2010

The History, Philosophy, and Practice of Agent-Based Modeling and the Development of the Conceptual Model for Simulation Diagram

Brian L. Heath
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Engineering Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/982

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu, library-corescholar@wright.edu.
The History, Philosophy, and Practice of Agent-Based Modeling and the Development of the Conceptual Model for Simulation Diagram

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

By

BRIAN L. HEATH
B.S., Kettering University, 2006
M.S., Wright State University, 2008

2010
Wright State University

Frank W. Ciarallo, Ph.D.
Dissertation Co-Adviser

Raymond R. Hill, Ph.D.
Dissertation Co-Adviser

Ramana V. Grandhi, Ph.D.
Director, Ph.D. in Engineering Program

John A. Bantle, Ph.D.
V.P. for Research and Graduate Studies and Interim Dean of Graduate Studies

Committee on Final Examination

Misty Blue, Ph.D.

Frank W. Ciarallo, Ph.D.

Thomas C. Hartrum, Ph.D.

Raymond R. Hill, Ph.D.

Yan Liu, Ph.D.

Ed Pohl, Ph.D.
ABSTRACT


This research advances ABM as a generic analysis tool such that ABM can reach its full potential as a revolution in modeling and simulation. To achieve this goal, the field of ABM is examined from many perspectives. The first perspectives examined are complex systems, the historical emergence of ABM, and philosophical issues related to ABM. These topics establish some clear foundations for the field across multiple disciplines. Next the current practice of ABM is investigated. Through a comprehensive 279 article survey some current deficiencies and opportunities in ABM are identified. Based on these opportunities, a new diagramming technique called the Conceptual Model for Simulation (CM4S) Diagram is developed. Fundamentally, the CM4S Diagram represents the first diagramming technique designed specifically for the effective representation, construction, and sanctioning of ABM computer simulations based on identified needs in the ABM modeling field and simulation modeling philosophy. Finally, the effectiveness of the CM4S Diagram is evaluated through the development of social science, military, and supply chain ABM simulations.
# Contents

1 Introduction 1

I. The History, Philosophy, and Current Practice of Agent-Based Modeling 20

2 What are Complex Systems? 21
   2.1 Understanding Systems and Their Complexity 21
   2.2 The Types of Model Systems and Their Complexity 26
   2.3 Exploring the Landscape of Model Systems and Complexity 30
   2.4 Conclusion 35

3 The Emergence of Agent-Based Modeling 37
   3.1 The Beginning: Computers 38
   3.2 The Synthesis of Natural Systems: Cellular Automata and Complexity 40
   3.3 The Analysis of Natural Systems: Cybernetics and Chaos 44
   3.4 Towards Today’s ABM: Complex Adaptive Systems 49
   3.5 Conclusion 52

4 Simulation and Agent-Based Modeling Validation Philosophy 54
   4.1 Why All Simulations are Invalid 56
   4.2 What Does Simulation Validation Really Mean in Practice? 63
   4.3 What Good are Simulations? 68
   4.4 What Good is ABM? 73
   4.5 Conclusion 75

5 A Survey of Agent-Based Modeling Practices 77
   5.1 Introduction 77
      5.1.1 What is Holding Back ABM? 77
      5.1.2 What is the Current State of ABM? 79
   5.2 Methodology 79
      5.2.1 Collection of the Sample 80
      5.2.2 Categorization and Data Collection Strategy 83
         5.2.2.1 Software 83
         5.2.2.2 Field of Study 83
         5.2.2.3 Reference to the Complete Model 84
5.2.2.4 Validation Technique(s) .................................................. 84
5.2.2.5 Purpose of the Simulation ............................................... 86
5.3 Results ..................................................................................... 88
5.3.1 Software .............................................................................. 88
5.3.2 Fields of Study ..................................................................... 89
5.3.3 Purpose of the Simulation ..................................................... 89
5.3.4 Reference to the Complete Model ......................................... 91
5.3.5 Validation (Not Considering Technique) ............................... 93
5.3.6 Validation Techniques .......................................................... 95
5.4 Discussion ............................................................................... 97
5.4.1 Software and Verification ..................................................... 97
5.4.2 Addressing the Many Fields of Study or Creating a New One 98
5.4.3 Redefining the Meaning of Results by Purpose .................... 99
5.4.4 Providing a Reference to the Complete Model ..................... 100
5.4.5 Complete Validation is Required for Every Model ................ 101
5.4.6 Statistical vs. Non-Statistical Validation Techniques ............ 102
5.5 Conclusion .............................................................................. 103

II. The Development of the Conceptual Model for Simulation
Diagram .......................................................... 105

6 The Sanctioning Solution Concept and Why It is Required ......... 106
6.1 Exploring Requirements .......................................................... 107
6.1.1 How Simulations are Developed .......................................... 107
6.1.2 Sanctioning Emphasis .......................................................... 109
6.1.3 Conceptual Modeling ............................................................ 111
6.1.4 Difficulties in Modeling Organized Complex Systems .......... 113
6.1.5 Re-visiting Peer-Level Sanctioning ........................................ 114
6.1.6 Summary of the Key Requirements ...................................... 115
6.2 Further Justification and Insight into a New ABM Sanctioning Methodology .................................................. 116

7 An Exploration of Diagramming Techniques and Their Capabilities 123
7.1 An Overview of Diagrams .......................................................... 123
7.2 Capabilities of Diagramming Techniques that Describe Model Systems 125
7.2.1 Organizational Diagramming Techniques ............................. 125
7.2.2 Behavioral Diagramming Techniques .................................... 127
7.2.2.1 Process Flow Behavioral Diagramming Techniques ......... 128
7.2.2.2 Machine Behavioral Diagramming Techniques .............. 130
7.3 Capability Gap Analysis and Conclusions ................................. 134

8 The Conceptual Model for Simulation Diagram ......................... 137
8.1 The Need for a New Diagramming Technique .......................... 137
8.2 Adapting Statecharts to Satisfy the Conceptual Sanctioning Requirements .................................................. 139
8.3 Description of the CM4S Diagram ............................................. 141
# List of Figures

1. A Landscape of Model Systems .................................. 8
2. The Roles of Simulations ........................................ 10
3. Screen shot of the DC Order Picker Simulation .................. 18
4. An Example Problem’s Progression and the Complexity of the Model System .................................................. 25
5. A Landscape of Model Systems and Complexity .................... 31
6. Sample Model Systems Landscape for Solving the Ant Colony Problem 33
7. Relationship between a System, a Theory/Model, and a Simulation .... 57
8. Different Roles of a Simulation ........................................ 73
9. Number of Articles per Year in the Sample ......................... 82
10. Articles per Publication Outlet in the Sample ..................... 82
11. A Simplified Simulation Development Process ..................... 85
12. Purpose of the Simulation ........................................... 87
13. Histogram of Top Used Software ................................... 88
14. Breakdown of Articles by Field .................................... 89
15. Simulation Purpose by Year ........................................ 90
16. Simulation Purpose by Field ........................................ 90
17. Reference to Complete Model by Year .............................. 91
18. Reference to Complete Model by Field ............................ 92
19. Reference to the Complete Model by Purpose ..................... 92
20. Validation of the Simulations (Not Considering Technique) ....... 93
21. Validation by Year .................................................. 94
22. Validation by Field of Study ....................................... 94
23. Validation by Purpose ............................................... 95
24. Validation Technique by Year ...................................... 96
25. Validation Technique by Field ..................................... 96
26. Validation Technique by Purpose ................................... 97
27. A Simplified Simulation Development Process .................... 108
28. Relationship between System Understanding and Simulation Sanctioning Emphasis ........................................ 110
29. Current Practice Needs Mapped to Diagramming Technique Requirements .................................................. 117
31. State Diagram Example of a Telephone ............................ 132
32. Statechart Diagram Example of a Ceiling Fan ..................... 133
33. Sugarscape CM4S Diagram Example ................................. 142
34. CM4S Diagram Shape Descriptions .................................. 146
## List of Tables

1. Diagramming Technique Capability Analysis . . . . . . . . . . . . . . 14
2. Wolfram’s Cellular Automata Classifications [124] . . . . . . . . . . . 43
3. Diagramming Technique Capability to Requirement Analysis . . . . . 135
4. CM4S Diagram Prototype Shape Descriptions . . . . . . . . . . . . . . 176
Introduction

First emerging during the Industrial Revolution and initially concerned with manufacturing, Industrial Engineering is broadly defined as:

“[a field] concerned with the design, improvement, and installation of integrated systems of [people], material, equipment, and energy...[that]...draws upon specialized knowledge and skill in the mathematical, physical, and social sciences together with the principles and methods of engineering analysis and design to specify, predict, and evaluate the results to be obtained from such systems.” [73]

Noticeably missing in this definition is the specific system that Industrial Engineers are interested in understanding. However, this is an accurate description because Industrial Engineering is primarily concerned with the design, understanding, and analysis of artificial (or man-made [107]) systems as well as how they relate to other systems. In other words, Industrial Engineering is not concerned with specializing in any particular system, but is focused on having the capability to understand, design, and evaluate any type of system. This does not mean that Industrial Engineers are not technical experts for any particular system, only that they have adopted and developed tools and techniques that aid in the understanding of systems in a holistic sense. Furthermore, the tools used in Industrial Engineering can be categorized as trying to solve one (or some) of the three problem types observed in systems.

