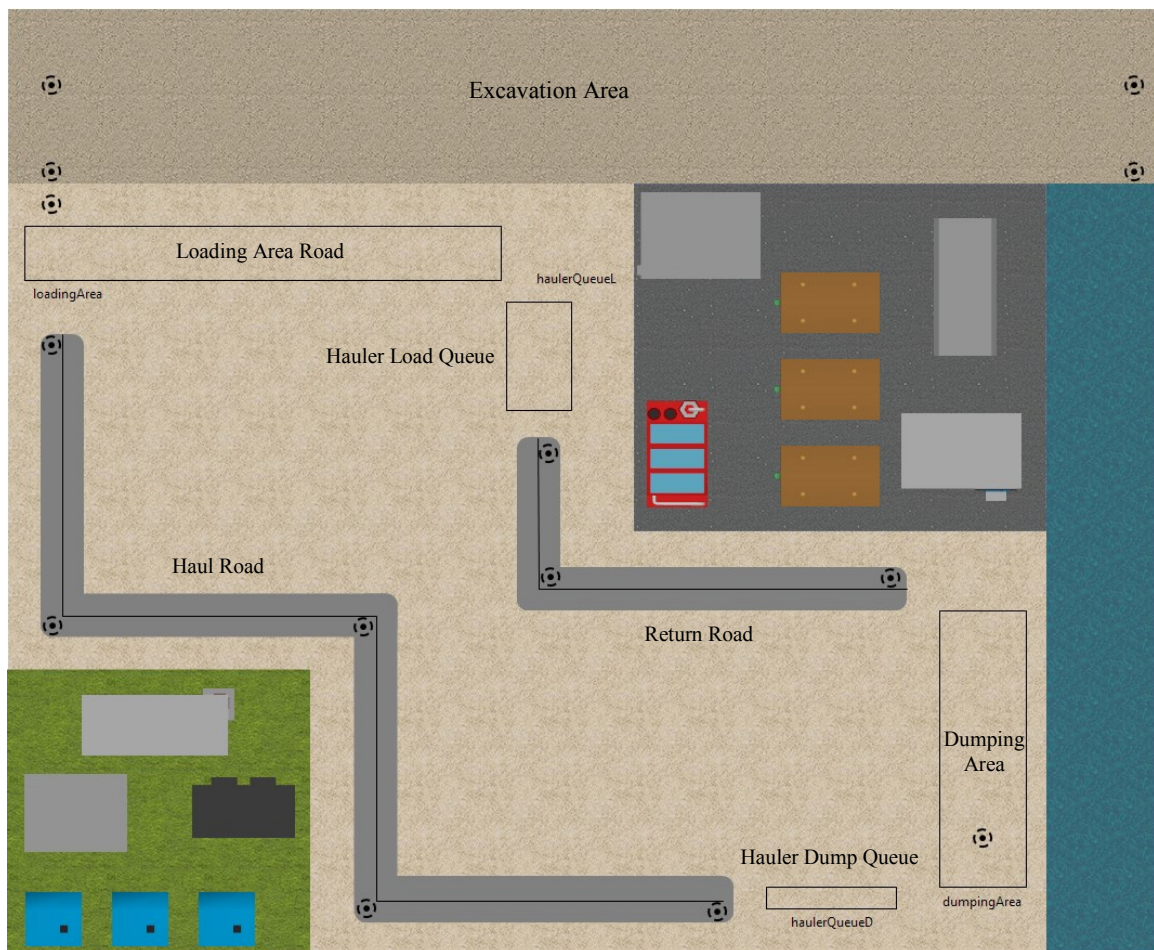











Figure 4-2: Earthmoving Environment

## 4.2.2 Agent Populations

Although the structure of each agent is represented by its state chart, agent populations are created in the main the class and they live and interact in its environment. Some major elements that were utilized in configuring and managing agent populations in the main class are listed and explained in Table 4-2. These elements mainly focus on creating agents, setting up their initial locations and managing their populations.









**Table 4-2:** Agent populations' elements in *ABSEMO*

Element	Purpose
 bulldozersTypeConfiguration	Creates bulldozer agents with types and properties specified by the user
 loadersTypeConfiguration	Creates loader agents with types and properties specified by the user
 haulersTypeConfiguration	Creates hauler agents with types and properties specified by the user
 spottersTypeConfiguration	Creates spotter agents with types and properties specified by the user
 startUpLocations	Sets up the initial location of agents in the environment
 bulldozers [..]	Represents the created population of bulldozers as one group for control purposes
 loaders [..]	Represents the created population of loaders as one group for control purposes
 haulers [..]	Represents the created population of haulers as one group for control purposes
 spotters [..]	Represents the created population of spotters as one group for control purposes

### 4.2.3 Agents' Queues

The function and queues in the main class, depicted in Table 4-3, govern the hauler agent's interaction with the loader agent, when it attains the point of being loaded, and with the spotter agent, when it reaches the step of dumping its load. These functions and queues are necessary prior to the loading and dumping activities. Before trucks get to the loading and dumping areas, queues, represented in the form of requests, need to be organized based on the First-In-First-Out (FIFO) principle.

















**Table 4-3:** Agent queues' elements in *ABSEMO*

Element	Purpose
 <code>haulerRequestsLoader</code>	Places a request for loading by the hauler agent as soon as it arrives at the loading queue
 <code>haulerRequestsSpotter</code>	Places a request for dumping by the hauler agent as soon as it arrives at the dumping queue
 <code>haulerQueueAtLoaders</code>	Stores the hauler agents' loading requests in a list
 <code>haulerQueueAtSpotters</code>	Stores the hauler agents' dumping requests in a list
 <code>thereAreRequestsLoader</code>	Called by the loader agent when it becomes idle to check if there are any stored loading requests
 <code>thereAreRequestsSpotter</code>	Called by the spotter agent when it becomes idle to check if there are any stored dumping requests
 <code>getRequestLoader</code>	Called by the loader agent to remove the first request in the list after responding to it
 <code>getRequestSpotter</code>	Called by the loader agent to remove the first request in the list after responding to it

#### 4.2.4 Model Run Control

The main class includes elements that are defined by the author to control the model run time, by pausing or stopping simulation based on conditions specified by the user. These elements are shown in Table 4-4. Hence, different elements are added to accommodate different choices of the user regarding model pause, resume and stop mechanisms.














**Table 4-4:** Model run control in *ABSEMO*

























Element/s	Purpose
 pauseAndResumeFunction  pauseAndResume	Pause and resume the simulation
 applyWholeQuantityDumpedOption  stopWhenWholeQuantityDumped	Stop the simulation when the whole quantity of soil is excavated, transported and dumped
 applyDumpedSoilOption  dumpedSoilToStopRun  stopAtDumpedQuantity	Stop the simulation when a specific quantity of soil is dumped
 applyExcavatedSoilOption  excavatedSoilToStopRun  stopAtExcavatedQuantity	Stop the simulation when a specific quantity of soil is excavated
 applyStopAtTimeOption  stopAtTimeEnabled  hoursToStopRun  minutesToStopRun  secondsToStopRun  stopAtTime	Stops the simulation at a specific point of time, which is defined in terms of hours, minutes and seconds of simulation time










#### 4.2.5 Results and Analysis

The main class is also responsible for updating major soil quantities, calculating productivity, gathering statics related to equipment utilization and updating the Excel sheets linked to the model. Such types of analysis and results performed by *ABSEMO* are depicted in Table 4-5.

**Table 4-5:** Results and Analysis in *ABSEMO*

Element/s	Purpose
 soilAvailableForExcavation	Refers to the quantity of soil available for excavation by bulldozers
 soilAvailableForLoading	Refers to the quantity of soil available for loading (or excavation) by loaders
 dumpedSoil	Refers to the quantity of soil dumped by haulers
 updateProductivity	Updates productivity every minute of simulation time
 productivity	Refers to the productivity of the system, which is obtained by dividing the quantity of dumped soil by the simulation time
 bulldozersIdle  bulldozersWorking  bulldozersExcavating  bulldozersRepositioning	Gather statistics on the average time spent by bulldozer agents at each state during simulation
 loadersIdle  loadersWorking  loadersLoading  loadersUnloading	Gather statistics on the average time spent by loader agents at each state during simulation

Element/s	Purpose
<ul style="list-style-type: none"> <li> haulersIdle</li> <li> haulersWorking</li> <li> haulersBeingLoaded</li> <li> haulersHauling</li> <li> haulersDumping</li> <li> haulersReturning</li> </ul>	<p>Gather statistics on the average time spent by hauler agents at each state during simulation</p>
<ul style="list-style-type: none"> <li> spottersIdle</li> <li> spottersWorking</li> <li> spottersSpotting</li> <li> spottersAdjusting</li> </ul>	<p>Gather statistics on the average time spent by spotter agents at each state during simulation</p>
<ul style="list-style-type: none"> <li> bulldozersIdleSet</li> <li> bulldozersWorkingSet</li> <li> bulldozersExcavatingSet</li> <li> bulldozersRepositioningSet</li> </ul>	<p>Create data sets for the average time spent in states obtained by the bulldozer agent's statistics</p>
<ul style="list-style-type: none"> <li> loadersIdleSet</li> <li> loadersWorkingSet</li> <li> loadersLoadingSet</li> <li> loadersUnloadingSet</li> </ul>	<p>Create data sets for the average time spent in states obtained by the loader agent's statistics</p>
<ul style="list-style-type: none"> <li> haulersIdleSet</li> <li> haulersWorkingSet</li> <li> haulersBeingLoadedSet</li> <li> haulersHaulingSet</li> <li> haulersDumpingSet</li> <li> haulersReturningSet</li> </ul>	<p>Create data sets for the average time spent in states obtained by the hauler agent's statistics</p>

Element/s	Purpose
 spottersIdleSet  spottersWorkingSet  spottersSpottingSet  spottersAdjustingSet	Create data sets for the average time spent in states obtained by the spotter agent's statistics
 updateStatistics	Updates the statistics of equipment utilization every 100 minutes of simulation time
 bulldozersExcel	Updates and excel file with statistics related to the utilization of bulldozer agents
 loadersExcel	Updates and excel file with statistics related to the utilization of loader agents
 haulersExcel	Updates and excel file with statistics related to the utilization of hauler agents
 spottersExcel	Updates and excel file with statistics related to the utilization of spotter agents

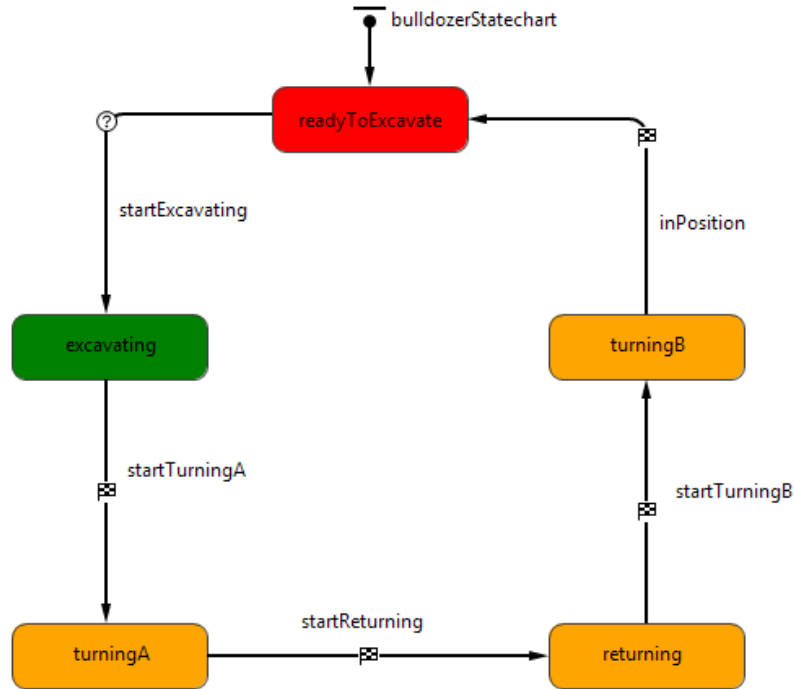
### 4.3 Agent Classes

Besides the main class, *ABSEMO* has four classes that represent the three equipment units and the spotters participating in the earthmoving operation. Each of these classes contains an agent's state chart, along with elements required in the operation of that state chart. Consequently, the same states, transitions, attributes and variables defined earlier in the methodology are used in the implementation of agent classes in *ABSEMO*. Some extra elements are added to the implemented system for programming details. Subsections 4.3.1 ~ 4.3.4 discuss each agent class individually. The state chart of the agent is displayed, followed by a table that explains major elements used in the agent's structure.



### 4.3.1 The Bulldozer Agent

Figure 4-3 demonstrates the state chart of the bulldozer agent, as implemented in *ABSEMO*.



















**Figure 4-3:** Bulldozer’s state chart in *ABSEMO*

Table 4-6 illustrates the major elements used in the structure of the bulldozer agent. The bulldozer agent does not have direct interactions with other agents in the proposed model. Two properties of the bulldozer agent are considered as variables.

**Table 4-6:** Bulldozer class elements in *ABSEMO*

Element	Purpose
colorB	Changes the color of the bulldozer agent based on its state
indexNumber	Refers to the ID of the bulldozer agent in the bulldozer population for assigning initial locations

Element	Purpose
 bulldozerCapacityUser	The push capacity of the bulldozer at the end of a full pass, as specified by the user
 timeToExcavateUser	The full pass duration of the bulldozer, as specified by the user
 timeToTurnUser	The direction change duration of the bulldozer, as specified by the user
 timeToReturnUser	The return movement duration of the bulldozer, as specified by the user
 bulldozerCapacityF	Transforms the user's input on the bulldozer's push capacity into a parameter usable by the bulldozer's state chart
 timeToExcavateF	Transforms the user's input on the bulldozer's full pass duration into a parameter usable by the bulldozer's state chart
 timeToTurnF	Transforms the user's input on the bulldozer's direction change duration into a parameter usable by the bulldozer's state chart
 timeToReturnF	Transforms the user's input on the bulldozer's return movement duration into a parameter usable by the bulldozer's state chart
 bulldozerCapacity	The push capacity of the bulldozer used in the bulldozer's state chart
 timeToExcavate	The full pass duration of the bulldozer used in the bulldozer's state chart
 timeToTurn	The direction change duration of the bulldozer used in the bulldozer's state chart
 timeToReturn	The return movement duration of the bulldozer used in the bulldozer's state chart
 excavationType	Determines the excavation quantity of the bulldozer based on the available soil
 excavationQuantity	The actual excavation quantity of the bulldozer
 excavationDuration	The actual excavation duration of the bulldozer
	Establishes a link between the agent class of the bulldozer and the main class

### 4.3.2 The Loader Agent

Figure 4-4 represents the state chart of the loader agent as implemented in *ABSEMO*.

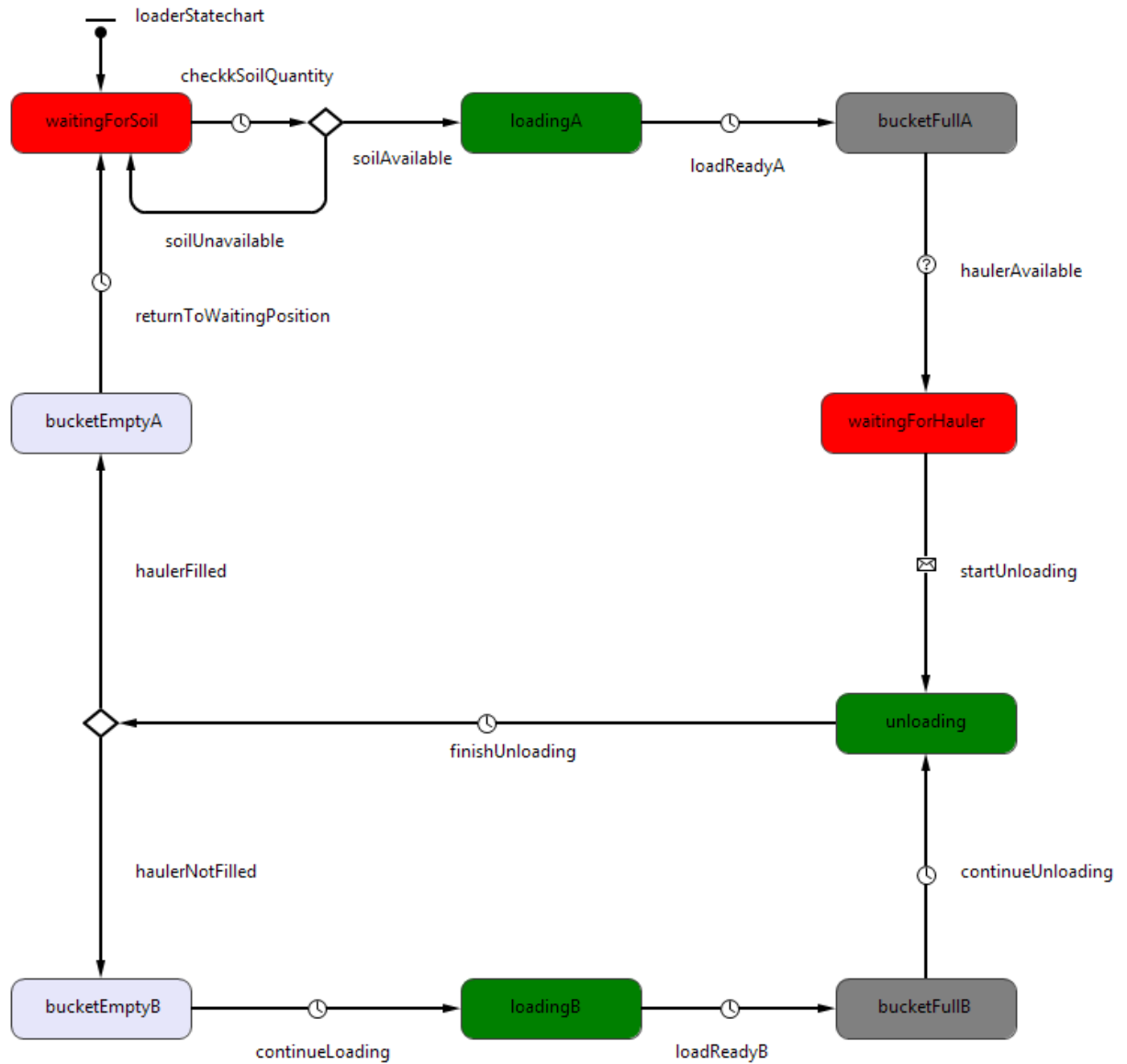



























Figure 4-4: Loader's state chart in *ABSEMO*

Elements used in the creation of the loader agent are displayed in Table 4-7. The loader agent has four changing properties (variables) and interacts only with the hauler agent.