In 1948, Weaver identified three types of problems that are encountered in science, and therefore in systems: Problems of Simplicity, Disorganized Complexity, and Organized Complexity [115]. In Problems of Simplicity, the abstraction of the system is such that only a few variables are examined and the relationship between them
is determined. These types of systems are typically represented with mathematical equations and were some of the first problems examined in Industrial Engineering. For example, in 1898 Frederick Taylor, often called the Father of Industrial Engineering, analyzed workers shoveling coal to determine the relationship between worker fatigue and how much they could shovel in a day. By focusing on a few variables Taylor determined that a worker should only shovel 21.4 lbs of coal per shovel load to minimize fatigue while maximizing output [33].

The second problem type identified by Weaver are Problems of Disorganized Complexity. In these problems, the abstracted system of interest is composed of hundreds or thousands of variables and the characteristics of the whole abstracted system are studied as opposed to the individual variables or entities. In other words, inferences are made about the system based on the net effect of all the individual variables and not from understanding each of the individual variables. Examples of these problems abound in Industrial Engineering. Examining these problems requires only the observation of the input and output and does not require understanding how the internal structure of the system creates the observed output [115]. For example, an Industrial Engineer can measure thousands of parts to develop some distribution that describes the net outcome of the quality of the parts for a manufacturing system to make an inference about the probability that the next 100 parts will pass the quality inspection. Notice that the Industrial Engineer does not need to understand all of the variables that impact the quality of each part to make an inference about a future set of parts [78]. Instead, the many disorganized variables impacting the parts create a simple abstraction from a complex abstraction.

The emergence of simplicity from a problem of Disorganized Complexity brings up an interesting question when considering the types of problems Industrial Engineers face: are Problems of Simplicity and Problems of Disorganized Complexity any different? Note that the idea that simplicity emerges from complexity is not a foreign idea
we naturally observe this phenomena everyday in the way we abstract systems. From a complex abstraction of atoms emerges a molecule abstraction and from many molecules emerges a cell abstraction and so forth. In essence, simplicity and complexity are a matter of abstraction. Therefore, it could be conjectured that Problems of Simplicity and Disorganized Complexity are only different in terms of abstracting complexity (the structure) and not in the absolute complexity that they represent. Any Simplicity problem can be broken into a Disorganized Complexity problem and vice versa. A further similarity is that these problems describe the “what” relationships at a given abstraction level. For example, these problems help answer questions like what will happen when force is applied to this mass or what level of a factor impacts the growth of a particular plant. Fundamentally, Problems of Simplicity and Disorganized Complexity provide no insight into how the characteristics observed at an abstraction level developed.

The final problem type identified by Weaver are Problems of Organized Complexity. In these problems, the system is abstracted into a medium number of highly interrelated variables that together produce an organic whole of the system. This type of problem is fundamentally different from the previous two because it looks across different abstraction levels of the system. For example, a Problem of Organized Complexity could help try to understand how individual birds come together to form a flock with no clear leader [25]. This problem is not just concerned with an individual bird’s behavior or an entire flock’s behavior, which are two possible independent abstraction levels, but with how the abstraction level of individual birds leads to the abstraction level of a flock. The key to Problems of Organized Complexity is that their analysis results in “how” questions that are aimed at gaining insight into the system of interest.

Industrial Engineering has many examples where analyzing Problems of Organized Complexity would be valuable. For example, understanding how individual cars on
a road can result in traffic jams can lead to better designed traffic control systems while understanding how individuals form groups to accomplish a task can lead to better designed organizations. However, unlike Problems of Simplicity and Disorganized Complexity, an extensive set of tools and techniques have not been thoroughly developed to aid in analyzing Problems of Organized Complexity.

The lack of tools and techniques for Problems of Organized Complexity does not mean a lack of interest. Many “classical” theories attempt to explain a Problem of Organized Complexity. Darwinian Evolution describes how the actions and interaction of individuals over time results in the global adaptation of the group. Also, Adam Smith’s Invisible Hand in Economics describes an overall improvement in a community that occurs when individuals each try to maximize their own utility [9].

A key reason for the lack of tool development in Problems of Organized Complexity is their exhibited nonlinear behavior and the general lack of analytical tools capable of analyzing nonlinear behavior. The computer has provided a tool capable of “break[ing] the present stalemate created by the failure of the purely analytical approach to nonlinear problems” [113]. The fundamental reason for this is the capability of computers to represent a natural process through time (Church-Turing Hypothesis [70]). Thus, we can mimic a system with all of its non-linearities within a computer and not be concerned with coming to formal, theoretical solutions; we can understand the systems through empirical means.

With the tremendous growth in computing capabilities has come an ability to simulate an abstracted system to help understand many different types of problems. The Discrete-Event Simulation Paradigm is quite a popular tool resulting in many advances. These advances include steps to build a successful simulation, improvement of technical aspects of simulation (e.g. random number generation), techniques to analyze and produce better output measures, and methods to help validate and verify that the simulation is an accurate representation of the system of interest [14, 69].
Although these advances improved computer simulation, they were mainly geared for the Discrete-Event Simulation Paradigm, which is primarily used for Problems of Simplicity or Disorganized Complexity.

A more recent development in simulation that is geared for Problems of Organized Complexity is the Agent-Based Modeling (ABM) Paradigm. ABM is a computational simulation paradigm composed of autonomous entities (agents) that interact with each other and their environment [39, 76]. ABM is capable of representing the kinds of systems that are encountered in Problems of Organized Complexity. They can represent how one level of abstraction (individual agents) can generate a new level of abstraction through the interactions that occur in the system, such as how individual birds form a flock. Due to ABM's ability to represent and analyze Problems of Organized Complexity, the ABM simulation paradigm has gained favor in many different fields, from those that have traditionally used simulation to those that have not.

The ability of ABM to analyze Problems of Organized Complexity, its relatively recent development, and its wide-spread use makes it an analysis tool rich in research opportunities and challenges. Since ABM is capable of analyzing Organized Complexity Problems, advancing the development of this tool allows science and engineering to expand their understanding of how systems transition from one abstraction level to another. Previously analytically unanswerable questions such as how do traffic jams form and how exactly do particular actions of individuals results in global adaptation can be examined. As with any new tool or field, there are often many theoretical, philosophical, and application questions that are not thoroughly addressed without a significant amount of research. The field of ABM is full of research and application questions such as how is ABM different from other simulation paradigms and what implications do these have on ABM as an analysis tool? Establishing and answering some of the key research questions for ABM contributes to the development of this important analysis and research tool by creating a more solid theoretical and
philosophical foundation that can lead to better ABM applications.

With these research opportunities and challenges in mind, the goal of this research is to advance ABM as a generic analysis tool to help ABM reach its full potential as a revolution in the modeling and simulation domain [13]. This document has three main parts. Part I examines the history, philosophy, and current practice of ABM to establish a philosophical foundation of ABM and to identify improvement opportunities in the domain. Part II discusses the development a new diagramming technique called the Conceptual Model for Simulation (CM4S) Diagram to address the special issues in validating agent-based models. Part III demonstrates the evolution and effectiveness the CM4S Diagram through the construction and evaluation of military and supply chain warehousing ABM simulations. The final chapter concludes with the contributions of this research as well as future research opportunities.

Part I: The History, Philosophy, and Current Practice of ABM

To advance ABM one must first understand that ABM is a generic analysis tool for understanding complex systems. To develop ABM so that its scope extends beyond one particular problem or domain of interest its practice must incorporate understanding of the complex theories, tools, and methods of systems and emphasize systems thinking rather than domain specific thinking. Developing ABM from the systems perspective is vital since a commonality between all problems and disciplines is that they all involve systems. For this reason, this research emphasizes ABM as an analysis tool used to understand complex systems. Chapter 1 describes the portion of research on complex systems and reconcile the various definitions of complexity that currently exist.