**Table 4-7:** Loader class elements in *ABSEMO*

Element	Purpose
 colorL	Changes the color of the loader agent based on its state
 bucketCapacityUser	The bucket capacity of the loader, as specified by the user
 timeToLoadFullBucketUser	The bucket load duration of the loader, as specified by the user
 timeToAdjustWhileFullUser	The position adjustment duration of the loader while its bucket is full, as specified by the user
 timeToUnloadFullBucketUser	The bucket unload duration of the loader, as specified by the user
 timeToAdjustWhileEmptyUser	The position adjustment duration of the loader while its bucket is empty, as specified by the user
 bucketCapacityF	Transforms the user's input on the loader's bucket capacity into a parameter usable by the loader's state chart
 timeToLoadFullBucketF	Transforms the user's input on the loader's bucket load duration into a parameter usable by the loader's state chart
 timeToAdjustWhileFullF	Transforms the user's input on the loader's position adjustment while full duration into a parameter usable by the loader's state chart
 timeToUnloadFullBucketF	Transforms the user's input on the loader's bucket unload duration into a parameter usable by the loader's state chart
 timeToAdjustWhileEmptyF	Transforms the user's input on the loader's position adjustment while empty duration into a parameter usable by the loader's state chart

Element	Purpose
 bucketCapacity	The bucket capacity of the loader used in the loader's state chart
 timeToLoadFullBucket	The bucket load duration of the loader used in the loader's state chart
 timeToAdjustWhileFull	The position adjustment duration of the loader while its bucket is full used in the loader's state chart
 timeToUnloadFullBucket	The bucket unload duration of the loader used in the loader's state chart
 timeToAdjustWhileEmpty	The position adjustment duration of the loader while its bucket is empty used in the loader's state chart
 loadingType	Determines the loading quantity of the loader's bucket based on the available soil
 unloadingType	Determines the required loading quantity of the hauler based on its available space
 carriedEarth	The actual quantity of soil carried by the loader
 unloadingQuantity	The actual quantity of soil in the loader's bucket to be dumped in the truck
 timeToLoadBucket	The actual bucket load duration of the loader
 timeToUnloadBucket	The actual bucket unload duration of the loader
 Hauler	Stores the hauler agent as a variable for communication purposes
 connections	Establishes a link between the agent class of the loader and the agent class of the hauler
	Establishes a link between the agent class of the loader and the main class

### 4.3.3 The Hauler Agent

The state chart of the hauler agent, as implemented in *ABSEMO*, is demonstrated in Figure 4-5.

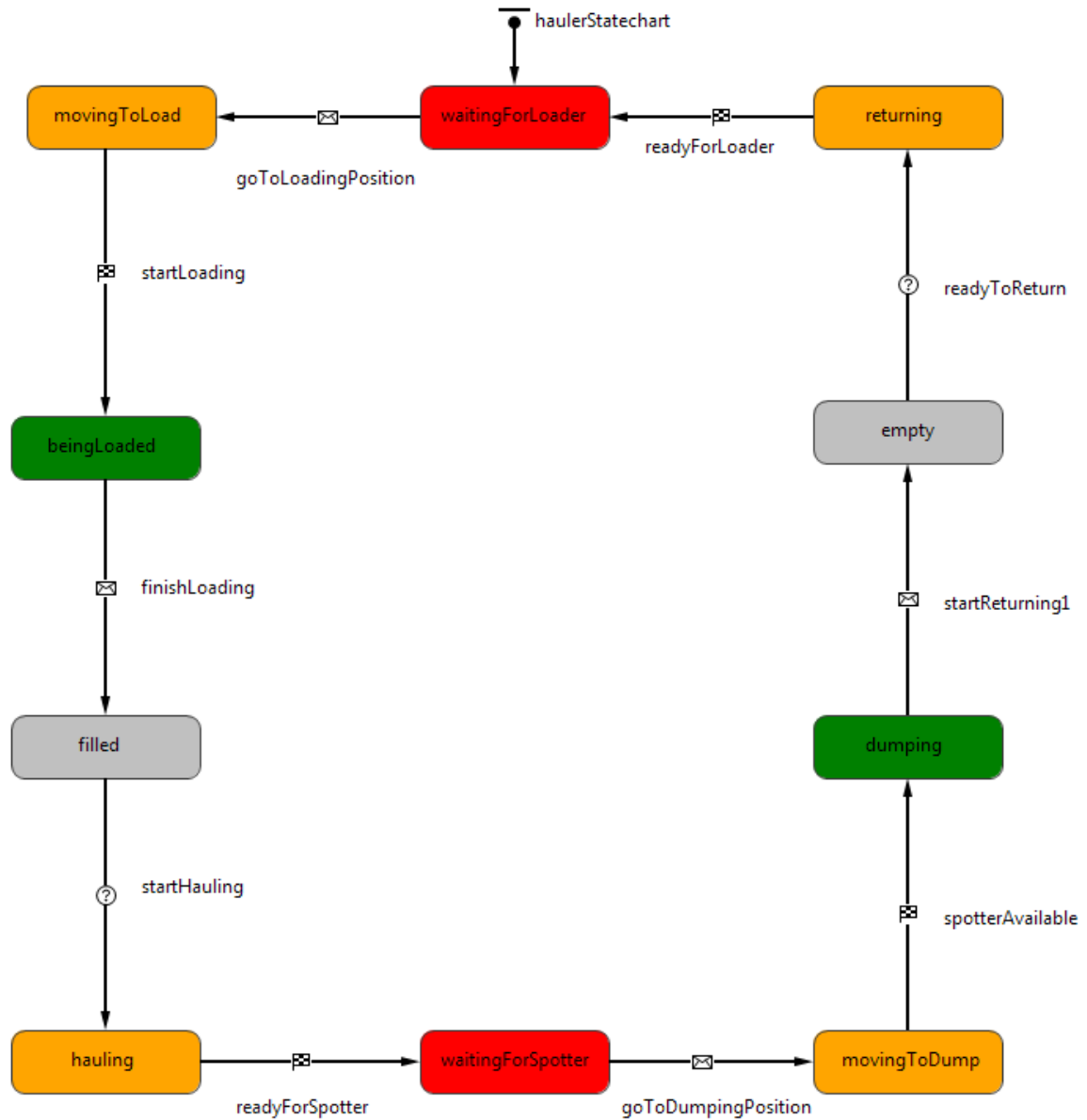



























Figure 4-5: Hauler's state chart in *ABSEMO*

Table 4-8 demonstrates elements used in the structure of the hauler agent. The hauler agent interacts with the loader and spotter agents and has two variables among its properties.

**Table 4-8:** Hauler class elements in *ABSEMO*

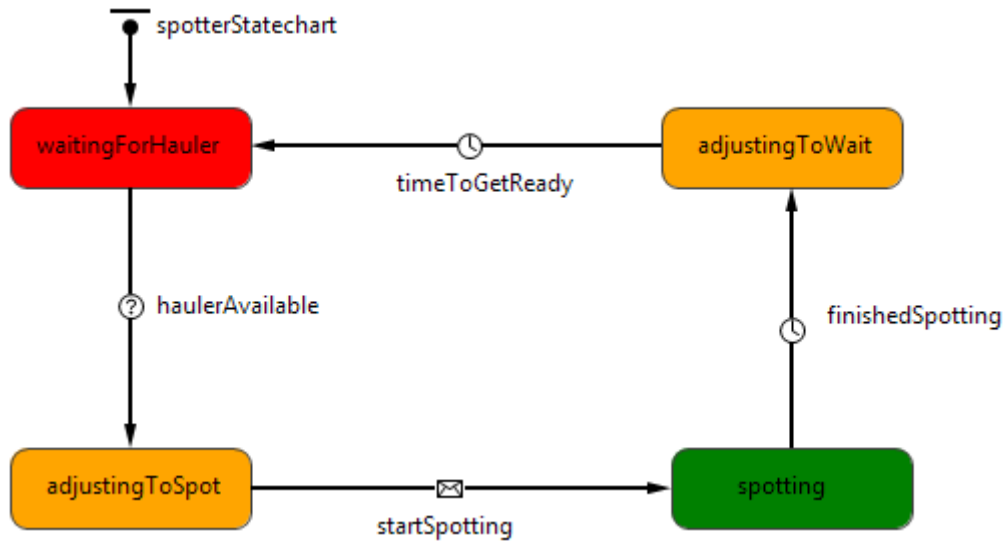
Element	Purpose
 colorH	Changes the color of the hauler agent based on its state
 capacityUser	The capacity of the hauler, as specified by the user
 movingToLoadPositionTimeUser	The get in load position duration of the hauler, as specified by the user
 haulingTimeUser	The haul duration of the hauler to the dumping site, as specified by the user
 movingToDumpPositionTimeUser	The get in dump position duration of the hauler, as specified by the user
 dumpingTimeUser	The dump duration of the hauler, as specified by the user
 returningTimeUser	The return duration of the hauler to the loading site, as specified by the user
 capacityF	Transforms the user's input on the hauler's capacity into a parameter usable by the hauler's state chart
 movingToLoadPositionTimeF	Transforms the user's input on the hauler's get in load position duration into a parameter usable by the hauler's state chart
 haulingTimeF	Transforms the user's input on the hauler's haul duration into a parameter usable by the hauler's state chart
 movingToDumpPositionTimeF	Transforms the user's input on the hauler's get in dump position duration into a parameter usable by the hauler's state chart

Element	Purpose
 dumpingTimeF	Transforms the user's input on the hauler's dump duration into a parameter usable by the hauler's state chart
 returningTimeF	Transforms the user's input on the hauler's return duration into a parameter usable by the hauler's state chart
 capacity	The capacity of the hauler used in the hauler's state chart
 movingToLoadPositionTime	The get in load position duration of the hauler used in the hauler's state chart
 haulingTime	The haul duration of the hauler to the dumping site used in the hauler's state chart
 movingToDumpPositionTime	The get in dump position duration of the hauler used in the hauler's state chart
 dumpingTime	The dump duration of the hauler used in the hauler's state chart
 returningTime	The return duration of the hauler to the loading site used in the hauler's state chart
 carriedEarth	The quantity of soil carried by the hauler
 availableSpace	The available space in the hauler
 Loader	Stores the loader agent as a variable for communication purposes
 Spotter	Stores the spotter agent as a variable for communication purposes
 connections	Establishes a link between the agent class of the hauler and the agent classes of the loader and the spotter
	Establishes a link between the agent class of the hauler and the main class



### 4.3.4 The Spotter Agent

Figure 4-6 represents the state chart of the spotter agent as implemented in *ABSEMO*.










**Figure 4-6:** Spotter’s state chart in *ABSEMO*

Elements used in the construction of the spotter agent are displayed in Table 4-9.

The spotter agent interacts only with the hauler agent and has no variables.

**Table 4-9:** Spotter class elements in *ABSEMO*

Element	Purpose
 colorS	Changes the color of the spotter agent based on its state
 timeToAdjustPositionUser	The position adjustment duration of the spotter, as specified by the user
 timeToAdjustPositionF	Transforms the user’s input on the spotter’s position adjustment duration into a parameter usable by the spotter’s state chart
 timeToAdjustPosition	The position adjustment duration of the spotter used in the spotter’s state chart

Element	Purpose
 Hauler	Stores the hauler agent as a variable for communication purposes
 connections	Establishes a link between the agent class of the spotter and the agent class of the hauler
	Establishes a link between the agent class of the spotter and the main class

#### 4.4 Graphical User Interface (Java Application)

The Graphical User Interface (GUI) of the implemented earthmoving model was developed after finalizing the structure of the main class and the four agent classes. The main purpose of the GUI is to create a user-friendly tool that allows for planning earthmoving operations with flexibility in inputs to fit different case studies.

Figure 4-7 demonstrates the introductory welcome page of *ABSEMO*, where general information about the owners and the application is presented. In addition, Figure 4-8 presents the environment of the earthmoving model and highlights its elements. Figure 4-9 displays the material input page of *ABSEMO*, where information about the material type, quantity of soil to be excavated and quantity of excavated soil to be transported (or quantity of soil to be excavated by loaders) are entered by users. The material type has no effect on the model's structure or operation; it was just added for appearance purposes related to the color of each type of material. On the other hand, the quantity of soil to be excavated refers to the quantity of soil that is planned to be excavated by bulldozers. The quantity of excavated soil to be transported (or quantity of soil to be excavated by loaders) is added to give users the ability to model earthmoving systems that have the following two scenarios: 1) a quantity of soil is already excavated

and ready to be loaded in trucks; 2) the earthmoving operation does not include the utilization of bulldozers, and loaders (or excavators) are the equipment units responsible for excavating and loading soil in trucks. *ABSEMO* accepts and employs inputs of both cases.

Furthermore, Figure 4-10 demonstrates the equipment and labor input page of *ABSEMO*. Users can select up to three types of each equipment unit to participate in the earthmoving operation. Having more than three types of the same equipment unit performing the same task is not realistic. However, the model can be easily upgraded to accommodate more types of equipment units if needed. A color legend is added for users to identify the state of each equipment unit during model run. Figure 4-11 displays the model run control elements in *ABSEMO*. Users can choose to stop the model when the whole quantity of soil is excavated, loaded and dumped, when the dumped or excavated soil reaches a specific quantity, or at a specific time instance specified by hours, minutes and seconds of simulation time.

Figure 4-12 and Figure 4-13 are snapshots of the model run in 2-D and 3-D views. While the model is running, users can monitor major statics including the quantity of soil available for excavation, the quantity of soil available for loading, the quantity of dumped soil, the productivity of work, the number of trucks in the loading queue and the number of trucks in the dumping queue. Users can alternate between the 2-D view, the 3-D view and the metrics view, which will be shown in Section 4.5. Users also have the ability to run the model at a fast speed, pause and resume simulation, or stop and terminate the simulation.