In Chapter 2, the meaning of complexity and complex systems is explored. First, a system is defined as something that translates input into output. Second, there exist two types of systems: real systems and model systems. Real systems are infinitely
complex and model systems are finite abstractions that are used to understand real systems. Thus, there are two types of systems people should refer to when discussing complex systems. Since real systems are infinitely complex and model systems can range in complexity, the term “complex systems” actually refers to complex model systems. Next, based upon Weaver’s problems encountered in science, model systems are decomposed into four main categories: Primitive Model Systems (PMSs), Simple Model Systems (SMSs), Disorganized Complex Model Systems (DCMSs), and Organized Complex Model Systems (OCMSs). Both DCMSs and OCMSs are complex model systems, but describe slight differences in the results produced from the model system. Together DCMS and OCMSs effectively reconcile the different meanings of complex systems.

After creating a classification of model systems a landscape is created to show the relationship between model systems based on the number of components and understanding level of the real system problem. In this framework shown in Figure 1, PMSs are found in the region where there are few number of components and understanding levels are low, SMSs are found in the regions where the number of components is low and understanding levels are high, DCMSs are found in the regions where the number of components is high and understanding levels are moderate, and OCMSs are found in regions where the number of components is high and understanding levels are low. This landscape also describes how real system problems are solved using model systems. Therefore, it can aid in solving problems by connecting the current level of understanding with appropriate model system tools to meet the objectives. Finally, this framework effectively shows how the connection between model systems, the number of components, and the level of understanding achieves the fundamental goal of science to make the complex simple.

After establishing the meaning of complex systems and complexity, the next step to advance ABM is to develop a sound understanding of how ABM came into exis-
Figure 1: A Landscape of Model Systems

<table>
<thead>
<tr>
<th>Number of Components in the Model System</th>
<th>Understanding Level of the Real System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

- Organized Complex Model Systems
- Disorganized Complex Model Systems
- Primitive Model Systems
- Simple Model Systems
tence. It seems that almost every article discussing ABM includes some account of the historical development of ABM. Often this history does not discuss the fundamental theories and diverse fields of inquiry that eventually led to ABM’s emergence and the corresponding shift of emphasis from top-down to bottom-up analysis. To begin to unfold and account for the detailed development of ABM, Chapter 3 explores some of the scientific developments in computers, cybernetics, complexity, chaos, and systems thinking that helped lead to the emergence of ABM. By connecting old theories to several key principles of ABM, a historical perspective into ABM and complexity is presented that provides a clearer understanding of the field, shows the benefits to be gained by understanding the diverse origins of ABM, and can serve as a starting point for others interested in exploring other theories and ideas that laid the foundation for the ABM Paradigm.

Another important need to advance ABM is to explore the philosophical issues related to simulation and particularly ABM. As the popularity and usage of simulation and ABM continues to expand, it is valuable to establish the philosophical issues related to computer simulation as well as its limitations. This is especially true when considering that some believe that simulation is becoming the epistemological engine of our time [65]. Chapter 4 establishes the relationship between simulation and the philosophy of science, discusses the limitations of all simulations as invalid representations of reality, redefines the process of simulation validation through the new concept called sanctioning, and discusses the practical application of simulation and validation.

This examination of the practical and philosophical application of simulation and validation identifies three primary roles of simulation: Generators, Mediators, and Predictors. A Generator is a simulation where little is known about the system of interest and it is used primarily to determine if a given conceptual model/theory is capable of generating observed behavior of the system. A Mediator is a simulation
where the system is moderately understood and it is used primarily to establish the capability of the conceptual model to represent the system and to then gain some insight into the system’s characteristics and behaviors. A Predictor is a simulation where the system is well understood and it is used primarily to estimate or predict a system’s behavior with little time spent on ensuring that the conceptual model is correct because this aspect of the simulation has already been established. A framework shown in Figure 2 connects the role of a simulation with the level of understanding about the real system. In turn this framework begins to define the appropriate roles, expectations, needs, and validation techniques for ABM.

Upon establishing philosophical and historical foundations, the next phase to advance ABM is to identify the current practice of ABM. Chapter 5 includes a comprehensive survey of 279 ABM articles conducted to identify opportunities that can advance the field as well as to compare and contrast current ABM practices with the key complexity, historical, and philosophical ABM concepts discussed in the previous chapters.

The survey identified six specific research directions, needs, and opportunities for ABM. First, development and documentation tools for ABM need to be independent of software. Second, ABM needs to be studied as an independent discipline yet also as a subset of the simulation discipline. From this standard techniques, practices,
philosophies, and methodologies are needed to extend ABM as a functional analysis tool. Third, simulationists should have appropriate expectations for ABM since ABM is used for the nontraditional simulation role of a generator. Fourth, published articles need to incorporate sufficient information about the model so other researchers can independently develop and evaluate the effectiveness of these models. The fifth, and most significant, conclusion reached from the survey is that reviewers and publication outlets must require the complete validation and documentation of model. Finally, both statistical and non-statistical validation techniques specifically appropriate for ABM need to be developed and become part of the training for those building these models.

Part II: The Development of the Conceptual Model for Simulation Diagram

Based upon evaluating the current ABM practices and identifying opportunities that could advance the field, Chapter 6 discusses the refinement of the needs and identifies a solution concept with a set of detailed requirements. Here the key solution concept is the development of a diagramming technique that impacts the way agent-based models are constructed, validated, and reported. The major reason for considering a diagramming technique as a solution concept is that diagrams are graphical languages that can describe entities and processes, provide documentation, communicate ideas, and emphasize important aspects of the artifacts being described [23]. These capabilities accurately address issues identified as needs in the ABM survey. Next, the requirements of the diagramming technique are developed by further investigating/re-investigating the process of simulation construction and finding the appropriate emphasis when validating agent-based models. This chapter also describes identifying the types of systems being simulated using the ABM paradigm.

By examining these topics, the following detailed requirements for a diagramming technique are derived:
1. Aids in learning and conveying system knowledge

2. Incorporates proper engineering judgment

3. Aids in translating the conceptual model into a computerized model

4. Emphasizes the development and sanctioning of the micro-level behaviors

5. Displays the theories and assumptions built into the model for quantitative analysis

6. Conveys the conceptual model’s logic and structure for qualitative analysis

7. Completely represents the simulation so it can be reproduced by independent evaluators

8. Provides justification for all structures and actions in the simulation

9. Reviewable by evaluators of varied simulation and domain expertise levels

10. Can represent Disorganized and Organized Complex Systems

The key aspect of these requirements is the emphasis on the development and validation of the conceptual model of the simulation. This technique of ABM are primarily used to explore real systems where the modeler has a low level of understanding. This creates a need to place more validation emphasis on the conceptual model rather than the output being correct. Thus, the diagramming technique must be able to aid in the development, construction, validation, evaluation, and documentation of the conceptual model of a computer simulation.

Chapter 7 explores diagramming techniques and their capabilities to determine if any existing diagramming technique satisfies all of the requirements. First, diagramming techniques are defined as graphical languages that communicate features of an object or concept of interest. Next, a variety of diagramming techniques from
systems engineering and computer science domains are identified and placed into two categories based on their capabilities and objectives: Organizational Diagramming Techniques (ODTs) and Behavioral Diagramming Techniques (BDTs). ODTs (e.g. Entity-Relation Diagrams, \(N^2\) Charts, UML 2.0 Structural Diagrams, etc.) capture static relationships and highlight the structure of various components of the model system. The inability of ODTs to represent dynamic systems eliminates them from consideration.

BDTs capture the dynamic control and execution of activities or functions of a model system and describe the desired high-level capabilities sought after in candidate diagramming techniques. Furthermore, BDTs are broken into two categories: Process Flow Behavioral Diagramming Techniques (PFBDTs) and Machine Behavioral Diagramming Techniques (MBDTs). PFBDTs (e.g. Functional Flow Block Diagrams, Control Flow Diagrams, SysML Activity Diagrams, etc.) describe the flow of control of activities or functions of a model system, but do not describe precisely how inputs are translated into outputs. The inherent inability of PFBDTs to provide a description of a model system such that others can mimic exactly how the model system operates eliminates them from consideration. MBDTs (e.g. Petri Nets, State Transition Diagrams, Statecharts, and UML 2.0 and SysML State Machine Diagrams) describe the dynamic modes of a model system and the events that cause the model system to transition to other modes. MBDTs have additional syntax and semantics to describe the engine of a model system that is missing in PFBDTs. Thus, MBDTs satisfy more of the requirements than any other diagramming techniques. The capabilities of each diagramming technique to satisfy the requirements is shown in Table 1.