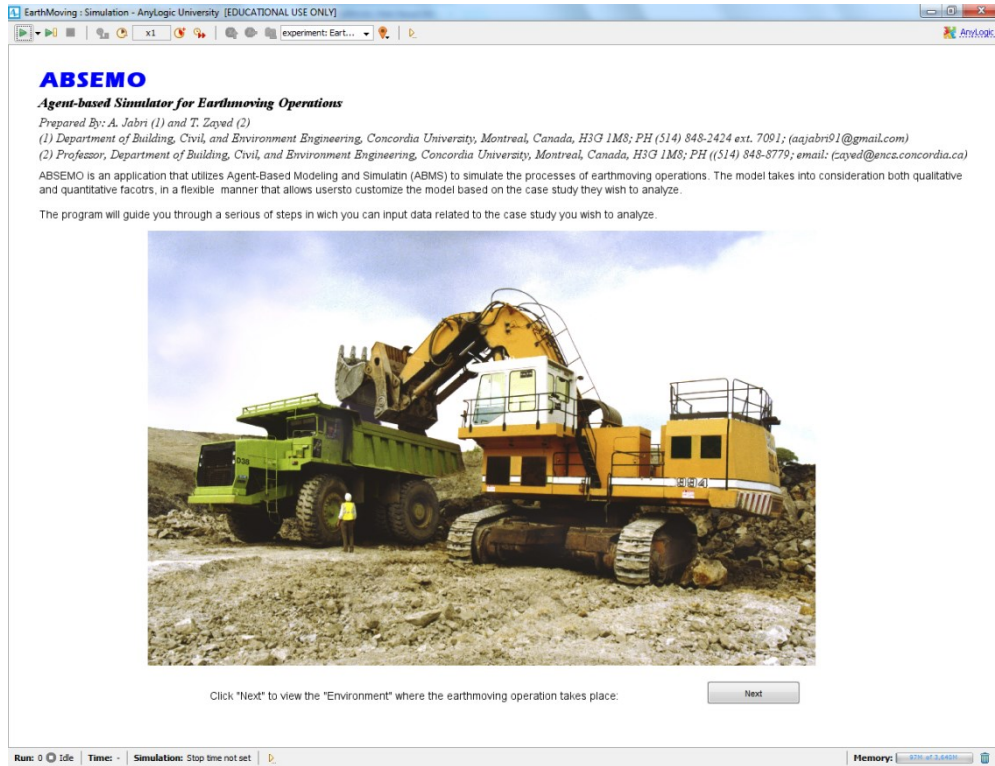


Figure 4-7: Welcome page of *ABSEMO*

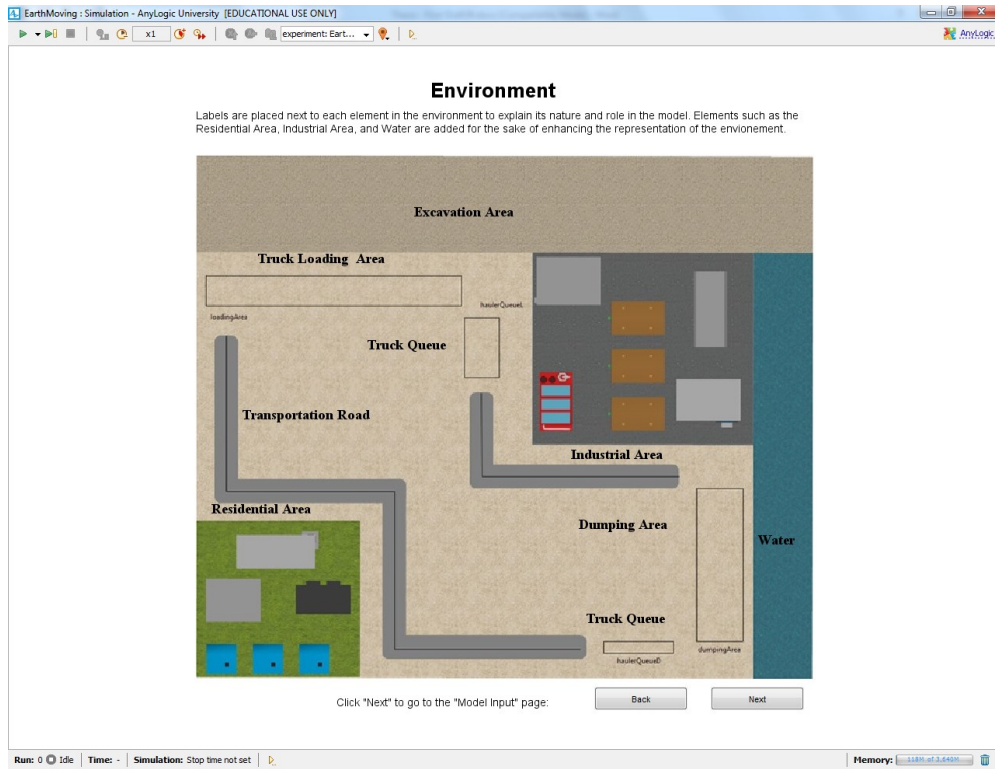


Figure 4-8: Earthmoving environment in *ABSEMO*

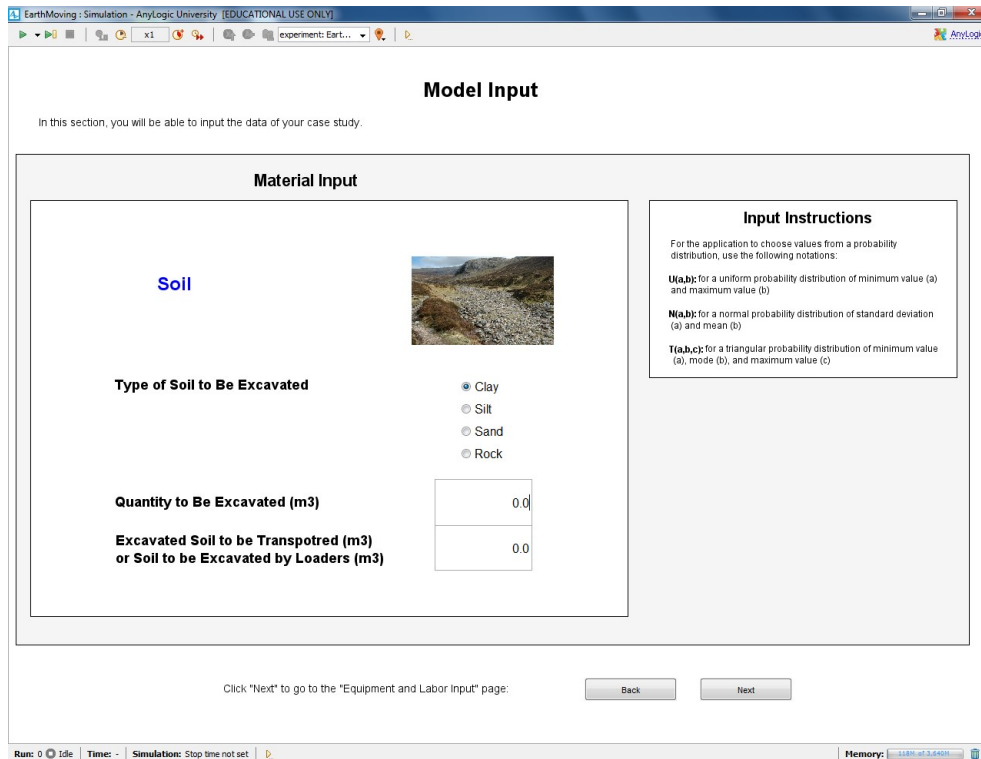


Figure 4-9: Material input in *ABSEMO*

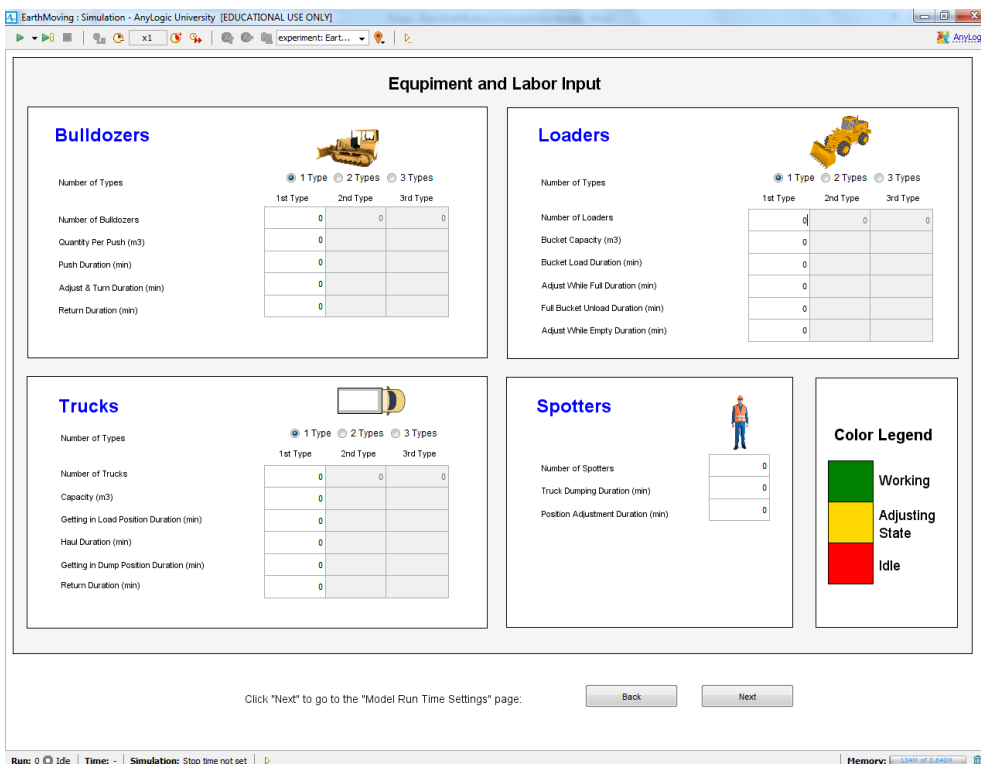


Figure 4-10: Equipment and labor input in *ABSEMO*

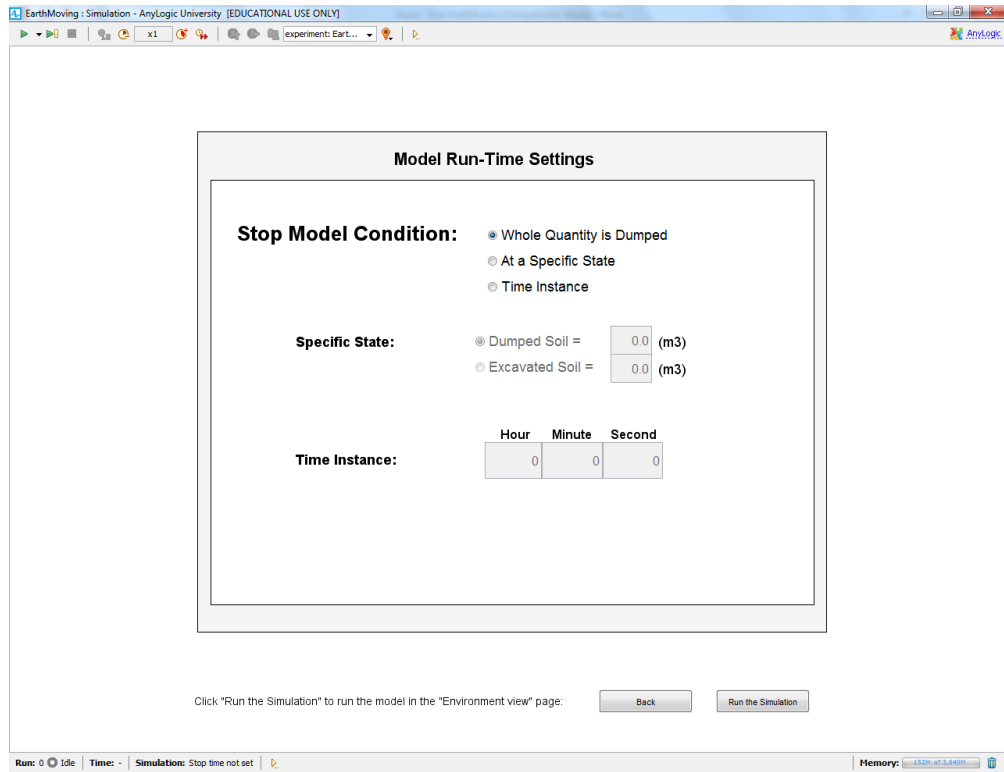


Figure 4-11: Model run control in *ABSEMO*

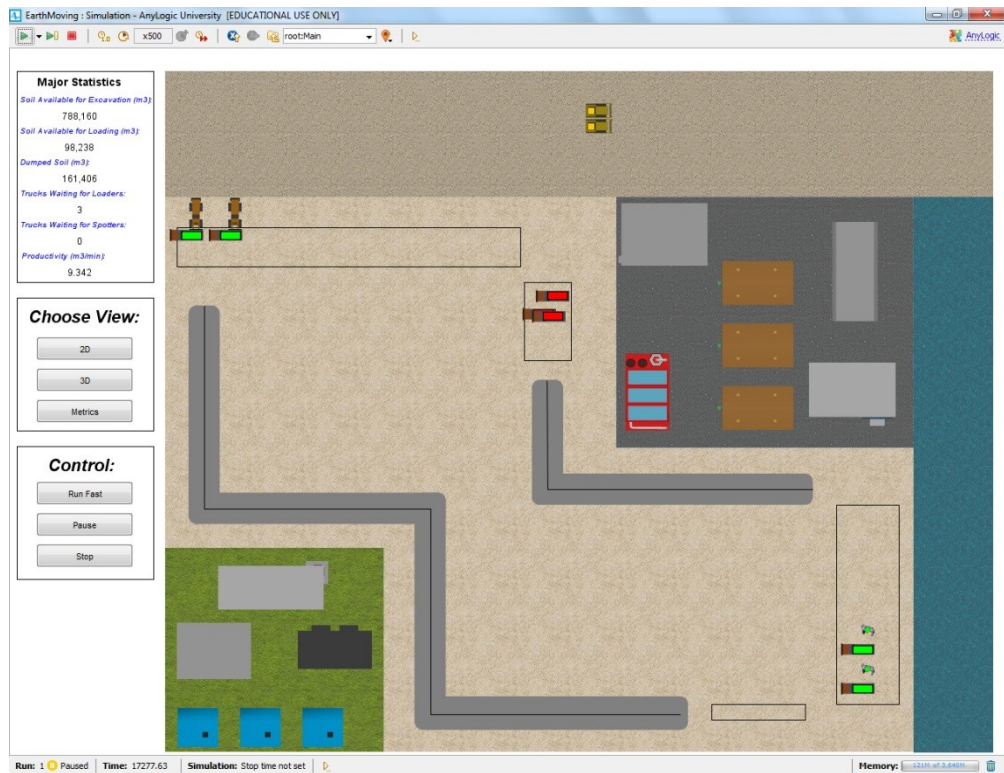
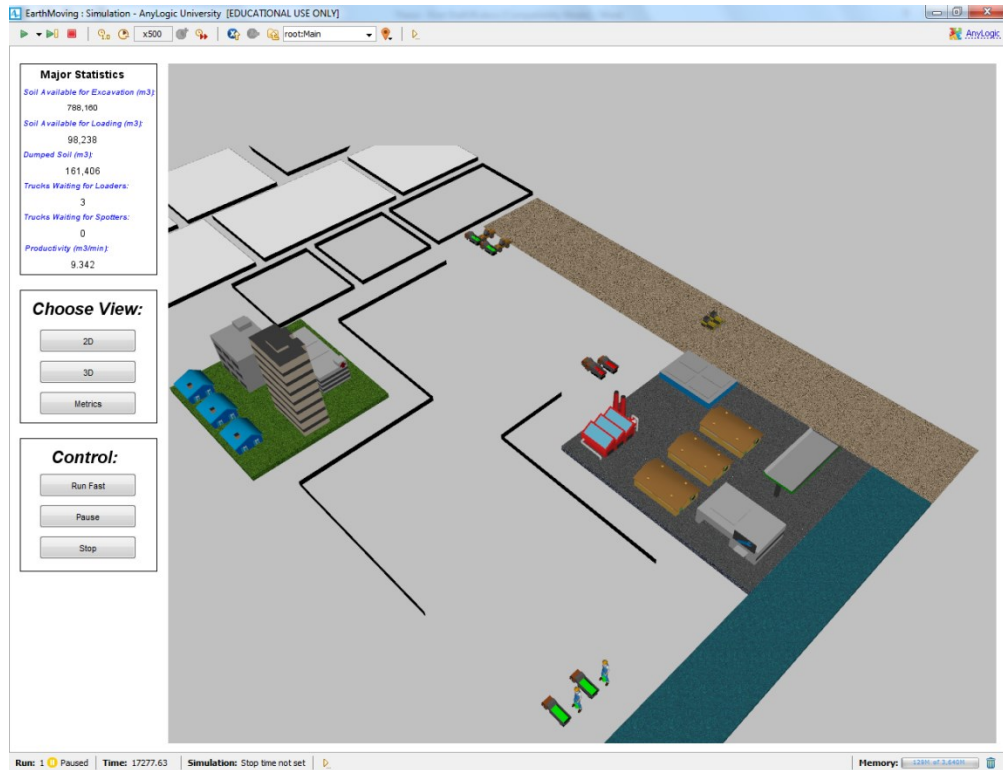


Figure 4-12: 2-D snapshot of model run in *ABSEMO*



**Figure 4-13:** 3-D snapshot of model run in *ABSEMO*

## 4.5 Reporting of Results

*ABSEMO* reports simulation results both during model run and after the simulation is concluded. Major statistics demonstrated in Section 0 are constantly updated while the model is running. Figures 4-14 ~ 4-18 are snap shots of the equipment and labor utilization graphs.

Figure 4-14 represents basic statistics on the average percentage of simulation time in which agent units belonging to the same class are either working or idle. Figure 4-15, Figure 4-16, Figure 4-17 and Figure 4-18 demonstrate a detailed version of the equipment and labor utilization statistics for the bulldozer, loader, hauler and spotter agents respectively. In these detailed graphs, the average percentage of time spent by the

agent group at each state is represented separately and given different colors. Major statistics and equipment utilization data are added to excel sheets which were previously created and linked to *ABSEMO*.