Despite MBDTs ability to satisfy many of the requirements, there are three requirements they fail to satisfy. These unsatisfied requirements are:

- The ability to emphasize the development and sanctioning of micro-level be-
## Table 1: Diagramming Technique Capability Analysis

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Behavioral Diagramming Technique</th>
<th>Process Flow</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Functional Flow Block Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Enhance Function Flow Block Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Behavior Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Control Flow Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>UML 2.0 &amp; SysML Activity Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>UML 2.0 &amp; SysML Interaction Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Petri Nets</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>State Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Statecharts</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>UML 2.0 &amp; SysML State Machine Diagrams</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1. Aids in learning and conveying system knowledge
2. Incorporates proper engineering judgment
3. Aids in translating the conceptual model into a computerized model
4. Emphasizes the developing and sanctioning of micro-level behaviors
5. Displays the theories and assumptions built into the model for quantitative analysis
6. Conveys the conceptual model's logic and structure for qualitative analysis
7. Completely represents the simulation so it can be reproduced by independent evaluators
8. Provides justification for all structures and actions in the simulation
9. Reviewable by evaluators of varied simulation and domain expertise levels
10. Must be able to represent Organized and Disorganized Complex Systems
haviors (Req. #4);

- The ability to display the theories and assumptions built into the model for quantitative analysis (Req. #5); and

- The ability to provide justification for all structures and actions in the simulation (Req. #8).

The common theme among requirements is their concern with the sanctioning and documenting of the theories and assumptions built into the simulation. This leads to the conclusion that current diagramming techniques focus primarily on describing the whats and the hows. They do not provide the whys, which is an important aspect of modeling and scientific evaluation. To fill this capability gap, a new diagramming technique is needed that describes the model system in a similar manner as MBDTs and includes valuable conceptual sanctioning information.

In Chapter 8, justification for the development of the CM4S Diagram is presented, its semantics and syntax are defined, and an example of its use is demonstrated. The goal of creating the CM4S Diagram is to address the unsatisfied requirements regarding the sanctioning and development of the conceptual model of ABM simulations. Thus, the CM4S Diagram adapts the fundamental components of Statecharts (a proven MBDT) to include new properties and shapes to distinguish elements that are important in the development of the conceptual model for a simulation. The unique element of the CM4S Diagram is inclusion of a database of properties that extends the diagramming technique to include a new dimension of information that aids in the development, construction, documentation, and sanctioning of the conceptual model.

To demonstrate the functionality and effectiveness of the CM4S Diagramming Technique a CM4S Diagram of the Sugarscape ABM Simulation [35] is constructed. The Sugarscape Simulation is selected because it is both instructive in how to con-
struct a CM4S Diagram for an ABM simulation and informative because it is a well-known in the literature, fairly basic, and a purely notional simulation that utilizes fundamental concepts found in the ABM Paradigm. This example demonstrates how the diagramming technique is capable of representing the conceptual model of an ABM simulation. It captures and documents the key activities, behaviors, timing, statistics, and characteristics of the conceptual model of the simulation as well as documents the “why” information of each shape for sanctioning purposes. Thus, the CM4S Diagram effectively satisfies all of the requirements based on the needs to advance ABM as a generic analysis tool. Furthermore, the CM4S Diagram represents the first diagramming technique designed specifically for the effective representation, construction, and sanctioning of ABM computer simulations based on identified needs in the ABM modeling field and simulation modeling philosophy.

**Part III: The Evaluation of the Conceptual Model for Simulation Diagram**

In Chapter 8, the CM4S Diagram is presented in its final version. However, as with any newly developed product, the CM4S Diagram went through several revisions and evaluations prior to the final version. Part III of the document reviews the evolution of the CM4S Diagram through two rounds of improvement studies where a version of the CM4S Diagram is developed, used to construct an ABM simulation, evaluated for flaws and overall effectiveness, and then improved. The revisions resulting from these studies represents a significant improvement and refinement of the CM4S Diagram. Furthermore, constructing unique ABM simulations from various domains and levels of complexity highlights the robustness of the CM4S Diagram as well as the successful application of the theories, principles, and philosophies developed and discussed in Part I.

In Chapter 9, the prototype version of the CM4S Diagram is used as a proof-of-concept to reproduce Scenario 1 of the Bay of Biscay Military ABM Simulation as
described by Champagne [26] to further advance the CM4S Diagram as a tool for ABM and to evaluate its effectiveness. The main reason for replicating this agent-based model is that it is well documented and validated. This places the emphasis not on developing a new simulation model but on constructing and evaluating the CM4S Diagram Prototype. In this WWII simulation, Allied Planes search the Bay of Biscay for German U-boats (submarines) in an attempt to replicate historical results and eventually improve aerial search patterns. Upon building the CM4S Diagram Prototype for this ABM simulation, it was found that the prototype was able to effectively capture and convey the conceptual model of a complex system, provide justification for all behaviors and actions, and aid in translating the conceptual model into a validated simulation model. However, several improvement opportunities for the initial prototype were identified. These included the need for more descriptive simulation properties for shapes, new shapes describing data collection, and a better representation of interactions between blocks. By utilizing the CM4S Diagram Prototype, the military ABM simulation is successfully and effectively sanctioned, documented, and replicated. Therefore, the proof-of-concept study is deemed successful and provides some evidence that the concept of the CM4S Diagram merits further development and could aid in the advancement of ABM as an analysis tool.

In Chapter 10, a revised version of the CM4S Diagram is used to develop a new ABM simulation of picking activities in a supply chain distribution center to further advance and evaluate the effectiveness of the CM4S Diagram. The key motivation for simulating this system is personal experience and the lack of literature discussing simulations capable of representing the congestion component of order pickers (something ABM simulation is capable of capturing). In this simulation, order pickers traverse the distribution center to pick all of the items on their order list while navigating to avoid collisions. Thus, congestion and travel time are directly measured. A screenshot of the simulation is shown in Figure 3. After constructing, sanctioning, running,
Figure 3: Screen shot of the DC Order Picker Simulation
and analyzing the results of this distribution center ABM simulation, the overall effectiveness of the revised CM4S Diagram is evaluated. It is found that the technical revisions to the CM4S Diagram from the prototype are effective in documenting and constructing the conceptual model and that throughout the use of the revised CM4S Diagram no behavior or activity was encountered that could not be represented or collected with the available shapes and properties. Therefore, no major technical changes to the CM4S Diagram are needed at this point. Utilizing the revised CM4S Diagram to construct this simulation demonstrated that its technical, sanctioning, and documenting capabilities align with its intended purpose and design criteria to advance ABM as a generic analysis tool.
I. The History, Philosophy, and Current Practice of Agent-Based Modeling
What are Complex Systems?

The process of science and engineering requires at least two things. The first is a system. The second is the ability to abstract models of that system into what we call model systems. While the process of modeling a system is discussed later (in particular using an agent-based model to model a system), it is worth examining the properties and characteristics of systems because “systems are as pervasive as the universe in which they exist” [17]. If a goal of science and engineering is to create models that explain and exploit our world, then understanding the properties of the modeled systems is vital. This chapter presents a classification scheme of model systems and their properties to gain insights into factors influencing the complexity of a model system and to develop the appropriate expectations of model systems. With this objective in mind the remainder of this chapter is organized as follows. The first section defines and discusses general properties and characteristics of systems and complexity. The second section discusses the complexity of model systems. The third section presents a model systems landscape that relates a model system classification to the number of components in the model system and the level of understanding of the real system problem. The final section summarizes the key points of this chapter.

2.1 Understanding Systems and Their Complexity

A system has many definitions. The 2008 Webster English Dictionary gives five primary definitions of a system [2]. These include defining a system as a “regularly interacting or interdependent group of items forming a unified whole,” as “a group
of interacting bodies under the influence of related forces,” and as “an organized set of doctrines, ideas, or principles usually intended to explain the arrangement or working of a systematic whole.” For this research, a system is defined as something that transforms inputs into outputs. A car takes gas and produces locomotion, a democratic government takes in their citizens’ needs and produces programs to meet those needs, and a person can take their thoughts and produce a book, a speech, or some other form of communication. The transformation of input into output is a fundamental characteristic of any system and describes what scientists and engineers try to understand.

Within this system definition there are two main types of systems. The first type of system is real systems; the systems that scientists and engineers try to understand. Examples include both natural systems such as the solar system or the digestive system of living animals, and artificial (man-made) systems such as cars and governments. A key characteristic of these real systems is that they are infinitely complex; we cannot completely understand them.