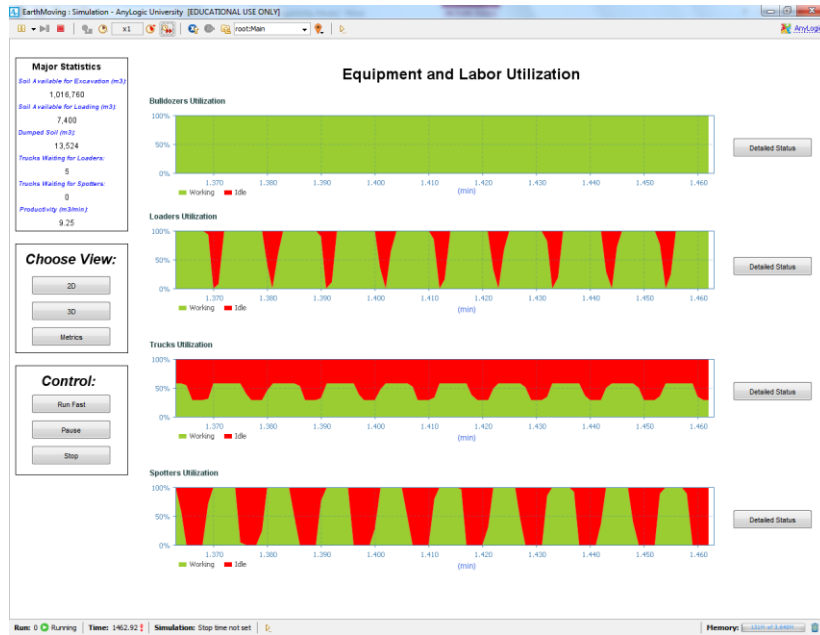


Figure 4-14: Equipment utilization statistics in *ABSEMO*

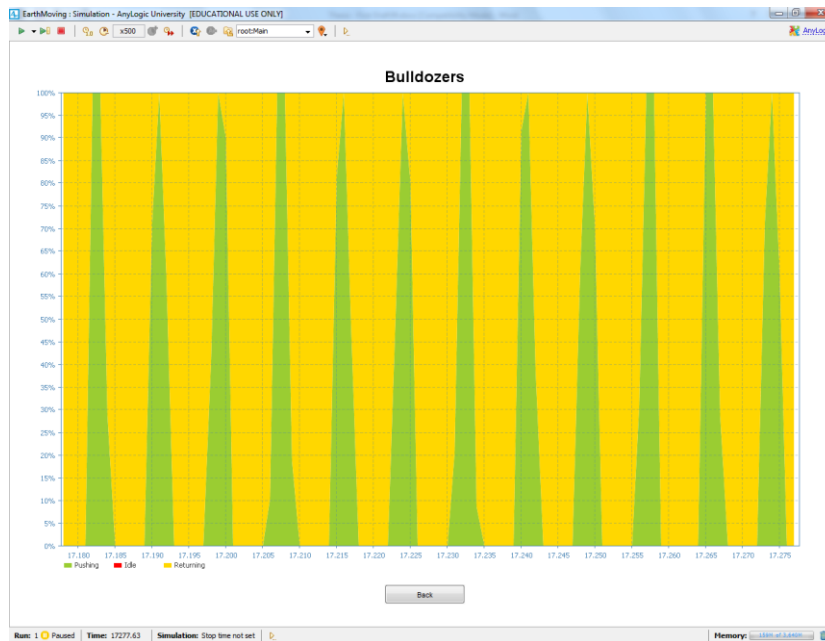


Figure 4-15: Bulldozers detailed utilization statistics in *ABSEMO*



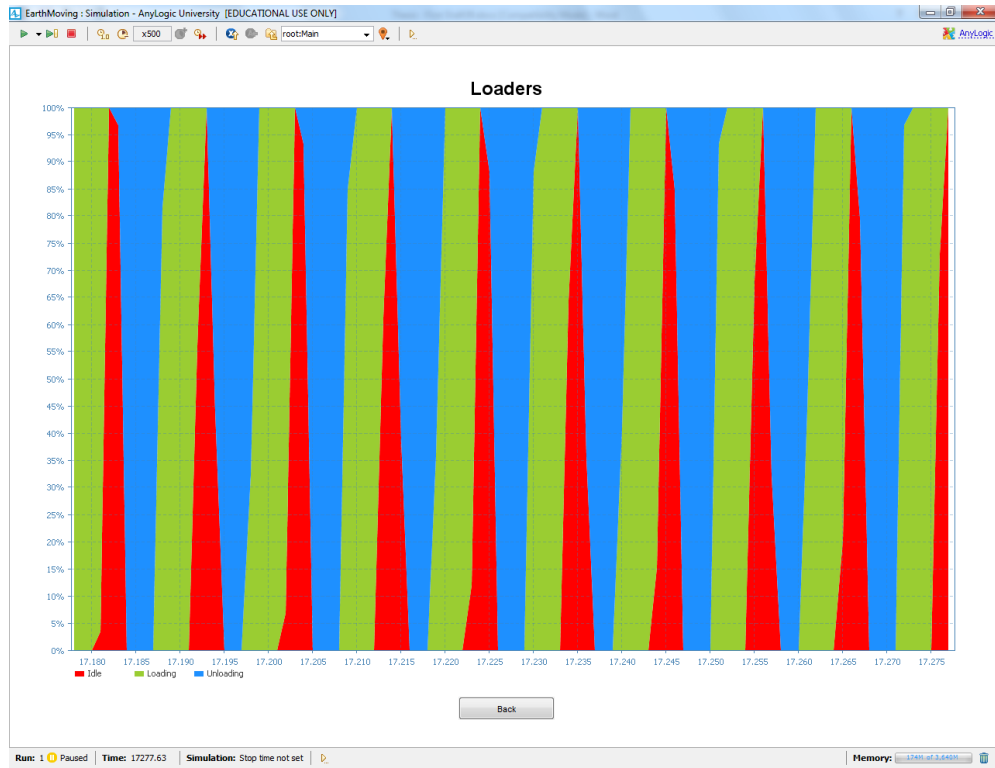


Figure 4-16: Loaders detailed utilization statistics in *ABSEMO*

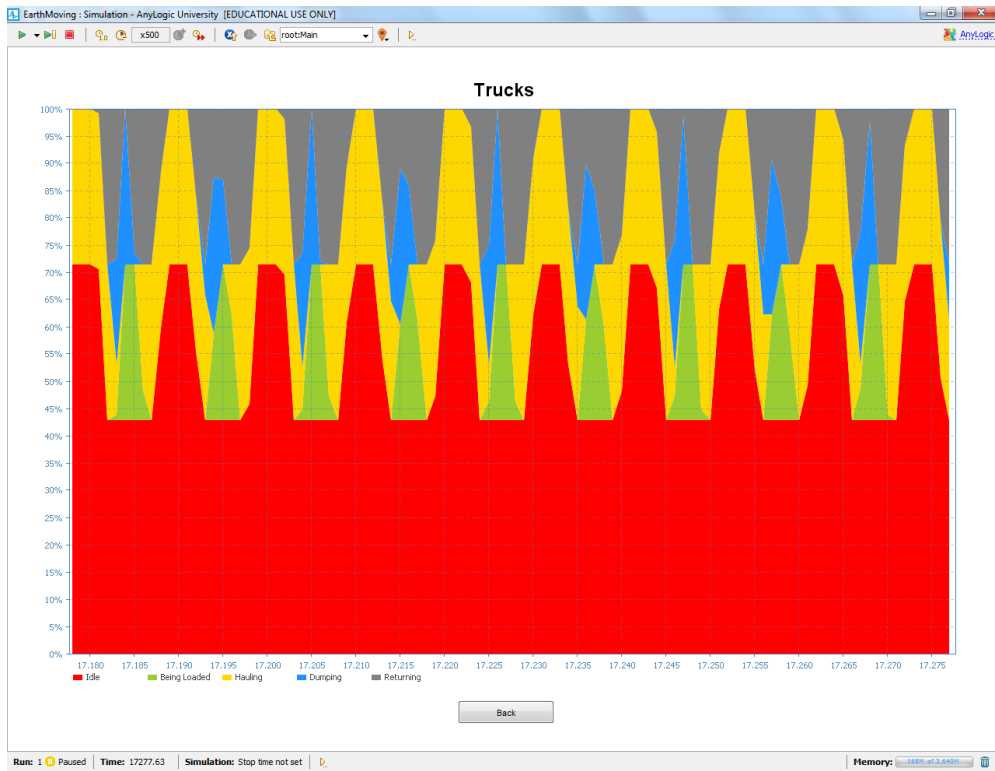
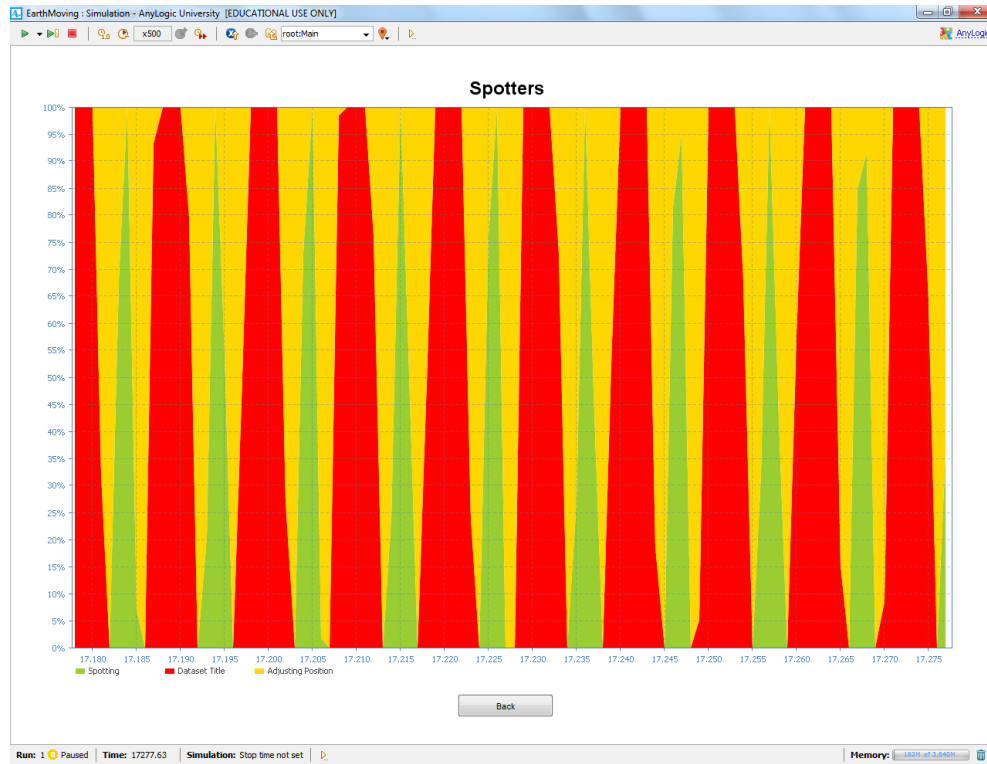


Figure 4-17: Haulers detailed utilization statistics in *ABSEMO*



**Figure 4-18:** Spotters detailed utilization statistics in *ABSEMO*

## 4.6 System Verification

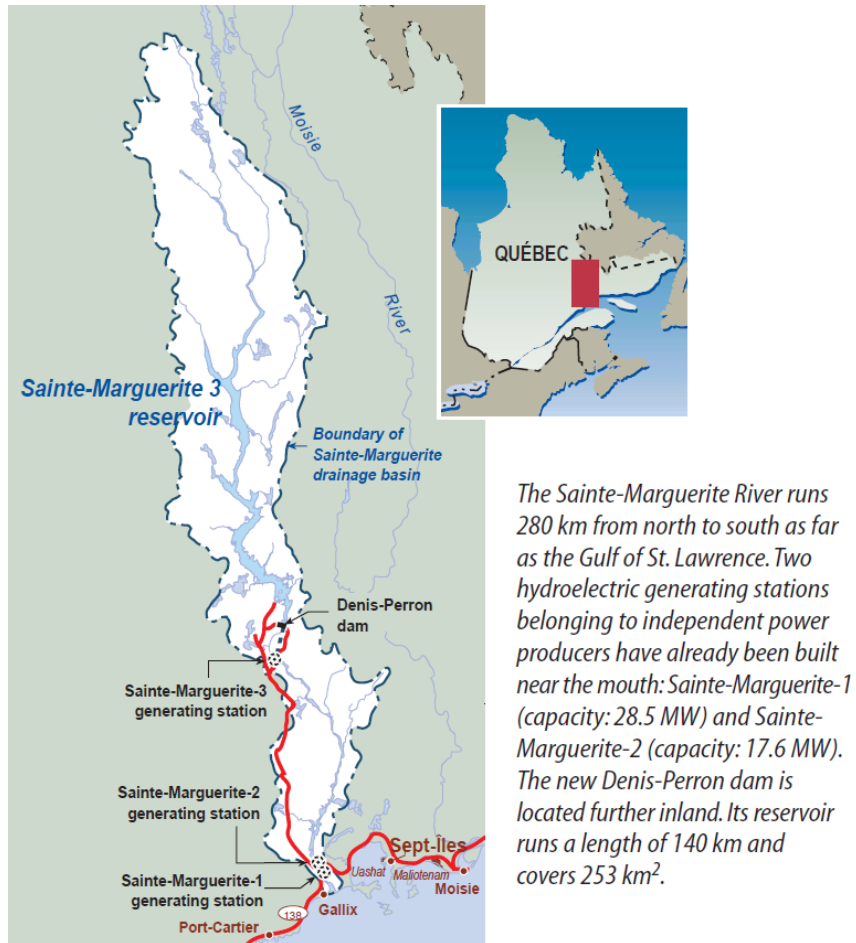
This section is dedicated to test and verify the developed AB simulation model using a real-world case study from the construction sector. The case study is related to earthmoving operations in a dam construction project in the province of Quebec, Canada. It involves two main operations: 1) the excavation of riverbed soil at the location of the dam; 2) the backfill of three types of soils in three stages. The scope of work involved in these two operations is estimated from the structural design of the dam, and the hauling and returning durations of trucks are calculated based on routes' profiles and rolling resistance. As a proof of concept, only the excavation of the riverbed soil will modeled and simulated using *ABSEMO*.

#### 4.6.1 Case Study Description

The case study is concerned with modeling and simulating the riverbed excavation operation involved in the construction of Sainte-Marguerite-3 (SM-3) Dam (1994-2002), which is located on the Sainte-Marguerite River in Sept-Îles City, 700 km northeast of Montréal, Canada. Figure 4-19 shows a map of the river's location and Figure 4-20 shows a picture of the dam after construction. Information about this case study were obtained from Peer (2001), Hydro Quebec (2003), Marzouk (2004) and Alzraiee (2013).

#### 4.6.2 Scope of Work

Earthmoving operations in the SM-3 dam construction project were allotted three years by the management. Regarding the riverbed excavation operation, which will be used to verify *ABSEMO*, the actual quantity of excavated natural soil was 1,038,000 m<sup>3</sup> (Peer 2001). The quantity of excavated soil was not used in the construction of the dam; instead, it was hauled away and dumped in another location. Accordingly, the backfill operation was performed by borrowing 6,300,000 m<sup>3</sup> of soil from three pits. Table 4-10 summarizes the scope of work. The excavation part, which is the interest in this study, is highlighted in bold.



**Figure 4-19:** Location of Sainte-Marguerite-3 dam (Hydro Quebec 2003)



**Figure 4-20:** Sainte-Marguerite-3 dam (Hydro Quebec 2003)

**Table 4-10:** Scope of work in in the SM-3 dam construction earthmoving operations (Alzraiee 2013)

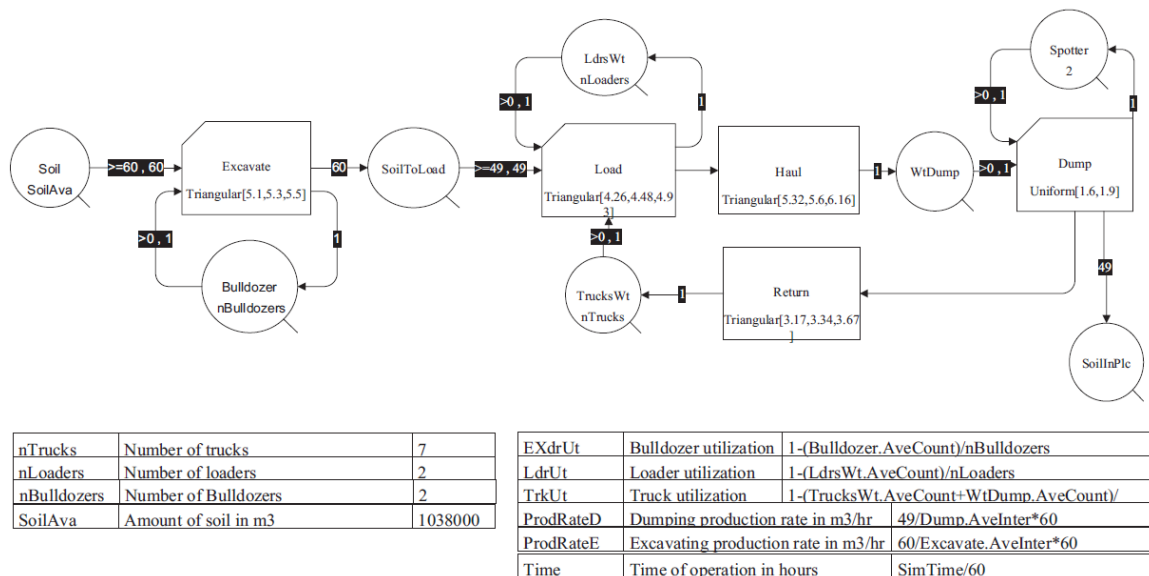
<b>Element</b>	<b>Rock</b>	<b>Granular</b>	<b>Moraine</b>	<b>Total</b>
Backfilling Stage 1 (m <sup>3</sup> )	192,700	14,500	29,200	236,400
Backfilling Stage 2 (m <sup>3</sup> )	3,209,400	286,500	555,900	4,051,800
Backfilling Stage 3 (m <sup>3</sup> )	1,602,900	139,000	269,000	2,011,800
Loose Density (t/m <sup>3</sup> )	1.66	1.72	1.66	1.6
Bank Density (t/m <sup>3</sup> )	2.73	1.93	2.02	2.4
Load Factor	80	90	100	100
Total Quantity of Soil	5,005,000	440,000	855,000	6,300,000
<b>Excavation (m<sup>3</sup>)</b>	<b>1,038,000</b>			<b>1,038,000</b>

#### 4.6.3 Fleet Selection and Configuration

Table 4-11 and Table 4-12 represent the equipment fleet combinations selected for performing the different earthmoving operations involved in the SM-3 dam construction. Triangular probability distributions were considered for the process durations of loading, hauling, returning, spreading and compaction. On the other hand, uniform distributions were considered for the dumping process (Alzraiee 2013). The last row of Table 4-11, which is highlighted in bold, depicts the fleet configuration, material properties and activity durations of the riverbed excavation operation.

#### 4.6.4 DES Simulation Results

Alzraiee (2013) created an EZStrobe DES simulation model for the riverbed excavation of the SM-3 dam construction project, which is portrayed in Figure 4-21. The model consists of two bulldozers, two loaders, seven haulers and two spotters. As demonstrated in Table 4-10, the quantity of soil to be excavated is 1,038,000 m<sup>3</sup>. Also, equipment properties and activity durations are the ones listed in the last row of Table 4-11. The results of the DES model suggested that a duration of 808.33 hours with an average productivity of 1284.52 m<sup>3</sup>/hour is required to complete the operation.



**Figure 4-21:** EZStrobe DES model of riverbed excavation in the Sainte-Marguerite dam construction project (Alzraiee 2013)

**Table 4-11:** Hauler and loader fleet configurations in the SM-3 dam construction earthmoving operations (Alzraiee 2013)

Hauled Material	Hauler Model	Loader Model	Bucket Capacity of Loader	Loose Density of Soil (ton/m <sup>3</sup> )	Hauled Soil Volume (m <sup>3</sup> )	Hauled Soil Weight (ton)	Loading Process-Time Distribution (min)	Hauling Process-Time Distribution (min)	Dumping Process-Time Distribution (min)	Returning Process-Time Distribution (min)
Rock	777D	992G	12.3	1.66	49	81.67	(3.94, 4.15, 4.57)	(4.3, 4.53, 4.98)	(1.9, 2.2)	(3.17, 3.34, 3.67)
Moraine	773D	990 SII	9.2	1.66	28	45.82	(3.01, 3.2, 3.32)	(19.47, 20.5, 22.55)	(1.6, 1.9)	(16.71, 17.59, 19.35)
Granular	769C	988F	6.9	1.72	20	34.36	(2.3, 2.42, 2.5)	(30.6, 32.34, 35.57)	(1.3, 1.5)	(25.85, 26.51, 29.16)
<b>Riverbed Soil</b>	<b>777D</b>	<b>375L</b>	<b>4.59</b>	<b>1.6</b>	<b>32</b>	<b>51.41</b>	<b>(4.26, 4.48, 4.93)</b>	<b>(5.32, 5.6, 6.16)</b>	<b>(1.6, 1.9)</b>	<b>(2.86, 3.01, 3.31)</b>

**Table 4-12:** Spread and compact equipment characteristics in the SM-3 dam construction earthmoving operations (Alzraiee 2013)

Process	Bulldozer Model	Productivity (m <sup>3</sup> /Cycle)	Time Distribution (min)
Spread	D8R	27	(2.47, 2.6, 2.86)
Compact	CS-583C	19	(1.8, 1.9, 2.09)

#### 4.6.5 ABMS Simulation Results

Activity durations presented in Table 4-10 are all of DES nature. The excavation, loading, hauling, dumping and returning activities are all single entities with single durations. Verifying *ABSEMO* using this data requires some manipulation. Consequently, some inputs of the riverbed excavation process used in *ABSEMO* are adjusted based on the following assumptions:

- Setting the excavation duration equal to the ‘push duration’ in *ABSEMO*. Other durations in the bulldozer agent inputs have no value (0 min).
- By dividing the hauler’s capacity over the loader’s bucket capacity, the number of loader cycles to fill one truck is obtained ( $49 \text{ m}^3 / 4.59 \text{ m}^3 = 10.675381$ ). The loading duration is then divided by that number to get the duration of the loader cycle ( $(4.26 \text{ min}, 4.48 \text{ min}, 4.93 \text{ min}) / 10.675381 = (0.399049 \text{ min}, 0.419657 \text{ min}, 0.461810 \text{ min})$ ). Finally, 75% of the duration of the loader cycle ( $0.299287 \text{ min}, 0.314743 \text{ min}, 0.3463575 \text{ min}$ ) is inputted as the ‘bucket load duration’ and 25% ( $0.099762 \text{ min}, 0.104914 \text{ min}, 0.115453 \text{ min}$ ) is inputted as the ‘full bucket unload duration’ in *ABSEMO*. Other durations in the loader agent inputs have no value (0 min).
- Setting the haul, dump and return durations as ‘haul duration’, ‘return duration’ and ‘dump duration’. Other durations in the hauler agent inputs have no value (0 min).
- The ‘position adjustment duration’ input of the spotter agent has no value (0 min)



After inputting the soil quantity to be excavated, the equipment capacities and the adjusted activity durations of the riverbed excavation in *ABSEMO*, a duration of 804.95 hours with an average productivity of 1289.52 m<sup>3</sup>/hour was obtained for the completion of the operation.

#### 4.6.6 Comparison between DES and ABMS Results

DES results represent a good verification tool for AB models. The nature of DES provides an accurate flow of resources, which allows for verifying the quantitative aspects of AB models. There are some examples in literature on the verification of ABMS outputs using DES results (Biswas and Merchawi 2000, Fortino et al. 2005). However, it is important to note that to validate the accuracy and precision of ABMS, the emergent behavior of the AB model should be compared to that of the real-world system.

Table 4-3 summarizes the comparison between the DES and ABMS results for the SM-3 riverbed excavation operation. The percentage difference between the DES and ABMS durations was 0.42%, indicating that the representation of agents' properties, the interaction logic and the flow of resources in *ABSEMO* are correct. Therefore, *ABSEMO* can now be used to plan earthmoving operations with more complex inputs that fit its capabilities and takes advantage of its strength.

**Table 4-13:** A comparison between DES and ABMS Results for the SM-3 Riverbed Excavation

Technique	Duration (hours)	Productivity (m <sup>3</sup> /hour)
DES	808.33	1284.52
ABMS	804.95	1289.52

#### 4.7 Superiority of ABMS

Subsequently to developing and implementing an AB model for earthmoving operations, several advantages of the ABMS technique are noted. Compared to DES and SD, the following points of superiority exist in ABMS:

- ABMS allows for creating combined time progression models (discrete and continuous), which helps capture a realistic flow of resources based on the activities at-hand. In DES and SD, modelers have to follow the time progression mechanism of the technique being used, regardless of the operation being modeled.
- ABMS is a stochastic approach, compared to the deterministic SD technique. ABMS heavily relies on randomness of variables to produce realistic emergent behaviors. In addition, unlike existing DES models, AB models can accept variables from different probability distributions, not only for activity durations, but also for agents' characteristics including quantity-related attributes. Examples from *ABSEMO* include bulldozers' push capacities, loaders' bucket loading quantities, haulers' capacities, etc.
- ABMS offers a huge flexibility in modeling agents' characteristics and roles to match those of the real-world participants of the systems being modeled. In *ABSEMO*, Different equipment attributes and the ability of the loader agent to perform the excavation as well as the loading activities or only the loading activity are examples of flexibility in AB models. DES and SD models are often rigid in nature, offering limited capabilities of representing resources of different characteristics.

- The smart behavior of agents in AB models establishes a strong mechanism for replicating the real-world operations, by adapting and behaving differently at different situations. In other words, changing quantities and activity durations based on agents' capacities and states are examples of adaption in AB models. DES and SD are often inflexible in terms of behavior of model entities, which translates into a predetermined performance of the systems being modeled using these techniques.

#### **4.8 Implementation of ABMS in Other CM Areas**

Although this research work mainly focused on the development of an AB model for earthmoving operations, the applied procedures and methodologies can be easily extended to the modeling of other construction operations. The heterogeneity of construction operations can be captured with great accuracy through the utilization of smart, flexible and comprehensive AB models. Accordingly, simulation knowledge, ABMS understanding and object-oriented programming skills are essential requirements for creating such models.

The generic procedure for creating AB models for construction management applications presented in Section 3.2 acts as a guide for planners who are interested in using this technique to model different construction operations. Generally, construction operations include different participants which are interacting to fulfil a certain goal. Identifying the properties and roles of these participants, understanding their interaction logic and communication mechanism as well as recognizing the environment they exist

and interact within are all key tasks in developing accurate AB models for construction operations.

Differences in AB models for construction management applications arise from the nature of the systems being modeled. Agents in an earthmoving AB model are expected to be the equipment units and labor participating in the different earthmoving activities. Similarly, technical operations in construction including tower crane operations, concrete pouring, formwork installation, and so on will have the same principles in developing agents and assigning their properties and roles. However, managerial aspects of construction management including workers' behavior, claims negotiations, conflict resolution, contract management, etc. involve a psychological human side that cannot be easily captured to produce emergent behaviors. In light of that, systems, in which humans are agents, represent a far bigger challenge to planners when developing AB models. So, although ABMS is a relatively simple technique to apply, the nature of the operations being modeled profoundly affects the ability of modelers to design agents and simulate their behavior.

## CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS

### 5.1 Summary and Conclusions

This research work proposes ABMS as an effective tool for modeling earthmoving operations. Essentially, the core idea is relying on the strength of the AB technique in representing model participants' properties, roles and interactions to enhance the planning of earthmoving operations and overcome limitations of current earthmoving simulation practices including poor graphical modeling support, predetermined and inflexible behaviors as well as the necessity of detailed data when modeling.

In turn, an in-depth literature review was conducted to investigate current simulation techniques and their applications in construction management in general and earthmoving operations in particular. Thus, DES and SD were examined by studying their methodologies, identifying their limitations and inspecting their applications. The ABMS technique was thoroughly examined by studying its major aspects including agents, environments, interactions and emergence and going over its applications in civil engineering and construction management. Gaps and limitations of current research were illustrated to highlight the need of ABMS in planning earthmoving operations.

Moreover, a detailed explanation of the research methodology was provided. A simple step-by-step procedure on how to develop AB models for different construction operations based on a set of rules and criteria was outlined, according to the author's perception of the AB technique. Then, the development of a comprehensive AB model for earthmoving operations consisting of bulldozers, loaders, haulers and spotters was demonstrated. The model governs the process logistics, information sharing, equipment

properties and activity durations. In addition, agents' types, attributes, roles and interaction logic were discussed in detail, and the guidelines for the implementation of the earthmoving AB model were highlighted.

A Java-based object-oriented software application, Agent-Based Simulator for Earthmoving Operations' (*ABSEMO*), was developed as an implementation of the proposed earthmoving AB model. *ABSEMO* was verified through a real-life case study of the riverbed excavation operation in the SM-3 dam construction project. Subsequently, the available data was manipulated to fit model inputs and the simulation results were compared with those obtained from a DES model of the same operation. A percentage difference of 0.42% from the DES results was obtained, verifying that the model's logic and flow of resources are accurate.

Fundamentally, the strength of ABMS is explored in this research work. The developed procedure for building AB models presents valuable guidelines on how to plan construction operations with ABMS. The proposed earthmoving AB model and the implemented system (*ABSEMO*) aim at enhancing the accuracy of current modeling practices of earthmoving operations, by adding flexibility and introducing adaption to the planning of these operations. Creating agents and assigning their attributes and roles from an individual AB prospective allows for capturing a realistic behavior of earthmoving operations. Hence, the proposed methodology can be extended to general applications in construction management, where studying the emergent behavior of participants' interactions can help planners account for the heterogeneity of operations in this field.

## 5.2 Research Contributions

Key contributions of this research are noted in the following points:

- Establish a generic methodology for applying ABMS in construction management based on a bottom-up procedure of nine steps.
- Develop a comprehensive AB model for planning earthmoving operations, which governs' the environment setup, the equipment and labor properties and roles, the interaction logic and the flow of resources.
- Produce a stand-alone software application (*ABSEMO*) as an implementation of the proposed earthmoving AB model, which offers flexibility in agents' roles and allows for modeling equipment units of different properties, in an intelligent manner that adapts to changes in material quantities, agent variables and other model conditions.

## 5.3 Research Limitations

The limitations of the proposed framework can be summarized in the following points:

- Earthmoving process logistics, which are related to handling soil quantities and assigning activity durations in the proposed AB model are mainly based on observations of earthmoving operations and include many assumptions.
- The proposed AB model assumes the availability of haul and return durations of trucks, and does not include a mechanism for measuring those parameters based on roads' characteristics and haulers' properties or for selecting the best haul and return roads from a number of options.

- Direct and indirect costs are not taken into consideration in the proposed earthmoving AB model and hence, *ABSEMO* does not provide users with an optimization tool to select optimum fleet combinations.

#### **5.4 Future Work and Recommendations**

Based on the aforementioned research limitations, the following future work areas and recommendations are presented:

➤ Current research enhancement areas:

- More effort should be put in upgrading agents' state charts to capture a more realistic behavior of handling soil quantities and assigning activity durations. Current assumption are based on linear interpolation. For example, the time required to fill half a bucket is equal to half the time required to fill the whole bucket. In reality, the distribution of the bucket filling duration might not be as linear as this.
- Some options can be added to *ABSEMO* to allow users to customize the handling of soil quantities and the assignment of activity durations based on their preference, instead of having fixed assumptions for these variables. For instance, if the user wishes that the loader does not start filling its bucket unless a quantity of soil equal to or larger than the buck capacity is available, they should be able to specify that prior to the simulation.



➤ Current research extension areas:

- The procedure of creating AB models for construction management applications can be made more specific, with added templates for common construction operations. This can be of great benefit to planners, especially if an automated tool is provided to guide users on the exact procedure of building their AB models and suggest the best modeling tool.
- The provision of a procedure to measure the haul and return durations of trucks and/or to select the best haul and return roads can be a valuable addition to the existing earthmoving AB model. Since the haul and return activities of trucks are lengthy in duration and represent variables that are highly affected by conditions external to the operation, obtaining accurate durations of these activities has always been a major goal in earthmoving simulation. In addition, selecting the best haul and return roads among a number of options is an interesting research area that can benefit from the nature of agents in ABMS, which are smart, independent and have the ability of making the best choices when facing alternatives.
- Accounting for direct and indirect costs in the proposed earthmoving AB model grasps the whole process and allows for adding an optimization tool to *ABSEMO* which can suggest optimum fleet combinations based on equipment availability and other preferences and conditions related to the earthmoving operation at-hand. In view of that, minimizing cost per unit of production, while adhering to the work schedule, is a crucial objective in planning earthmoving operations.

## REFERENCES

- Abdel-Hamid, T., and Madnick, S. E. (1991). *Software project dynamics: an integrated approach*, Prentice-Hall, Englewood Cliffs, NJ.
- AbouRizk, S. M., and Hajjar, D. (1998). "A framework for applying simulation in construction." *Canadian Journal of Civil Engineering*, 25(3), 604-617.
- Ahn, S., Lee, S., and Steel, R. P. (2013). "Effects of workers' social learning: Focusing on absence behavior." *Journal of Construction Engineering and Management*, 139(8), 1015-1025.
- Alzraiee, H. S. (2013). *Hybrid Simulation for Construction Operations*. Ph.D. Dissertation, Building, Civil and Environmental Engineering Department, Concordia University, Montreal, QC, Canada.
- Alzraiee, H., Moselhi, O., and Zayed, T. (2012). "Dynamic planning of earthmoving projects using system dynamics." *Gerontechnology*, 11(2), 316.
- Alzraiee, H., Zayed, T., and Moselhi, O. (2012). "Methodology for synchronizing discrete event simulation and system dynamics models." *Proceedings of the 2012 Winter Simulation Conference*, IEEE, Berlin, Germany, No. 54.
- Banks, J., Carson, J., and Nelson, B. (2000). *Discrete-Event System Simulation*, Prentice-Hall, Englewood Cliffs, NJ.
- Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2004). *Winning Ways for Your Mathematical Plays*, Volume 4, A. K. Peters, Wellesley, MA.
- Biswas, S., & Merchawi, S. (2000). "Use of discrete event simulation to validate an agent based scheduling engine." *Proceedings of the 32nd Conference on Winter Simulation*, SCS, Orlando, FL, USA, 1778-1782.
- Borshchev, A., and Filippov, A. (2004). "From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools." *Proceedings of the 22nd international conference of the system dynamics society*, Oxford, England, No. 22.
- Brailsford, S., and Hilton, N. (2001). "A comparison of discrete event simulation and system dynamics for modelling health care systems." School of Management, University of Southampton, UK.
- Ceylan, B. K., and Ford, D. N. (2002). "Using Options to Manage Dynamic Uncertainty in Acquisition Projects." *Acquisition Review Quarterly*, 9(4), 243-258.

- Chan, W. K. V., Son, Y., and Macal, C. M. (2010). "Agent-based simulation tutorial-simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation." *Proceedings of the 2010 Winter Simulation Conference*, Baltimore, WSC, MD, USA, 135-150.
- Chang, D. Y., & Carr, R. I. (1987). "RESQUE: A resource oriented simulation system for multiple resource constrained processes." *In Proceedings of the PMI Seminar/Symposium*, Milwaukee, WI, USA, 4-19.
- Cooper, K. G. (1980). "Naval ship production: A claim settled and a framework built." *Interfaces*, 10(6), 20-36.
- Dzeng, R., and Lin, Y. (2004). "Intelligent agents for supporting construction procurement negotiation." *Expert Systems with Applications*, 27(1), 107-119.
- El-Adaway, I. H., and Kandil, A. A. (2009). "Multiagent system for construction dispute resolution (MAS-COR)." *Journal of Construction Engineering and Management*, 136(3), 303-315.
- Elwakil, E. (2011). *Knowledge Discovery Based Simulation System in Construction*. Ph.D. Dissertation, Building, Civil and Environmental Engineering Department, Concordia University, Montreal, QC, Canada.
- Ford, D. N. (1995). *The Dynamics of Project Management: An Investigation of the Impacts of Project Process and Coordination on Performance*. Ph.D. Dissertation, Massachusetts Institute of Technology, Sloan School of Management, Cambridge, MA.
- Ford, D. N., and Sterman, J. D. (1998). "Dynamic modeling of product development processes." *System Dynamics Review*, 14(1), 31-68.
- Forrester, J. (1961). *Industrial dynamics*, Productivity Press, Acton, MA.
- Fortino, G., Garro, A., & Russo, W. (2005). "A Discrete-Event Simulation Framework for the Validation of Agent-based and Multi-Agent Systems." *Annual Western Orthopaedic Association Meeting*, Camerino, MC, Italy, 75-84.
- Garcia, R. (2005). "Uses of Agent-Based Modeling in Innovation/New Product Development Research\*." *Journal of Product Innovation Management*, 22(5), 380-398.
- Gilbert, N., and Troitzsch, K. (2005). *Simulation for the social scientist*. McGraw-Hill International, Maidenhead, Berkshire.
- Grimm, V., and Railsback, S. F. (2013). *Individual-based modeling and ecology*. University Press, Princeton, NJ.