To cope with the infinite complexity of real systems, scientists and engineers use model systems to bound the infinite complexity of the real system into a finite system (a model system) that is easier to comprehend. Models are also systems because they translate input into output. However, not all systems are models since models are truncated abstractions of a real system. It is through these models of real systems that scientific and engineering progress is made. All science and engineering theories and hypotheses that exist are models of real systems and are thus inaccurate representations of the true system.

Real systems and model systems differ in terms of their complexity. While real systems are infinitely complex, the complexity of model systems depends upon the modeler and how much they know about the real system being modeled. Thus, the complexity of a real system is absolute and the complexity of a model system is rel-
ative. Modelers can make a model system as “simple” or “complex” as they want or need. The relative complexity of model systems indicates an inherent relationship between the complexity of the model system and the real system problem it represents. To understand this relationship consider how one tries to solve a problem by constructing a model system of the real system problem.

When trying to solve a problem using a model there are two important considerations. The first is how much detail to include or which components from the real system are required for an effective model system. Including more detail in the model system may produce a more accurate solution, but at the cost of having a more complex model system; since more details directly relates to more complexity in terms of model system size and component interactions. Including less detail means that the model system is inherently simpler, but this can come at the cost of representation accuracy. This is the Size Component of a model system’s complexity. Finding the appropriate level of detail for the model system that best solves a given problem is one driver determining a model system’s complexity.

The second consideration is how much is understood about the real system. Fully understanding the real system for a problem means the developed model system may seem simpler relative to other model systems of less understood real systems. Conversely, understanding less about the real system means the model system developed may seem more complex than model systems of the well understood real systems. Any developed model system of an unexplored real system will inherently seem more complex. This is called the Unexplored Component of a model system’s complexity. Together the number of components (or level of detail) of the model system and the level of understanding of the real system problem drives two components of complexity in a model system: the Size and Unexplored Components of Complexity.

Model system complexity evolves as scientists and engineers learn about the problem of interest and evolve the model system based their understanding of the system.
Considering each model system instance as separate entities means model systems cannot always be directly compared to the older model systems due to changes in assumptions and circumstances. This complexity framework does not contradict scientific tradition, but focuses on how the model system for a particular problem of a real system evolves based on the number of components and the understanding level.

Consider Figure 4 as a notional example of using model systems to understand a problem. The top portion shows the progression of understanding a particular real system problem. This continuous progression is broken into four sections depicting the interplay between the understanding level of the real system (U) and the number of components in the model system (C). The first block represents the starting point in solving the problem. Here understanding and the number of components is relatively low on the scale where the ratio represents a notional scale where 100/100 represents complete understanding and all possible components included in the system. There are only a few components here because not much is known about what influences the real system. A model system at this level is likely not a very formal model, but will be more primitive in nature. To progress, more information on the real system is needed. Block 2 shows a significant increase in the number of components in the model system based on this data collection and a slight increase in the understanding level that often comes from data collection. In Block 2 a formal model exists and can be analyzed. Block 3 shows a significant increase in the understanding level of the real system due to interpreting the results from the model system. This understanding is then utilized the reduce the number of components in the model system to only those that are deemed significant. Block 4 represents this refinement of the model system.

The bottom portion of Figure 4 notionally shows the corresponding complexities of the model systems as a solution to the real system problem progresses. The solid line represents the Unexplored Component and the dashed line represents the Size Component. Initially, the real system example is not well understood. Therefore,
Figure 4: An Example Problem’s Progression and the Complexity of the Model System

- **Problem 1 Progression**
  1. U: 25/100, C: 15/100
  2. U: 32/100, C: 50/100
  3. U: 75/100, C: 50/100
  4. U: 79/100, C: 30/100

- **Interpretation**
  - Data Collection
  - Refinement

- **Complexity vs. Understanding**
  - Unexplored Component
  - Size Component

- **Legend**
  - U: Understanding Level of the Real System
  - C: # of Components in the Model System
the model system is quite complex in the Unexplored Component, but less complex in the Size Component. As the understanding level of the real system increases, the model system becomes less complex in the Unexplored Component domain but more complex in the Size Component as it is systematically refined to a simpler model system.

Figure 4 demonstrates two key things. First, it demonstrates the impact that the understanding level of the real system and the number of components in the model system have on the complexity of the model system. Second, it demonstrates that the complexity of a model system varies based on the perspective that the model system is viewed from. This highlights the importance of precisely describing the complexity of model systems and demonstrates that the complexity of model systems has at least two dimensions (Size and Unexplored) that are inherently interrelated through the process of solving a problem.

This example is only representative of the process of solving a real system problem and certainly does not represent every possible case. For example, a certain model system may fail at increasing understanding no matter how many components are added. In these examples, new model systems may be required. However, under these and other scenarios the fundamental components of the complexity of model systems still apply. Only the rate and scale of progression changes for the given problem with set assumptions and circumstances, but when the problem changes the complexity of previously related model systems cannot be easily compared.

2.2 The Types of Model Systems and Their Complexity

The fundamental goal of model systems is to provide solutions and insights to problems related to real systems. To solve such problems a series of model systems may be constructed with varying levels of complexity with the model system deemed most fit
for use. Since problems partially dictate the model systems constructed, we next examine the types of problems encountered in science and engineering. These problems are then connected to the types model systems typically used to represent them.

In 1948, Weaver identified three types of problems encountered in science: Problems of Simplicity, Disorganized Complexity, and Organized Complexity [115]. Problems of Simplicity are characterized by having only a few variables that are typically linearly related. Problems of Disorganized Complexity are characterized by thousands of variables that when analyzed together create a whole entity. Typically these problems are analyzed using statistical models; one can analyze all of the individual variables and predict an average outcome for the whole system. Problems of Organized Complexity are characterized by a medium number of highly interrelated variables that together produce a new organic whole problem and often exhibit non-linear relationships. Examples of these problems include the flocking behavior of birds or the development of societies.

Each of these problem types identifies several things that aid in characterizing the model systems. The first is that each problem type defines an appropriate number of components. Problems of Simplicity have a few variables and Problems of Disorganized and Organized Complexity have many variables. Each problem type has associated characteristics or behaviors. Problems of Simplicity and Disorganized Complexity have more linear relationships and Problems of Organized Complexity have more non-linear relationships. Weaver created a framework to describe model systems of problems and their associated complexity. His description of these problems are detailed enough to describe the number of components and characteristics of the model systems used to understand these problems. Building on this framework, a landscape of four types of model systems are identified and characterized: Primitive Model Systems (PMS), Simple Model Systems (SMS), Disorganized Complex Model Systems (DCMS), and Organized Complex Model Systems (OCMS) (see Figure 5)
SMSs are model systems characterized as primarily representing problems having linear relationships between a few essential variables. This requires relatively few components and representations of well understood real systems. A prime example of SMSs are basic mathematical models and equations. Note that SMSs are relatively simple in both the Size and Unexplored Components of model system complexity. Simplicity here does not mean the model system is insignificant and that the problem is not worth studying. Here simplicity indicates the success of science and engineering in effectively capturing the fundamental behavior of the real system for a particular problem. As stated by Simon [107], “The goal of science is to make the wonderful and complex understandable and simple-but not less wonderful.”

While simpler model systems, such as SMS, are useful, often more complex model systems are needed to begin understanding the real system problem. Two categories of complex model systems are DCMS and OCMS. This follows Weaver’s lead by breaking complex model systems into two categories, even though complex model systems (or complex systems in most circles) are often discussed as a singular whole. The meaning and associated characteristics of complex model systems differs across disciplines despite similar definitions of complex model systems. A survey of complex model systems today provides the singular definition of a complex model system as being large, composed of many interacting parts, and producing a whole that is greater than the sum of the parts [9, 17, 25, 51, 74, 106]. While ‘large’ and ‘composed of many interactions’ is easy to interpret, different interpretations of ‘the whole greater than the sum of the parts’ (the whole concept) can lead to very different classifications of a model system’s complexity. For example, consider a model system of a car that captures the interaction of the physical pieces of the car at the part level. A mechanical or systems engineer may classify this as a complex model system because it is large, has many interactions, and together the parts of the car produces results that individually they could not obtain (i.e. the whole is greater than the sum of
the parts). Conversely, a complexity theorist likely would not consider this model system complex because they define the whole concept as being the production of transportation behavior. To reconcile these two groups, complex model systems are broken into two categories, DCMS and OCMS.