- Hajjar, D., and AbouRizk, S. (1999). "Symphony: an environment for building special purpose construction simulation tools." *Proceedings of the 31st Conference on Winter Simulation: Simulation---a Bridge to the Future-Volume 2*, ACM, Phoenix, AZ, USA, 998-1006.
- Hajjar, D., and AbouRizk, S. (1997). "AP2-Earth: a simulation based system for the estimating and planning of earth moving operations." *Proceedings of the 29th Conference on Winter Simulation*, IEEE, Atlanta, GA, USA, 1103-1110.
- Hajjar, D., and AbouRizk, S. M. (2002). "Unified modeling methodology for construction simulation." *Journal of Construction Engineering and Management*, 128(2), 174-185.
- Hajjar, D., and AbouRizk, S. M. (1998). "Modeling and analysis of aggregate production operations." *Journal of Construction Engineering and Management*, 124(5), 390-401.
- Hajjar, D., AbouRizk, S., and Xu, J. (1998). "Construction site dewatering analysis using a special purpose simulation-based framework." *Canadian Journal of Civil Engineering*, 25(5), 819-828.
- Halpin, D. W. (1977). "CYCLONE-method for modeling job site processes." *Journal of the Construction Division*, 103 (ASCE 13234 Proceeding).
- Halpin, D. W., Jen, H., and Kim, J. (2003). "A construction process simulation web service." *Proceedings of the 2003 Winter Simulation Conference*, ACM, New Orleans, LA, USA, 1503-1509.
- Halpin, D. W., and Martinez, L. (1999). "Real world applications of construction process simulation." *Proceedings of the 31st Conference on Winter Simulation: Simulation---a Bridge to the Future-Volume 2*, ACM, Phoenix, AZ, USA. 956-962.
- Halpin, D., and Riggs, L. (1992). *Planning and analysis of construction operations*. John Wiley and Sons, New York, NY.
- Homer, J., Sterman, J., Greenwood, B., and Perkola, M. (1993). "Delivery time reduction in pulp and paper mill construction projects: a dynamic analysis of alternatives." *SYSTEM 93*, Cancun, Mexico. 212-221.
- Huang, R., and Halpin, D. W. (1994). "Visual construction operation simulation: the DISCO approach." *Computer-Aided Civil and Infrastructure Engineering*, 9(3), 175-184.
- Hydro Quebec (2003). "Construction of the Sainte-Marguerite-3 Hydroelectric Development 1994-2002." *Bibliothèque Nationale du Québec*.
- Ioannou, P. (1989). "UM-CYCLONE user's manual." *Division of Construction Engineering and Management, Purdue University*, West Lafayette, IN.

- Lane, D. C. (2000). "You just don't understand me: Modes of failure and success in the discourse between system dynamics and discrete event simulation." *Working Paper, London School of Economics and Political Sciences*, London, UK.
- Lee, S., Han, S., and Peña-Mora, F. (2007). "Hybrid system dynamics and discrete event simulation for construction management." *Proceeding of the 2007 ASCE International Workshop on Computing in Civil Engineering*, TCCIT, Pittsburgh, Pennsylvania. 232-239.
- Lyneis, J. M., Cooper, K. G., and Els, S. A. (2001). "Strategic management of complex projects: a case study using system dynamics." *System Dynamics Review*, 17(3), 237-260.
- Macal, C. M., and North, M. J. (2010). "Tutorial on agent-based modelling and simulation." *Journal of Simulation*, 4(3), 151-162.
- Macal, C. M., and North, M. J. (2008). "Agent-based modeling and simulation: ABMS examples." *Proceedings of the 40th Conference on Winter Simulation*, WSC, Miami, FL, USA. 101-112.
- Martínez, J. C. (1998). "Earthmover-simulation tool for earthwork planning." *Winter Simulation Conference Proceedings*, ACM, Washington DC, USA. 1263-1271.
- Martinez, J. C., and Ioannou, P. G. (1999). "General-purpose systems for effective construction simulation." *Journal of Construction Engineering and Management*, 125(4), 265-276.
- Martinez, J. C. (1996). *STROBOSCOPE: State and resource based simulation of construction processes*. Ph.D. Dissertation, University of Michigan, Ann Arbor, MI, USA.
- Marzouk, M. (2002). *Optimizing Earthmoving Operations using Computer Simulation*, Ph.D. Dissertation, Building, Civil and Environmental Engineering Department, Concordia University, Montreal, QC, Canada.
- Marzouk, M., and Moselhi, O. (2003). "Object-oriented simulation model for earthmoving operations." *Journal of Construction Engineering and Management*, 129(2), 173-181.
- Mccabe, B. Y. (1997). *An automated modeling approach for construction performance improvement using simulation and belief networks*. Ph.D. Dissertation, Civil and Environmental Engineering Department, Alberta University, Edmonton, AB, Canada.
- Meadows, D. H. (1980). "The unavoidable a priori." *Elements of the System Dynamics Method*, 23-57, Productivity Press, Acton, MA.

- Min, J. U., and Bjornsson, H. C. (2008). "Agent-Based Construction Supply Chain Simulator (CS 2) for Measuring the Value of Real-Time Information Sharing in Construction." *Journal of Construction Engineering and Management*, 24(4), 245-254.
- Moselhi, O., and Alshibani, A. (2009). "Optimization of earthmoving operations in heavy civil engineering projects." *Journal of Construction Engineering and Management*, 135(10), 948-954.
- Ndumu, D., Collis, J., Owusu, G., Sullivan, M., and Lee, L. (1999). ZUES: "A Toolkit for Building Distributed Multi-Agent Systems." *AgentLink News* (2). pp. 6-9.
- Niazi, M., and Hussain, A. (2011). "Agent-based computing from multi-agent systems to agent-based models: a visual survey." *Scientometrics*, 89(2), 479-499.
- North, M., and Macal, C. (2006). "Tutorial on Agent-Based Modeling and Simulation Part 2: How to Model with Agents." *Proceedings of the 2006 Winter Simulation Conference*, WSC, Monterey, CA, USA.
- North, M. J., and Macal, C. M. (2007). *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*. Oxford University Press, Oxford, UK.
- Oloufa, A. A. (1993). "Modeling operational activities in object-oriented simulation." *Journal of Construction Engineering and Management*, 7(1), 94-106.
- Osman, H. (2012). "Agent-based simulation of urban infrastructure asset management activities." *Automation in Construction*, 28 45-57.
- Palaniappan, S., Sawhney, A., Janssen, M. A., and Walsh, K. D. (2007). "Modeling construction safety as an agent-based emergent phenomenon." *The 24th International Symposium on Automation and Robotics in Construction*, Indian Institute of Technology, Madras, India.
- Park, M., and Peña-Mora, F. (2003). "Dynamic change management for construction: introducing the change cycle into model-based project management." *System Dynamics Review*, 19(3), 213-242.
- Paulson Jr, B. C., Chan, W. T., and Koo, C. C. (1987). "Construction operations simulation by microcomputer." *Journal of Construction Engineering and Management*, 113(2), 302-314.
- Peer, G. (2001). "Powerhouse takes shape far from dam SM-3 project." *Heavy Construction News*, 14-16.

- Peña-Mora, F., Han, S., Lee, S., and Park, M. (2008). "Strategic-operational construction management: Hybrid system dynamics and discrete event approach." *Journal of Construction Engineering and Management*, 134(9), 701-710.
- Pritsker, A. A. B. (1986). *Introduction to stimulation and Slam II*, John Wiley & Sons, New York, NY.
- Ren, Z., and Anumba, C. J. (2002). "Learning in multi-agent systems: a case study of construction claims negotiation." *Advanced Engineering Informatics*, 16(4), 265-275.
- Reynolds, C. W. (1999). "Steering behaviors for autonomous characters." *Game developers conference*, San Jose, CA, USA, 763-782.
- Richardson, G. P., and Pugh III, A. I. (1981). *Introduction to system dynamics modeling with DYNAMO*, Productivity Press, Acton, MA.
- Sanford Bernhardt, K., and McNeil, S. (2008). "Agent-based modeling: approach for improving infrastructure management." *Journal of Infrastructure Systems*, 14(3), 253-261.
- Sawhney, A., and AbouRizk, S. M. (1995). "HSM-simulation-based planning method for construction projects." *Journal of Construction Engineering and Management*, 121(3), 297-303.
- Schelling, T. C. (1971). "Dynamic models of segregation†." *Journal of Mathematical Sociology*, 1(2), 143-186.
- Shannon, R. E. (1992). "Introduction to simulation." *Proceedings of the 24th Conference on Winter Simulation*, ACM, Arlington, VA, USA, 65-73.
- Shi, J., and AbouRizk, S. M. (1997). "Resource-based modeling for construction simulation." *Journal of Construction Engineering and Management*, 123(1), 26-33.
- Shi, J., and AbouRizk, S. S. (1998). "An automated modeling system for simulating earthmoving operations." *Computer-Aided Civil and Infrastructure Engineering*, 13(2), 121-130.
- Sterman, J. (2000). *Business dynamics: Systems thinking and modeling for a complex world*, McGraw-Hill Irwin, Boston, MA.
- Tah, J. H. (2005). "Towards an agent-based construction supply network modelling and simulation platform." *Automation in Construction*, 14(3), 353-359.
- Tommelein, I., Carr, R., and Odeh, A. (1994). "Assembly of simulation networks using designs, plans, and methods." *Journal of Construction Engineering and Management*, 120(4), 796-815.

Touran, A. (1990). "Integration of simulation with expert systems." *Journal of Construction Engineering and Management*, 116(3), 480-493.

Wartha, C., Peev, M., Borshchev, A., and Filippov, A. (2002). "Manufacturing supply chain applications 1: decision support tool-supply chain." *Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers*, WSC, San Diego, CA, USA, 1297-1301.

Wolstenholme, E. F. (1990). *System enquiry: a system dynamics approach*, John Wiley & Sons, New York, NY.





Wooldridge, M., and Jennings, N. R. (1995). "Intelligent agents: Theory and practice." *The Knowledge Engineering Review*, 10(02), 115-152.








## APPENDIX A









### Main Class Java Code:

**Table A-1:** Agent populations' elements in *ABSEMO* (Java code)





Element/s	Java Code
Startup code	<pre>stopAtTimeEnabled(); bulldozersTypeConfiguration(); loadersTypeConfiguration(); haulersTypeConfiguration(); spottersTypeConfiguration(); startUpLocations();</pre>
 bulldozers [..]  bulldozersTypeConfiguration	<pre>//Adds the data based on the user's input for ( int i = 0; i &lt; numberOfBulldozers; i++){ add_bulldozers(timeToExcavate, bulldozerCapacity, timeToTurn, timeToReturn, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfBulldozers2; i++){ add_bulldozers(timeToExcavate2, bulldozerCapacity2, timeToTurn2, timeToReturn2, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfBulldozers3; i++){ add_bulldozers(timeToExcavate3, bulldozerCapacity3, timeToTurn3, timeToReturn3, 0, 0, 0, 0); }</pre>
 loaders [..]  loadersTypeConfiguration	<pre>//Adds the data based on the user's input for ( int i = 0; i &lt; numberOfLoaders; i++){ add_loaders(timeToLoadFullBucket, bucketCapacity, timeToAdjustWhileEmpty, timeToUnloadFullBucket, timeToAdjustWhileFull, 0, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfLoaders2; i++){ add_loaders(timeToLoadFullBucket2, bucketCapacity2, timeToAdjustWhileEmpty2, timeToUnloadFullBucket2, timeToAdjustWhileFull2, 0, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfLoaders3; i++){ add_loaders(timeToLoadFullBucket3, bucketCapacity3, timeToAdjustWhileEmpty3,</pre>













Element/s	Java Code
	<pre>timeToUnloadFullBucket3, timeToAdjustWhileFull3, 0, 0, 0, 0, 0); }</pre>
<ul style="list-style-type: none"> <li> haulers [..]</li> <li> haulersTypeConfiguration</li> </ul>	<pre>//Adds the data based on the user's input for ( int i = 0; i &lt; numberOfHaulers; i++){ add_haulers(movingToLoadPositionTime, capacity, returningTime, movingToDumpPositionTime, haulingTime, 0, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfHaulers2; i++){ add_haulers(movingToLoadPositionTime2, capacity2, returningTime2, movingToDumpPositionTime2, haulingTime2, 0, 0, 0, 0, 0); } for ( int i = 0; i &lt; numberOfHaulers3; i++){ add_haulers(movingToLoadPositionTime3, capacity3, returningTime3, movingToDumpPositionTime3, haulingTime3, 0, 0, 0, 0, 0); }</pre>
<ul style="list-style-type: none"> <li> spotters [..]</li> <li> spottersTypeConfiguration</li> </ul>	<pre>//Adds the data based on the user's input for ( int i = 0; i &lt; numberOfSpotters; i++){ add_spotters(timeToAdjustPosition, dumpingTime, 0, 0); }</pre>
<ul style="list-style-type: none"> <li> startUpLocations</li> </ul>	<pre>//Bulldozers Locations double a= bulldozerPointer.getX(); double b= bulldozerPointer.getY(); for(int i=0;i&lt;bulldozers.size();i++){ bulldozers.get(i).setXY(a,b); b = b-20; } //Loaders Locations double c= loaderPointer.getX(); double d= loaderPointer.getY(); for(int i=0;i&lt;loaders.size();i++){ loaders.get(i).setXY(c,d); c = c+50; } //Spotters Locations double e= spotterPointer.getX(); double f= spotterPointer.getY(); for(int i=0;i&lt;spotters.size();i++){ spotters.get(i).setXY(e,f); f = f-50; }</pre>

**Table A-2:** Agent queues' elements in *ABSEMO* (Java code)




Element/s	Java Code
 haulerRequestsLoader  haulerQueueAtLoaders	<pre>//add service request to the queue haulerQueueAtLoaders.addLast( truck ); //and make all service crews check the request queue for( Loader ld : loaders ) ld.receive( "CHECK QUEUE" );</pre>
 haulerRequestsSpotter  haulerQueueAtSpotters	<pre>//add service request to the queue haulerQueueAtSpotters.addLast( truck ); //and make all service crews check the request queue for( Spotter sp : spotters ) sp.receive( "CHECK QUEUE" );</pre>
 thereAreRequestsLoader	<pre>return ! haulerQueueAtLoaders.isEmpty();</pre>
 thereAreRequestsSpotter	<pre>return ! haulerQueueAtSpotters.isEmpty();</pre>
 getRequestLoader	<pre>if( ! haulerQueueAtLoaders.isEmpty()) return haulerQueueAtLoaders.removeFirst( ); else return null;</pre>
 getRequestSpotter	<pre>if( ! haulerQueueAtSpotters.isEmpty()) return haulerQueueAtSpotters.removeFirst( ); else return null;</pre>

**Table A-3:** Model run control in *ABSEMO* (Java code)

Element/s	Java Code
 pauseAndResumeFunction  pauseAndResume	<pre>if (button16.getText() == "Pause"    button13.getText() == "Pause"    button10.getText() == "Pause"){ pauseSimulation(); pauseAndResume = "Resume"; } if (button16.getText() == "Resume"    button13.getText() == "Resume"    button10.getText() == "Resume"){ runSimulation(); pauseAndResume = "Pause"; }</pre>
 applyWholeQuantityDumpedOption  stopWhenWholeQuantityDumped	<pre>Condition: dumpedSoil &gt;= dumpedSoilToStopRun &amp;&amp; applyDumpedSoilOption finishSimulation();</pre>

Element/s	Java Code
 applyDumpedSoilOption  dumpedSoilToStopRun  stopAtDumpedQuantity	Condition: dumpedSoil >= dumpedSoilToStopRun && applyDumpedSoilOption finishSimulation();
 applyExcavatedSoilOption  excavatedSoilToStopRun  stopAtExcavatedQuantity	Condition: soilAvailableForLoading >= excavatedSoilToStopRun && applyExcavatedSoilOption finishSimulation();
 applyStopAtTimeOption  stopAtTimeEnabled  hoursToStopRun  minutesToStopRun  secondsToStopRun  stopAtTime	<pre>if (applyStopAtTimeOption){ stopAtTime.restart(); } Timeout: (hoursToStopRun * 60) + (minutesToStopRun) + (secondsToStopRun/60) finishSimulation();</pre>

**Table A-4:** Results and Analysis in *ABSEMO* (Java code)

Element/s	Java Code
 productivity  updateProductivity	Cyclic: every 1 minute productivity = dumpedSoil / time();
 updateStatistics	<pre>//record results bulldozersIdleSet.add( bulldozersIdle.mean() ); bulldozersWorkingSet.add( bulldozersWorking.mean() ); bulldozersExcavatingSet.add( bulldozersExcavating.mean() ); bulldozersRepositioningSet.add( bulldozersRepositioning.mean() ); loadersIdleSet.add( loadersIdle.mean() ); loadersWorkingSet.add( loadersWorking.mean() ); loadersLoadingSet.add( loadersLoading.mean() ); loadersUnloadingSet.add( loadersUnloading.mean() ); haulersIdleSet.add( haulersIdle.mean() ); haulersWorkingSet.add( haulersWorking.mean() ); haulersBeingLoadedSet.add( haulersBeingLoaded.mean() );</pre>