DCMS are model systems characterized by being large, having many interactions, and producing an abstractable macro-structure that can be analyzed as a whole. Such model systems occur when the real system problem is moderately to well understood and the number of components is relatively large. In general, DCMS are complex in the Size Component and relatively simple in the Unexplored Component of model systems complexity. Traditional DCMS’s include statistical models or discrete event simulations. The majority of engineering model systems today fall into this category.

OCMS are characterized by a large number of components, having many interactions, and producing a surprising or unexpected abstractable macro-structure that is often classified as non-linear. The fundamental difference between DCMS and OCMS is that OCMS produces what is often called emergent behavior; unexpected behavior that results from a low level of understanding concerning the real system problem. Examples of these model systems are found when the real system problem is not well understood and when the number of components is relatively high. In general, OCMS are complex in both the Size and Unexplored Components of the complexity of model systems. Particular examples of OCMS include agent-based models and simulations that capture the evolution and development of societies or model systems describing shoppers’ behavior in a store.

There are two advantages in breaking complex model systems into separate categories. First, it reconciles the two schools of thought concerning complex model systems and establishes the relationship of OCMS and DCMS. Second, it defines the difference between simple and complex in relative terms such as many and few, and low and high. This ensures the framework is based on definitions of simplicity and
complexity that are enduring through time. Since our perception of what is few and what is many is always changing, so too will the definitions of simple and complex. Avoiding definitions that concretely define complexity ensures the future relevance of the framework.

The final model system category is PMS. A PMS arises when a problem is first identified and the model system has not been fully developed or even identified. PMSs exist when there is a low understanding of the real system and the model system has only a few components. In general, PMSs are simple in the Size Component and relatively complex in the Unexplored Component of model system complexity. Often PMSs are only high-level mental abstractions and theories of a real system that are needed to begin studying the problem. An example of a PMS is a child’s understanding that letting go of an object means it falls to the floor. There is no detailed understanding in this real system, just the recognition of a natural phenomena. Together PMS, SMS, DCMS, and OCMS cover the model system’s landscape in terms of complexity.

2.3 Exploring the Landscape of Model Systems and Complexity

Figure 5 shows the landscape for model systems types and their complexity. In this landscape there are two continuous axes. The horizontal axis represents the how well the real system problem is understood (the Unexplored Component of model system complexity). The vertical axis represents the number of components included in the model system for a particular problem (the Size Component of model system complexity). Both of these axes are scaled low to high, where low represents few components and no understanding and high represents many components and complete understanding of the real system problem. Within this two dimensional space are positioned the four types of model systems. PMSs are found in the regions where there
Figure 5: A Landscape of Model Systems and Complexity

- **Number of Components in the Model System**
  - Low
  - High

- **Understanding Level of the Real System**
  - Low
  - High

- **Model Systems**
  - Primitive Model Systems
  - Simple Model Systems
  - Organized Complex Model Systems
  - Disorganized Complex Model Systems
are a lower number of components in the model system and understanding of the real system is low. SMSs are found in the regions where there are a lower number of components in the model systems and understanding levels of the real system is high. DCMs are found in the regions where there are many components and understanding levels are moderate. OCMSs are found in regions where there are many components and understanding levels are low.

This landscape provides several interesting insights beyond showing the relationships between model systems. The complexity of a model system is dependent upon both a controllable and an uncontrollable factor. The controllable factor is the number of components in the model system. Modelers determine the details and assumptions included in the model system. Finding the right balance between details and results is the art of modeling [104]. The uncontrollable factor is the level of understanding. How much is understood about the real system problem depends upon previous work and the current state of knowledge on that problem. By combining these two factors together, one can begin to see how problems are solved and how the appropriate model systems can be used to solve them. When faced with a real system problem the first step is to establish the current level of understanding concerning that real system. Once the level of understanding is fixed, the next step is determining the appropriate level of detail to include. Collectively these determine the model system’s type, level and type of complexity, and ultimately the expectations from the model system.

This landscape can also describe how model systems evolve when solving real system problems. Consider the sample landscape in Figure 6. This landscapes demonstrates using model systems to solve the real system problem of how an ant colony retrieves a food source larger than any single ant can carry. Each arc represents a new type of model system and the various segments of the arcs show the development and advancements of that particular model system. The first model system starts
Figure 6: Sample Model Systems Landscape for Solving the Ant Colony Problem
in the low understanding and low number of components portion of the landscape and is classified as a PMS. Initially, the problem is just recognized and the model system is an informal observation of the food collecting phenomena of ant colonies. As more components are added and data is collected, the understanding of the real system problem improves slightly. At the end of this model system's arc some key components of the real system have been identified such as individual ants and their various classifications (worker, warrior, etc) and it is determined that a formal model is needed to better understand this system problem.

The second model system arc adds components, further increases understanding, and is now classified as a OCMS. Here all of the components of the model system, such as ants randomly walking around and carrying items, helps the modelers understand that some sort of communication is occurring between the ants in order to coordinate the carrying of a large object. Thus, at the end of this model system arc it is determined that a new model system is needed to better replicate the communication of ants. In the third model system, data is collected on the stigmergistic communication of ants via pheromones and these components are included in the model system. Here the ants in the model system exhibit unexpected behavior, but they begin the mimic real ant colony behavior. This represents a significant increase in the understanding of how ants coordinate their activities to collect food for the colony.

The fourth model system arc uses the increased understanding to reduce the number of components in the model system to those that are the most significant. For example, the size distribution of the ants, their age, and the physical attributes of the ants are no longer modeled. This results in a slight improvement in the understanding of the real system. The model system is now classified as a DCMS because there are still many components, the ant colony can now be observed as a whole entity to study, and the understanding level is fairly high. From here the fifth model system is developed that contains fewer components and is classified as a SMS. Note that
this simplification may not increase understanding and may decrease model system
explainability. However, this simplification allows the representation of the key com-
ponents in a few short rules. Mainly, ants randomly travel out from their nest looking
for food and when they encounter food they return to the nest leaving a pheromone
trail that other ants then follow to bring the food back to the colony. Thus, the goal of
science to make the complex simple involves traversing this model system landscape.

Note that the discussion of model systems types do not map to specific modeling
paradigm techniques, such as simulation, statistics, rule-based logic, or mathematical
equations. The discussion simply classified the model system in terms of complexity.
In fact, the ant colony example could be modeled completely using just the Agent-
Based Modeling (ABM) paradigm.

2.4 Conclusion

A hot topic in today’s literature is the study of complex systems. However, under-
standing what complex systems are can be a challenging task. This is especially true
when discussing the differing opinions scientists and engineers have when trying to
determine the complexity of a system. A system translates input into output and
there are two main types of systems. The first are the infinitely complex real systems
that scientists and engineers seek to understand and exploit. The second are finite
abstractions of real system called model systems. Model systems are used by scien-
tists and engineers to solve real system problems. Both types of systems should be
understood when discussing complex systems and it should be recognized that often
the term “complex systems” refers to model systems.

A complexity landscape was introduced based upon the Size and Unknown Com-
ponents of model system complexity. In this framework, the four categories of model
systems (PMSs, SMSs, DCMSs, and OCMSs) are easily positioned. This landscape
helps to describe how real systems problems are progressively solved using model sys-
tems and effectively shows how the connection between model systems, their number of components, and their real system understanding achieves the fundamental goal of science to make the complex simple.
The Emergence of Agent-Based Modeling

Over the years Agent-Based Modeling (ABM) has become a popular tool used to model and understand the many complex, nonlinear systems seen in our world [39]. As a result, many papers geared toward modelers discuss the various aspects and uses of ABM. The topics typically covered include an explanation of ABM, when to use it, how to build it and with what software, how results can be analyzed, research opportunities, and discussions of successful applications of the modeling paradigm. These papers usually include brief discussions about the origins of ABM, discussions that tend to emphasize the diverse applications of ABM as well as how some fundamental properties of ABM were discovered. However, these historical discussions often do not go into much depth about the fundamental theories and fields of inquiry that led ABM’s emergence. Thus, in this chapter I re-examine some of the scientific developments in computers, complexity, and systems thinking that helped lead to the emergence of ABM by shedding new light onto some old theories and connecting them to several key ABM principles of today. This chapter is not a complete account of the field, but instead a historical perspective into ABM and complexity intended to provide a clearer understanding of the field, show the benefits to be gained by understanding the diverse origins of ABM, and hopefully spark further interest into the many other theories and ideas that laid the foundation for the ABM paradigm of today.

1To be published in the Journal of Simulation, 4(2).


3.1 The Beginning: Computers

The true origins of ABM can be traced back hundreds of years to a time when scientists first began discovering and attempting to explain the emergent and complex behavior seen in nonlinear systems. Some of these more familiar discoveries include Adam Smith’s Invisible Hand in Economics, Donald Hebb’s Cell Assembly, and the Blind Watchmaking in Darwinian Evolution [9]. In each of these theories simple individual entities interact with each other to produce new complex phenomena that seemingly emerge from nowhere. In Adam Smith’s theory, this emergent phenomena is called the Invisible Hand, which occurs when each individual tries to maximize their own interests and as a result tend to improve the entire community. Similarly, Donald Hebb’s Cell Assembly Theory posits that individual neurons interacting together form a hierarchy that results in the storage and recall of memories in the human brain. In this case, the emergent phenomena is the memory formed by the relatively simple interactions of individual neurons. Lastly, the emergent phenomena in Darwinian Evolution is that complex and specialized organisms resulted from the interaction of simple organisms and the principles of natural selection.

Although these theories were brilliant for their time, in retrospect, they appear marred by the prevalent scientific philosophy of the time. Newton’s Philosophy, which is still common today, posited that given an approximate knowledge of a system’s initial condition and an understanding of natural law, one can calculate the approximate future behavior of the system [44]. Essentially, this view creates the idea that nature is a linear system reducible into parts that eventually can be put back together to resurrect the whole system. Interestingly, it was widely known at the time that there were many systems where this reductionism approach did not work. These type of systems were called nonlinear because the sum output of the parts did not equal the the output of the whole system. One of the more famous nonlinear systems is the Three Body Problem of classical mechanics, which shows that it is impossi-
ble to mathematically determine the future states of three bodies given the initial conditions.

Despite observing and theorizing about emergent behavior in systems, scientists of the time did not have the tools available to fully study and understand these nonlinear systems. Therefore, it was not until theoretical and technological advances were made that would lead to the invention of the computer that scientists could begin building models of these complex systems to better understand their behavior. Some of the more notable theoretical advances that led to the invention of the computer were first made by Gödel with his famous work in establishing limitations of mathematics [25] and then by Turing in 1936 with his creation of the Turing Machine. The fundamental idea of the theoretical Turing Machine is that it can replicate any mathematical process, which was a big step in showing that machines were capable of representing systems. Furthermore, Turing and Church later developed the Church-Turing Hypothesis which hypothesized that a machine could duplicate not only the functions of mathematics, but also the functions of nature [70]. With these developments, scientists had the theoretical foundation onto which they could begin building machines to try to recreate the nonlinear systems they observed in nature.

Eventually, these machines would move from theoretical ideas to the computers that we are familiar with today. The introduction of the computer into the world has certainly had a huge impact, but its impact in science as more than just a high speed calculator or storage device is often overlooked. When the first computers were introduced, Von Neumann saw them has having the ability to “break the present stalemate created by the failure of the purely analytical approach to nonlinear problems” by giving scientists the ability to heuristically use the computer to develop theories [113]. The heuristic use of computers, as viewed by Von Neumann and Ulam, is very much like the traditional scientific method except that the computer replaces or supplements the experimentation process [113]. By using a computer to replace real
experiments, Von Neumann’s process would first involve making a hypothesis based on information known about the system, building the model in the computer, running the computer experiments, comparing the hypothesis with the results, forming a new hypothesis, and repeating these steps as needed [113]. The essential idea of this empirical method is to understand that the computer serves as a simulation of the real system, which allows more flexibility in collecting data and controlling conditions as well as better control of the timeliness of the results.

3.2 The Synthesis of Natural Systems: Cellular Automata and Complexity

Once computers became established, several different research areas appeared with respect to understanding natural systems. One such area was focused primarily on synthesizing natural systems [66] and was led primarily by the work of Von Neumann and his theory on self-reproducing automata, which are self-operating machines or entities. In a series of lectures, Von Neumann presents a complicated machine that possesses a blueprint of information that controls how the machine acts, including the ability to self-reproduce [113]. This key insight by Von Neumann to focus not on engineering a machine, but on passing information was a precursor to the discovery of DNA which would later inspire and lead to the development of genetic algorithm search processes. However, despite his many brilliant insights, Von Neumann’s machine was very complicated since he believed that a certain level of complexity was required in order for organisms to be capable of life and self-reproduction [70]. Although it is certainly true that organisms are fairly complex, Von Neumann seemed to miss the idea that would later be discovered that global complexity can emerge from simple local rules [44].

With the idea that complexity was needed to produce complex results, reductionism still being the prevalent scientific methodology employed, and perhaps spurred
on by the idea of powerful serial computing capabilities, many scientists began trying to synthesize systems from the top-down. As briefly discussed earlier, the idea of top-down systems analysis is to take the global behavior, decompose it into small pieces, understand those pieces, and then put them back together to reproduce or predict future global behavior. This top-down methodology was primarily employed in the early applications of Artificial Intelligence, where the focus was more on defining the rules of intelligence-looking and creating intelligent solutions rather than the focus being on the structure that creates intelligence [25]. Steeped in the traditional idea that systems are linear, this approach did not prove to be extremely successful in understanding the complex nonlinear systems found in nature [66].

Although Von Neumann believed that complexity was needed to represent complex systems, his colleague Ulam suggested that this self-reproducing machine could be more easily represented using a Cellular Automata (CA) approach [66]. As the name may suggest, CA are self-operating entities that exist in individual cells that are adjacent to one another in a 2-D space like a checkerboard and have the capability to interact with the cells around it. The impact of taking the CA approach was significant for at least two reasons. The first is that CA is a naturally parallel system where each cell can make autonomous decisions simultaneously with other cells in the system [66]. This change from serial to parallel systems was significant because it is widely recognized that many natural systems are parallel [113]. The second reason the CA approach had a significant impact on representing complex systems is that CA systems are composed of many locally controlled cells that together create a global behavior. This CA architecture requires engineering a cell’s logic at the local level in hopes that it will create the desired global behavior [66]. Ultimately, CA would lead to the bottom-up approach now mainly employed by the field of Artificial Life because it is more naturally inclined to produce the same global behavior that is seen to emerge in complex, nonlinear systems.
Eventually Von Neumann and Ulam were able to successfully create a paper-based self-reproducing CA system which was much simpler than Von Neumann’s previous efforts [66]. As a result, some scientists began using CA systems to synthesize and understand complexity and natural systems. Probably the most notable and famous use of CA was Conway’s “Game of Life.” In this CA system, which started out as just a Go Board with pieces representing the cells, only three simple rules were used by each cell to determine whether it would be colored white or black based on the color of cells around it. Using this game, it was found that depending upon the starting configuration, certain shapes or patterns such as the famous glider would emerge and begin to move across the board where it might encounter other shapes and create new ones as if mimicking a very crude form of evolution. After some research, a set of starting patterns were found that would lead to self-reproduction in this very simple system [70]. For more information on the “Game of Life,” to see some of the famous patterns, and to see the game in action the reader can go to http://en.wikipedia.org/wiki/Conway’s_Game_of_Life. However, this discovery that simple rules can lead to complex and unexpected emergent behavior was not an isolated discovery. Many others would later come to the same conclusions using CA systems, including Schelling’s famous work in housing segregation which showed that the many micromotives of individuals can lead to macrobehavior of the entire system [102].