Element/s	Java Code
	<pre> haulersHaulingSet.add( haulersHauling.mean() ); haulersDumpingSet.add( haulersDumping.mean() ); haulersReturningSet.add( haulersReturning.mean() ); spottersIdleSet.add( spottersIdle.mean() ); spottersWorkingSet.add( spottersWorking.mean() ); spottersSpottingSet.add( spottersSpotting.mean() ); spottersAdjustingSet.add( spottersAdjusting.mean() ); //rest the results each time interval bulldozersIdle.reset(); bulldozersWorking.reset(); bulldozersExcavating.reset(); bulldozersRepositioning.reset(); loadersIdle.reset(); loadersWorking.reset(); loadersLoading.reset(); loadersUnloading.reset(); haulersIdle.reset(); haulersWorking.reset(); haulersBeingLoaded.reset(); haulersHauling.reset(); haulersDumping.reset(); haulersReturning.reset(); spottersIdle.reset(); spottersWorking.reset(); spottersSpotting.reset(); spottersAdjusting.reset(); </pre>


## APPENDIX B

### Agent Class Java Code:

#### The Bulldozer Agent:

Table A-5: Bulldozer class elements and transitions in *ABSEM* (Java code)




Element/s	Java Code
 colorB	Changes the color of the bulldozer agent based on its state
 bulldozerCapacityUser  bulldozerCapacityF  bulldozerCapacity	<pre> boolean isTriangular = bulldozerCapacityUser.startsWith("T"); boolean isUniform = bulldozerCapacityUser.startsWith("U"); boolean isNormal = bulldozerCapacityUser.startsWith("N"); if(isTriangular){ int parLocation1 = bulldozerCapacityUser.indexOf('(',0); int parLocation2 = bulldozerCapacityUser.indexOf(')',parLocation1); int commaLocation1 = bulldozerCapacityUser.indexOf(',',0); int commaLocation2 = bulldozerCapacityUser.indexOf(',',commaLocation1+1 ); String tempvalue1 = bulldozerCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bulldozerCapacityUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = bulldozerCapacityUser.substring(commaLocation2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); bulldozerCapacity = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = bulldozerCapacityUser.indexOf('(',0); int parLocation2 = bulldozerCapacityUser.indexOf(')',parLocation1); int commaLocation1 = bulldozerCapacityUser.indexOf(',',0); String tempvalue1 =           </pre>






Element/s	Java Code
	<pre> bulldozerCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bulldozerCapacityUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); bulldozerCapacity = uniform(value1,value2); } else if(isNormal){ int parLocation1 = bulldozerCapacityUser.indexOf('(',0); int parLocation2 = bulldozerCapacityUser.indexOf(')',parLocation1); int commaLocation1 = bulldozerCapacityUser.indexOf(',',0); String tempvalue1 = bulldozerCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bulldozerCapacityUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); bulldozerCapacity = normal(value1,value2); } else{ bulldozerCapacity = Double.parseDouble(bulldozerCapacityUser); } </pre>
<ul style="list-style-type: none"> <li> timeToExcavateUser</li> <li> timeToExcavateF</li> <li> timeToExcavate</li> </ul>	<pre> boolean isTriangular = timeToExcavateUser.startsWith("T"); boolean isUniform = timeToExcavateUser.startsWith("U"); boolean isNormal = timeToExcavateUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToExcavateUser.indexOf('(',0); int parLocation2 = timeToExcavateUser.indexOf(')',parLocation1); int commaLocation1 = timeToExcavateUser.indexOf(',',0); int commaLocation2 = timeToExcavateUser.indexOf(',',commaLocation1+1); String tempvalue1 = timeToExcavateUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToExcavateUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = timeToExcavateUser.substring(commaLocation2+1,parL </pre>

Element/s	Java Code
	<pre> ocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToExcavate = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToExcavateUser.indexOf('(',0); int parLocation2 = timeToExcavateUser.indexOf(')',parLocation1); int commaLocation1 = timeToExcavateUser.indexOf(',',0); String tempvalue1 = timeToExcavateUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToExcavateUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToExcavate = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToExcavateUser.indexOf('(',0); int parLocation2 = timeToExcavateUser.indexOf(')',parLocation1); int commaLocation1 = timeToExcavateUser.indexOf(',',0); String tempvalue1 = timeToExcavateUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToExcavateUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToExcavate = normal(value1,value2); } else{ timeToExcavate = Double.parseDouble(timeToExcavateUser); } </pre>
<ul style="list-style-type: none"> <li> timeToTurnUser</li> <li> timeToTurnF</li> <li> timeToTurn</li> </ul>	<pre> boolean isTriangular = timeToTurnUser.startsWith("T"); boolean isUniform = timeToTurnUser.startsWith("U"); boolean isNormal = timeToTurnUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToTurnUser.indexOf('(',0); int parLocation2 = </pre>



Element/s	Java Code
	<pre> timeToTurnUser.indexOf(',')',parLocation1); int commaLocation1 = timeToTurnUser.indexOf(',')',0); int commaLocation2 = timeToTurnUser.indexOf(',')',commaLocation1+1); String tempvalue1 = timeToTurnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToTurnUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = timeToTurnUser.substring(commaLocation2+1,parLocat ion2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToTurn = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToTurnUser.indexOf('('',0); int parLocation2 = timeToTurnUser.indexOf(')',parLocation1); int commaLocation1 = timeToTurnUser.indexOf(',')',0); String tempvalue1 = timeToTurnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToTurnUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToTurn = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToTurnUser.indexOf('('',0); int parLocation2 = timeToTurnUser.indexOf(')',parLocation1); int commaLocation1 = timeToTurnUser.indexOf(',')',0); String tempvalue1 = timeToTurnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToTurnUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToTurn = normal(value1,value2); } else{ timeToTurn = Double.parseDouble(timeToTurnUser); </pre>

Element/s	Java Code
	}
<ul style="list-style-type: none"> <li> timeToReturnUser</li> <li> timeToReturnF</li> <li> timeToReturn</li> </ul>	<pre> boolean isTriangular = timeToReturnUser.startsWith("T"); boolean isUniform = timeToReturnUser.startsWith("U"); boolean isNormal = timeToReturnUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToReturnUser.indexOf('(',0); int parLocation2 = timeToReturnUser.indexOf(')',parLocation1); int commaLocation1 = timeToReturnUser.indexOf(',',0); int commaLocation2 = timeToReturnUser.indexOf(',',commaLocation1+1); String tempvalue1 = timeToReturnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToReturnUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = timeToReturnUser.substring(commaLocation2+1,parLoc ation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToReturn = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToReturnUser.indexOf('(',0); int parLocation2 = timeToReturnUser.indexOf(')',parLocation1); int commaLocation1 = timeToReturnUser.indexOf(',',0); String tempvalue1 = timeToReturnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToReturnUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToReturn = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToReturnUser.indexOf('(',0); </pre>

Element/s	Java Code
	<pre> int parLocation2 = timeToReturnUser.indexOf(')',parLocation1); int commaLocation1 = timeToReturnUser.indexOf(',',0); String tempvalue1 = timeToReturnUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToReturnUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToReturn = normal(value1,value2); } else{ timeToReturn = Double.parseDouble(timeToReturnUser); } </pre>
<ul style="list-style-type: none"> <li> excavationType</li> <li> excavationQuantity</li> <li> excavationDuration</li> </ul>	<pre> if(get_Main().soilAvailableForExcavation &gt; bulldozerCapacity){ return bulldozerCapacity; } else{ return get_Main().soilAvailableForExcavation; } </pre>
<ul style="list-style-type: none"> <li> indexNumber</li> </ul> <p>startExcavating transition</p>	<pre> Condition: get_Main().soilAvailableForExcavation &gt; 0 excavationQuantity = excavationType(); excavationDuration = timeToExcavate * (excavationType() / bulldozerCapacity); get_Main().soilAvailableForExcavation = get_Main().soilAvailableForExcavation - excavationQuantity; moveToInTime(get_Main().bulldozerPointer2.getX(), get_Main().bulldozerPointer.getY() - (indexNumber * 20), timeToExcavate); </pre>
<ul style="list-style-type: none"> <li> indexNumber</li> </ul> <p>startTurningA transition</p>	<pre> moveToInTime(get_Main().bulldozerPointer3.getX(), get_Main().bulldozerPointer3.getY() - (indexNumber * 20), timeToTurn); get_Main().soilAvailableForLoading = get_Main().soilAvailableForLoading + excavationQuantity; </pre>
<ul style="list-style-type: none"> <li> indexNumber</li> </ul> <p>startReturning transition</p>	<pre> moveToInTime(get_Main().bulldozerPointer4.getX(), get_Main().bulldozerPointer4.getY() - (indexNumber * 20), timeToReturn); </pre>
<ul style="list-style-type: none"> <li> indexNumber</li> </ul> <p>startTurningB transition</p>	<pre> moveToInTime(get_Main().bulldozerPointer.getX(), get_Main().bulldozerPointer.getY() - (indexNumber * 20), timeToTurn); </pre>

## The Loader Agent:

Table A-6: Loader class elements and transitions in *ABSEMO* (Java code)

Element/s	Java Code
<ul style="list-style-type: none"> <li> bucketCapacityUser</li> <li> bucketCapacityF</li> <li> bucketCapacity</li> </ul>	<pre> boolean isTriangular = bucketCapacityUser.startsWith("T"); boolean isUniform = bucketCapacityUser.startsWith("U"); boolean isNormal = bucketCapacityUser.startsWith("N"); if(isTriangular){ int parLocation1 = bucketCapacityUser.indexOf('(',0); int parLocation2 = bucketCapacityUser.indexOf(')',parLocation1) ; int commaLocation1 = bucketCapacityUser.indexOf(',',0); int commaLocation2 = bucketCapacityUser.indexOf(',',commaLocation 1+1); String tempvalue1 = bucketCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bucketCapacityUser.substring(commaLocation1+ 1, commaLocation2); String tempvalue3 = bucketCapacityUser.substring(commaLocation2+ 1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); bucketCapacity = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = bucketCapacityUser.indexOf('(',0); int parLocation2 = bucketCapacityUser.indexOf(')',parLocation1) ; int commaLocation1 = bucketCapacityUser.indexOf(',',0); String tempvalue1 = bucketCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bucketCapacityUser.substring(commaLocation1+ </pre>

Element/s	Java Code
	<pre> 1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); bucketCapacity = uniform(value1,value2); } else if(isNormal){ int parLocation1 = bucketCapacityUser.indexOf('(',0); int parLocation2 = bucketCapacityUser.indexOf(')',parLocation1) ; int commaLocation1 = bucketCapacityUser.indexOf(',',0); String tempvalue1 = bucketCapacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = bucketCapacityUser.substring(commaLocation1+ 1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); bucketCapacity = normal(value1,value2); } else{ bucketCapacity = Double.parseDouble(bucketCapacityUser); } </pre>
<ul style="list-style-type: none"> <li> timeToLoadFullBucketUser</li> <li> timeToLoadFullBucketF</li> <li> timeToLoadFullBucket</li> </ul>	<pre> boolean isTriangular = timeToLoadFullBucketUser.startsWith("T"); boolean isUniform = timeToLoadFullBucketUser.startsWith("U"); boolean isNormal = timeToLoadFullBucketUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToLoadFullBucketUser.indexOf('(',0); int parLocation2 = timeToLoadFullBucketUser.indexOf(')',parLoca tion1); int commaLocation1 = timeToLoadFullBucketUser.indexOf(',',0); int commaLocation2 = timeToLoadFullBucketUser.indexOf(',',commaLo cation1+1); String tempvalue1 = timeToLoadFullBucketUser.substring(parLocati on1+1, commaLocation1); String tempvalue2 = timeToLoadFullBucketUser.substring(commaLoca </pre>

Element/s	Java Code
	<pre> tion1+1, commaLocation2); String tempvalue3 = timeToLoadFullBucketUser.substring(commaLoca tion2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToLoadFullBucket = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToLoadFullBucketUser.indexOf('(',0); int parLocation2 = timeToLoadFullBucketUser.indexOf(')',parLoca tion1); int commaLocation1 = timeToLoadFullBucketUser.indexOf(', ',0); String tempvalue1 = timeToLoadFullBucketUser.substring(parLocati on1+1, commaLocation1); String tempvalue2 = timeToLoadFullBucketUser.substring(commaLoca tion1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToLoadFullBucket = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToLoadFullBucketUser.indexOf('(',0); int parLocation2 = timeToLoadFullBucketUser.indexOf(')',parLoca tion1); int commaLocation1 = timeToLoadFullBucketUser.indexOf(', ',0); String tempvalue1 = timeToLoadFullBucketUser.substring(parLocati on1+1, commaLocation1); String tempvalue2 = timeToLoadFullBucketUser.substring(commaLoca tion1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToLoadFullBucket = </pre>

Element/s	Java Code
	<pre> normal(value1,value2); } else{ timeToLoadFullBucket = Double.parseDouble(timeToLoadFullBucketUser) ; } </pre>
<ul style="list-style-type: none"> <li> timeToAdjustWhileFullUser</li> <li> timeToAdjustWhileFullF</li> <li> timeToAdjustWhileFull</li> </ul>	<pre> boolean isTriangular = timeToAdjustWhileFullUser.startsWith("T"); boolean isUniform = timeToAdjustWhileFullUser.startsWith("U"); boolean isNormal = timeToAdjustWhileFullUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToAdjustWhileFullUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileFullUser.indexOf(')',parLocation1);  int commaLocation1 = timeToAdjustWhileFullUser.indexOf(',',0); int commaLocation2 = timeToAdjustWhileFullUser.indexOf(',',commaLocation1+1); String tempvalue1 = timeToAdjustWhileFullUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileFullUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = timeToAdjustWhileFullUser.substring(commaLocation2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToAdjustWhileFull = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToAdjustWhileFullUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileFullUser.indexOf(')',parLocation1); int commaLocation1 = bucketCapacityUser.indexOf(',',0); String tempvalue1 = timeToAdjustWhileFullUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileFullUser.substring(commaLocation1+1, parLocation2); String tempvalue3 = timeToAdjustWhileFullUser.substring(parLocation2+1, bucketCapacityUser.length()); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToAdjustWhileFull = uniform(value1,value2,value3); } } </pre>

Element/s	Java Code
	<pre> ion1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileFullUser.substring(commaLoc ation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToAdjustWhileFull = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToAdjustWhileFullUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileFullUser.indexOf(')',parLoc ation1); int commaLocation1 = timeToAdjustWhileFullUser.indexOf(',',0); String tempvalue1 = timeToAdjustWhileFullUser.substring(parLocat ion1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileFullUser.substring(commaLoc ation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToAdjustWhileFull = normal(value1,value2); } else{ timeToAdjustWhileFull = Double.parseDouble(timeToAdjustWhileFullUser ); } </pre>
<ul style="list-style-type: none"> <li> timeToUnloadFullBucketUser</li> <li> timeToUnloadFullBucketF</li> <li> timeToUnloadFullBucket</li> </ul>	<pre> boolean isTriangular = timeToUnloadFullBucketUser.startsWith("T"); boolean isUniform = timeToUnloadFullBucketUser.startsWith("U"); boolean isNormal = timeToUnloadFullBucketUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToUnloadFullBucketUser.indexOf('(',0); int parLocation2 = timeToUnloadFullBucketUser.indexOf(')',parLo cation1); int commaLocation1 = timeToUnloadFullBucketUser.indexOf(',',0); int commaLocation2 = timeToUnloadFullBucketUser.indexOf(',',comma </pre>



Element/s	Java Code
	<pre> Location1+1); String tempvalue1 = timeToUnloadFullBucketUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToUnloadFullBucketUser.substring(commaLo cation1+1, commaLocation2); String tempvalue3 = timeToUnloadFullBucketUser.substring(commaLo cation2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToUnloadFullBucket = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToUnloadFullBucketUser.indexOf('(',0); int parLocation2 = timeToUnloadFullBucketUser.indexOf(')',parLo cation1); int commaLocation1 = bucketCapacityUser.indexOf(', ',0); String tempvalue1 = timeToUnloadFullBucketUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToUnloadFullBucketUser.substring(commaLo cation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToUnloadFullBucket = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToUnloadFullBucketUser.indexOf('(',0); int parLocation2 = timeToUnloadFullBucketUser.indexOf(')',parLo cation1); int commaLocation1 = timeToUnloadFullBucketUser.indexOf(', ',0); String tempvalue1 = timeToUnloadFullBucketUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToUnloadFullBucketUser.substring(commaLo </pre>

Element/s	Java Code
	<pre> cation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToUnloadFullBucket = normal(value1,value2); } else{ timeToUnloadFullBucket = Double.parseDouble(timeToUnloadFullBucketUse r); } </pre>
<ul style="list-style-type: none"> <li> timeToAdjustWhileEmptyUser</li> <li> timeToAdjustWhileEmptyF</li> <li> timeToAdjustWhileEmpty</li> </ul>	<pre> boolean isTriangular = timeToAdjustWhileEmptyUser.startsWith("T"); boolean isUniform = timeToAdjustWhileEmptyUser.startsWith("U"); boolean isNormal = timeToAdjustWhileEmptyUser.startsWith("N"); if(isTriangular){ int parLocation1 = timeToAdjustWhileEmptyUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileEmptyUser.indexOf(')',parLo cation1); int commaLocation1 = timeToAdjustWhileEmptyUser.indexOf(',',0); int commaLocation2 = timeToAdjustWhileEmptyUser.indexOf(',',comma Location1+1); String tempvalue1 = timeToAdjustWhileEmptyUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileEmptyUser.substring(commaLo cation1+1, commaLocation2); String tempvalue3 = timeToAdjustWhileEmptyUser.substring(commaLo cation2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); timeToAdjustWhileEmpty = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = timeToAdjustWhileEmptyUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileEmptyUser.indexOf(')',parLo </pre>

Element/s	Java Code
	<pre> cation1); int commaLocation1 = bucketCapacityUser.indexOf(',',0); String tempvalue1 = timeToAdjustWhileEmptyUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileEmptyUser.substring(commaLo cation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToAdjustWhileEmpty = uniform(value1,value2); } else if(isNormal){ int parLocation1 = timeToAdjustWhileEmptyUser.indexOf('(',0); int parLocation2 = timeToAdjustWhileEmptyUser.indexOf(')',parLo cation1); int commaLocation1 = timeToAdjustWhileEmptyUser.indexOf(',',0); String tempvalue1 = timeToAdjustWhileEmptyUser.substring(parLoca tion1+1, commaLocation1); String tempvalue2 = timeToAdjustWhileEmptyUser.substring(commaLo cation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); timeToAdjustWhileEmpty = normal(value1,value2); } else{ timeToAdjustWhileEmpty = Double.parseDouble(timeToAdjustWhileEmptyUse r); } </pre>
<ul style="list-style-type: none"> <li><span style="color: blue;">F</span> loadingType</li> <li><span style="color: orange;">V</span> carriedEarth</li> <li><span style="color: orange;">V</span> timeToLoadBucket</li> </ul>	<pre> if(get_Main().soilAvailableForLoading &gt; bucketCapacity){ return bucketCapacity; } else { return get_Main().soilAvailableForLoading; } </pre>
<ul style="list-style-type: none"> <li><span style="color: blue;">F</span> unloadingType</li> <li><span style="color: orange;">V</span> unloadingQuantity</li> <li><span style="color: orange;">V</span> timeToUnloadBucket</li> </ul>	<pre> if (carriedEarth &lt; Hauler.availableSpace){ return carriedEarth; }else{ get_Main().soilAvailableForLoading = </pre>

Element/s	Java Code
	<pre> get_Main().soilAvailableForLoading + carriedEarth - Hauler.availableSpace; return Hauler.availableSpace; } </pre>
soilAvailable transition	<pre> Condition: get_Main().soilAvailableForLoading &gt; 0 carriedEarth = loadingType(); get_Main().soilAvailableForLoading = get_Main().soilAvailableForLoading - carriedEarth; timeToLoadBucket = timeToLoadFullBucket *(carriedEarth/bucketCapacity); </pre>
loadReadyA transition	<pre> Timeout: timeToLoadBucket + timeToAdjustWhileFull setRotation(3.14159265359); </pre>
haulerAvailable transition	<pre> Condition: get_Main().thereAreRequestsLoader() Hauler = get_Main().getRequestLoader(); send( this, Hauler ); </pre>
startUnloading transition	<pre> Message: "BEGIN UNLOADING" unloadingQuantity = unloadingType(); timeToUnloadBucket = timeToUnloadFullBucket *(unloadingQuantity/bucketCapacity); </pre>
finishUnloading transition	<pre> Timeout: timeToUnloadBucket Hauler.carriedEarth = Hauler.carriedEarth + unloadingQuantity; Hauler.availableSpace = Hauler.availableSpace - unloadingQuantity; carriedEarth=0; </pre>
haulerNotFilled transition	<pre> setRotation(0); </pre>
continueLoading transition	<pre> Timeout: timeToAdjustWhileEmpty </pre>
loadReadyB transition	<pre> Timeout: timeToLoadBucket carriedEarth = loadingType(); get_Main().soilAvailableForLoading = get_Main().soilAvailableForLoading - carriedEarth; </pre>
continueUnloading transition	<pre> Timeout: timeToAdjustWhileFull setRotation(3.14159265359); unloadingQuantity = unloadingType(); timeToUnloadBucket = timeToUnloadFullBucket *(unloadingQuantity/bucketCapacity); </pre>
haulerFilled transition	<pre> Condition: (Hauler.carriedEarth == Hauler.capacity)    (get_Main().soilAvailableForLoading == 0) (Hauler.carriedEarth == Hauler.capacity)    (get_Main().soilAvailableForLoading == 0) </pre>
returnToWaitingPosition transition	<pre> Timeout: timeToAdjustWhileEmpty </pre>



## The Hauler Agent:

Table A-7: Hauler class elements and transitions in *ABSEMO* (Java code)

Element/s	Java Code
Startup code	<pre>Point pt = get_Main().haulerQueueL.randomPointInside(); setXY(pt.x, pt.y); setRotation(3.14159265359);</pre>
<ul style="list-style-type: none"> <li> capacityUser</li> <li> capacityF</li> <li> capacity</li> </ul>	<pre>boolean isTriangular = capacityUser.startsWith("T"); boolean isUniform = capacityUser.startsWith("U"); boolean isNormal = capacityUser.startsWith("N"); if(isTriangular){ int parLocation1 = capacityUser.indexOf('(',0); int parLocation2 = capacityUser.indexOf(')',parLocation1); int commaLocation1 = capacityUser.indexOf(',',0); int commaLocation2 = capacityUser.indexOf(',',commaLocation1+1); String tempvalue1 = capacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = capacityUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = capacityUser.substring(commaLocation2+1,parL ocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); capacity = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = capacityUser.indexOf('(',0); int parLocation2 = capacityUser.indexOf(')',parLocation1); int commaLocation1 = capacityUser.indexOf(',',0); String tempvalue1 = capacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = capacityUser.substring(commaLocation1+1, parLocation2);</pre>

Element/s	Java Code
	<pre> double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2);  capacity = uniform(value1,value2); } else if(isNormal){ int parLocation1 = capacityUser.indexOf('(',0); int parLocation2 = capacityUser.indexOf(')',parLocation1); int commaLocation1 = capacityUser.indexOf(',',0); String tempvalue1 = capacityUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = capacityUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); capacity = normal(value1,value2); } else{ capacity = Double.parseDouble(capacityUser); } </pre>
<ul style="list-style-type: none"> <li> movingToLoadPositionTimeUser</li> <li> movingToLoadPositionTimeF</li> <li> movingToLoadPositionTime</li> </ul>	<pre> boolean isTriangular = movingToLoadPositionTimeUser.startsWith("T") ; boolean isUniform = movingToLoadPositionTimeUser.startsWith("U") ; boolean isNormal = movingToLoadPositionTimeUser.startsWith("N") ; if(isTriangular){ int parLocation1 = movingToLoadPositionTimeUser.indexOf('(',0); int parLocation2 = movingToLoadPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToLoadPositionTimeUser.indexOf(',',0); int commaLocation2 = movingToLoadPositionTimeUser.indexOf(',',com maLocation1+1); String tempvalue1 = movingToLoadPositionTimeUser.substring(parLo cation1+1, commaLocation1); String tempvalue2 = </pre>

Element/s	Java Code
	<pre> movingToLoadPositionTimeUser.substring(comma Location1+1, commaLocation2); String tempvalue3 = movingToLoadPositionTimeUser.substring(comma Location2+1,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); movingToLoadPositionTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = movingToLoadPositionTimeUser.indexOf('(',0); int parLocation2 = movingToLoadPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToLoadPositionTimeUser.indexOf(', ',0); String tempvalue1 = movingToLoadPositionTimeUser.substring(parLo cation1+1, commaLocation1); String tempvalue2 = movingToLoadPositionTimeUser.substring(comma Location1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); movingToLoadPositionTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = movingToLoadPositionTimeUser.indexOf('(',0); int parLocation2 = movingToLoadPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToLoadPositionTimeUser.indexOf(', ',0); String tempvalue1 = movingToLoadPositionTimeUser.substring(parLo cation1+1, commaLocation1); String tempvalue2 = movingToLoadPositionTimeUser.substring(comma Location1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); </pre>

Element/s	Java Code
	<pre> movingToLoadPositionTime = normal(value1,value2); } else{ movingToLoadPositionTime = Double.parseDouble(movingToLoadPositionTimeU ser); } </pre>
<ul style="list-style-type: none"> <li> haulingTimeUser</li> <li> haulingTimeF</li> <li> haulingTime</li> </ul>	<pre> boolean isTriangular = haulingTimeUser.startsWith("T"); boolean isUniform = haulingTimeUser.startsWith("U"); boolean isNormal = haulingTimeUser.startsWith("N"); if(isTriangular){ int parLocation1 = haulingTimeUser.indexOf('(',0); int parLocation2 = haulingTimeUser.indexOf(')',parLocation1); int commaLocation1 = haulingTimeUser.indexOf(',',0); int commaLocation2 = haulingTimeUser.indexOf(',',commaLocation1+1 ); String tempvalue1 = haulingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = haulingTimeUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = haulingTimeUser.substring(commaLocation2+1,p arLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); haulingTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = haulingTimeUser.indexOf('(',0); int parLocation2 = haulingTimeUser.indexOf(')',parLocation1); int commaLocation1 = haulingTimeUser.indexOf(',',0); String tempvalue1 = haulingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = </pre>



Element/s	Java Code
	<pre> haulingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); haulingTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = haulingTimeUser.indexOf('(',0); int parLocation2 = haulingTimeUser.indexOf(')',parLocation1); int commaLocation1 = haulingTimeUser.indexOf(',',0); String tempvalue1 = haulingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = haulingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); haulingTime = normal(value1,value2); } else{ haulingTime = Double.parseDouble(haulingTimeUser); } </pre>
<ul style="list-style-type: none"> <li> movingToDumpPositionTimeUser</li> <li> movingToDumpPositionTimeF</li> <li> movingToDumpPositionTime</li> </ul>	<pre> boolean isTriangular = movingToDumpPositionTimeUser.startsWith("T") ; boolean isUniform = movingToDumpPositionTimeUser.startsWith("U") ; boolean isNormal = movingToDumpPositionTimeUser.startsWith("N") ; if(isTriangular){ int parLocation1 = movingToDumpPositionTimeUser.indexOf('(',0); int parLocation2 = movingToDumpPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToDumpPositionTimeUser.indexOf(',',0); int commaLocation2 = movingToDumpPositionTimeUser.indexOf(',',com maLocation1+1); String tempvalue1 = movingToDumpPositionTimeUser.substring(parLo </pre>

Element/s	Java Code
	<pre> cation1+1, commaLocation1); String tempvalue2 = movingToDumpPositionTimeUser.substring(comma Location1+1, commaLocation2); String tempvalue3 = movingToDumpPositionTimeUser.substring(comma Location2+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); movingToDumpPositionTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = movingToDumpPositionTimeUser.indexOf('(',0); int parLocation2 = movingToDumpPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToDumpPositionTimeUser.indexOf(',',0); String tempvalue1 = movingToDumpPositionTimeUser.substring(parLo cation1+1, commaLocation1); String tempvalue2 = movingToDumpPositionTimeUser.substring(comma Location1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); movingToDumpPositionTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = movingToDumpPositionTimeUser.indexOf('(',0); int parLocation2 = movingToDumpPositionTimeUser.indexOf(')',par Location1); int commaLocation1 = movingToDumpPositionTimeUser.indexOf(',',0); String tempvalue1 = movingToDumpPositionTimeUser.substring(parLo cation1+1, commaLocation1); String tempvalue2 = movingToDumpPositionTimeUser.substring(comma Location1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); </pre>

Element/s	Java Code
	<pre> double value2 = Double.parseDouble(tempvalue2); movingToDumpPositionTime = normal(value1,value2); } else{ movingToDumpPositionTime = Double.parseDouble(movingToDumpPositionTimeU ser); } </pre>
<ul style="list-style-type: none"> <li> dumpingTimeUser</li> <li> dumpingTimeF</li> <li> dumpingTime</li> </ul>	<pre> boolean isTriangular = dumpingTimeUser.startsWith("T"); boolean isUniform = dumpingTimeUser.startsWith("U"); boolean isNormal = dumpingTimeUser.startsWith("N"); if(isTriangular){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); int commaLocation2 = dumpingTimeUser.indexOf(',',commaLocation1+1 ); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = dumpingTimeUser.substring(commaLocation2+1,p arLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); dumpingTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, </pre>

Element/s	Java Code
	<pre> commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); dumpingTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); dumpingTime = normal(value1,value2); } else{ dumpingTime = Double.parseDouble(dumpingTimeUser); } </pre>
<ul style="list-style-type: none"> <li> returningTimeUser</li> <li> returningTimeF</li> <li> returningTime</li> </ul>	<pre> boolean isTriangular = returningTimeUser.startsWith("T"); boolean isUniform = returningTimeUser.startsWith("U"); boolean isNormal = returningTimeUser.startsWith("N"); if(isTriangular){ int parLocation1 = returningTimeUser.indexOf('(',0); int parLocation2 = returningTimeUser.indexOf(')',parLocation1); int commaLocation1 = returningTimeUser.indexOf(',',0); int commaLocation2 = returningTimeUser.indexOf(',',commaLocation1 +1); String tempvalue1 = returningTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = </pre>

Element/s	Java Code
	<pre> returningTimeUser.substring(commaLocation1+1 , commaLocation2); String tempvalue3 = returningTimeUser.substring(commaLocation2+1 ,parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); returningTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = returningTimeUser.indexOf('(',0); int parLocation2 = returningTimeUser.indexOf(')',parLocation1); int commaLocation1 = returningTimeUser.indexOf(',',0); String tempvalue1 = returningTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = returningTimeUser.substring(commaLocation1+1 , parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); returningTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = returningTimeUser.indexOf('(',0); int parLocation2 = returningTimeUser.indexOf(')',parLocation1); int commaLocation1 = returningTimeUser.indexOf(',',0); String tempvalue1 = returningTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = returningTimeUser.substring(commaLocation1+1 , parLocation2);  double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); returningTime = normal(value1,value2); } </pre>

Element/s	Java Code
	<code>else{ returningTime = Double.parseDouble(returningTimeUser); }</code>
goToLoadingPosition transition	Message: unconditional (any message) Loader=msg; moveToInTime(Loader.getX(), Loader.getY() +30, movingToLoadPositionTime);
startLoading transition	send("BEGIN UNLOADING", Loader);
finishLoading transition	Message: "FINISHED LOADING" Loader = null;
startHauling transition	Point pt = get_Main().haulerQueueD.randomPointInside(); moveToInTime(pt.x, pt.y, haulingTime);
readyForSpotter transition	get_Main().haulerRequestsSpotter( this );
goToDumpingPosition transition	Message: unconditional (any message) Spotter=msg; moveToInTime(Spotter.getX(), Spotter.getY() +25, movingToDumpPositionTime);
spotterAvailable transition	send("BEGIN DUMPING", Spotter); setRotation(3.14159265359);
startReturning1 transition	Message: "FINISHED DUMPING" get_Main().dumpedSoil = get_Main().dumpedSoil + carriedEarth; carriedEarth = 0; availableSpace = capacity; Spotter = null;
readyToReturn transition	Point pt = get_Main().haulerQueueL.randomPointInside(); moveToInTime(pt.x, pt.y, returningTime);
readyForLoader transition	setRotation(3.14159265359);

## The Spotter Agent:

**Table A-8:** Spotter class elements and transitions in *ABSEMO* (Java code)

Element/s	Java Code
<ul style="list-style-type: none"> <li> timeToAdjustPositionUser</li> <li> timeToAdjustPositionF</li> <li> timeToAdjustPosition</li> </ul>	<pre> boolean isTriangular = dumpingTimeUser.startsWith("T"); boolean isUniform = dumpingTimeUser.startsWith("U"); boolean isNormal = dumpingTimeUser.startsWith("N"); if(isTriangular){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); int commaLocation2 = dumpingTimeUser.indexOf(',',commaLocation1+1 ); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, commaLocation2); String tempvalue3 = dumpingTimeUser.substring(commaLocation2+1,p arLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); double value3 = Double.parseDouble(tempvalue3); dumpingTime = triangular(value1,value2,value3); } else if(isUniform){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = </pre>

Element/s	Java Code
	<pre> Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); dumpingTime = uniform(value1,value2); } else if(isNormal){ int parLocation1 = dumpingTimeUser.indexOf('(',0); int parLocation2 = dumpingTimeUser.indexOf(')',parLocation1); int commaLocation1 = dumpingTimeUser.indexOf(',',0); String tempvalue1 = dumpingTimeUser.substring(parLocation1+1, commaLocation1); String tempvalue2 = dumpingTimeUser.substring(commaLocation1+1, parLocation2); double value1 = Double.parseDouble(tempvalue1); double value2 = Double.parseDouble(tempvalue2); dumpingTime = normal(value1,value2); } else{ dumpingTime = Double.parseDouble(dumpingTimeUser); } </pre>
haulerAvailable transition	<pre> Condition: get_Main().thereAreRequestsSpotter() Hauler = get_Main().getRequestSpotter(); send( this, Hauler ); </pre>
startSpotting transition	Condition: "BEGIN DUMPING"
finishedSpotting transition	Timeout: dumpingTime (Hauler variable)
timeToGetReady transition	Timeout: timeToAdjustPosition