Upon discovering that relatively simply CA systems were capable of producing emergent behavior, scientists started conducting research to further determine the characteristics and properties of these CA systems. One of the first of these scientists was mathematician Wolfram, who published a series of papers in the 1980’s on the properties and potential uses of 2-dimensional CA. In his papers, Wolfram creates four classifications into which different CA systems can be placed based on their long-term behavior [124]. A description of these classifications is found in Table
Table 2: Wolfram’s Cellular Automata Classifications [124]

<table>
<thead>
<tr>
<th>Class</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Evolves to a homogeneous state, changes to the initial state has no impact on final state</td>
</tr>
<tr>
<td>2</td>
<td>Evolves into a set of simple periodic states, changes to the initial state has a finite regional impact on the final state</td>
</tr>
<tr>
<td>3</td>
<td>Evolves into patterns that grow indefinitely, changes to the initial state leads large changes to the final state</td>
</tr>
<tr>
<td>4</td>
<td>Evolves to complex localized patterns that expand and contract with time, changes to the initial state leads to irregular changes to the final state</td>
</tr>
</tbody>
</table>

2. Langton would later take this research further and described that life, or the synthesis of life, exists only in Class 4 systems, which is to say that life and similar complex systems exist between order and complete instability [70]. As a result, it was concluded that in order to create complex systems that exhibit emergent behavior, one must be able to find the right balance between order and instability (termed the “edge of chaos”) or else the system will either collapse on itself or explode indefinitely. It should be pointed out that the “edge of chaos” concept has been an issue of debate. In particular, there are arguments that suggest that it is not well defined and that experiments attempting to reproduce some of the earlier work concerning the “edge of chaos” have failed [77]. Others, such as Czerwinski [28], define nonlinear systems with three regions of behavior with his transition between the Complex behavior region and the Chaos region aligning with the “edge of chaos” concept. Hill et al. [49] describe an ABM of two-sided combat whose behavior demonstrated the stage transitions described in [28]. However, the debate is primarily focused on whether the particular trade-off mechanism used by natural systems is appropriately described by the “edge of chaos” and not whether a trade-off mechanism exists [9]. Thus, until the debate comes to a conclusion, this document will take the stance that the “edge of chaos” represents the idea of a trade-off mechanism that is thought to exist in natural systems.
Armed with these discoveries about synthesizing complex systems and emergent behavior, many scientists in the fields of ecology, biology, economics, and other social sciences began using CA to model systems that were traditionally very hard to study due to their non-linearity [35]. However, as technology improved, the lessons learned in synthesizing these nonlinear systems with CA would eventually lead to models where autonomous agents would inhabit environments free from restriction of their cells. One such models include Reynold’s “boids” which exhibited the flocking behavior of birds [70]. Advanced studies include the influential Epstein and Axtell [35] exposition of CA models involving their Sugarscape model and Illachinski’s [55] ISAAC effort that arguably introduced the military to the use of CA. However, to better understand agents, their origins, and behaviors another important perspective of agents, the analysis of natural systems, should be examined.

3.3 The Analysis of Natural Systems: Cybernetics and Chaos

While Von Neumann was working on his theory of self-reproducing automata and asking, “what makes a complex system,” Wiener and others were developing the field of cybernetics [66] and asking the question, “what do complex systems do,” [6]. Although these two questions are related, each is clearly focused on different aspects of the complexity problem and led to two different, but related, paths toward discovering the nature of complexity, the latter course of inquiry becoming cybernetics. According to Wiener, cybernetics is “the science of control and communication in the animal and the machine” [117] and has its origins in the control of the anti-aircraft firing systems of World War II [66]. Upon fine tuning the controls, scientists found that feedback and sensitivity were very important and began formalizing theories about the control and communications of these systems having feedback. Eventually they would discover that the same principles found in the control of machines were also true for animals,
such as the activity of recognizing and picking up an object [117]. This discovery would lead cybernetics to eventually be defined by Ashby as a “field concerned with understanding complexity and establishing foundations to study and understand it better” [6], which includes the study of both machines and organisms as one system entity.

One of the main tools used in cybernetics to begin building theories about systems was Information Theory as it allowed scientists to think about systems in terms of coordination, regulation, and control. Armed with this new mathematical theory of the time, those studying cybernetics began to develop and describe many theories and properties of complex model systems. One of these discoveries about systems was the importance of feedback on the long-term patterns and properties of systems. In general, complex model systems consist of a large number of tightly coupled pieces that together receive feedback influencing the system’s future behavior. Based on this information, Ashby explains that complex model systems will exhibit different patterns depending upon the type of feedback found in the system. If the feedback is negative (i.e., the Lyapunov Exponent, $\lambda < 0$), then the patterns will become extinct or essentially reach a fixed point. If the feedback is zero ($\lambda = 0$), then the pattern will remain constant or essentially be periodic. Finally, if the feedback is positive ($\lambda > 0$), then the patterns would grow indefinitely and out of control [6].

However, just as Von Neumann failed to make certain observations about complexity, so did the founders of cybernetics fail to consider what would happen if both positive and negative feedback simultaneously existed in a system. It was not until later that Shaw used Information Theory to show that if at least one component of a complex model system has a positive Lyapunov Exponent, and was mixed with other components with varying exponent values, then the system will exhibit chaotic patterns [44]. With Shaw’s discovery that complex model systems can exhibit chaotic behavior, scientists began considering what further impacts Chaos Theory might have.
on understanding systems.

In general, any model system exhibiting chaos will appear to behave randomly while being completely deterministic in nature [25]. However, this does not mean that the real system is completely predictable. As Lorenz was first to discover with his simulation of weather patterns, it is impossible to make long-term predictions of a system with a simulated model because it is infeasible to record all of the initial conditions at the required level of significance [44]. This sensitivity to initial conditions results from the fact that all possible initial conditions are infinite. Therefore, collecting these initial conditions to the required level of significance is impossible without a measurement device capable of collecting an infinite number of infinitely long numbers as well as finding a computer capable of handling all of those infinitely long numbers.

It may seem that this property of chaos has at some level discredited the previously mentioned Church-Turing Hypothesis by suggesting that these types of natural systems cannot be duplicated by a machine. However, there are several other properties of chaos that help those attempting to model and understand these complex systems despite the inability to directly represent them. The first is that chaotic systems have a strange attractor property that keep these aperiodic systems within some definable region [44]. This is obviously good for those studying these complex systems because it limits the region of study into a finite space. The other property of these systems is that they can be generated using a very simple set of rules or equations. By using a small set of rules or equations, and allowing the results to act as a feedback into the system, the complexity of these systems seems to emerge out of nowhere. As one can recall, the same discovery was made in CA when cells with simple rules were allowed to interact dynamically with each other [44]. Therefore, it appears that although natural complex systems cannot be modeled directly, some of the same emergent properties and behavior of these systems can be generated in a
computer using simple rules (i.e., the bottom-up approach) without complete knowledge of the entire real system. Perhaps it is not surprising that the idea that systems can be represented sufficiently with a simpler model, often called a Homomorphic Model, has long been a fundamental concept when studying systems [6].

Whenever discussing the idea that simple rules can be used to model complex systems it is valuable to mention fractals, which are a closely related to and often a fundamental component of Chaos Theory. First named by Mandelbrot, fractals are geometric shapes that regardless of the scale show the same general pattern [72]. The interesting aspect of fractals is that because of their scale-free, self-similar nature they can both fit within a defined space and have an infinite perimeter, which makes them complex and relates them very closely to the effect strange attractors can have on a system. Furthermore, forms of fractals can be observed in nature and, in turn, generated in labs using very simple rules, which shows that they also exhibit the same type of emergent behavior and properties as the previously discussed complex systems [44]. As a result, although fractals, chaos, and complex systems have a lot in common, fractals, due to their physical representation, provide an insightful look into the architecture of complexity.

Fractals are composed of many similar subsystems of infinitely many more similar subsystems of the same shapes, which results in a natural hierarchy and the emergence of other, similar shapes. The architecture of fractals directly shows why reductionism does not work for nonlinear systems. With fractals, a scientist could forever break the fractal into smaller pieces and never be able to measure its perimeter. Another interesting aspect about the architecture of fractals is that they naturally form a hierarchy, which means the properties of hierarchies could possibly be exploited when attempting to model and understand complex systems. For example, the fact that Homomorphic models are effective at modeling complex systems could come from the fact that hierarchical systems are composed of subsystems such that the subsystems
can be represented not as many individual entities but as a single entity [106].

Besides showing that emergent behavior can be explained using chaos, which in turn can be simply represented in a model, there are other properties of chaos which give insight into complex natural systems and ABM. Returning to the idea that it is impossible to satisfactorily collect all of the initial conditions to obtain an exact prediction of a chaotic system, one might ask what would happen if the needed initial conditions were collected, but not to the infinite level of detail? It turns out that such a model would be close for the very short term, but would eventually diverge from the actual system being modeled. This example brings about another property of chaotic systems; they are very sensitive to the initial conditions [25]. This sensitivity property of chaos ultimately leads to unreliable results when comparing a homomorphic model to the actual system. Thus, in general it can be seen that these computer models are unlikely to aid any decision about how to precisely handle the real system. Instead, it can be concluded that these models should be used primarily to provide insights into the general properties of a complex system. Essentially, this methodology of using a computer for inference and insight harps back to Von Neumann’s idea of using a computer to facilitate an experiment with hopes to gain insights about the system rather than using the computer to generate exact results about the future states of the system [113].

The final property of chaos that can give insight into complex natural systems and ABM is that a strange attractor not only limits the state space of the system, but it also causes the system to be aperiodic. In other words, the system with a strange attractor will never return to a previous state, which results in tremendous variety within the system [25]. In 1962, Ashby examined the issue of variety in systems and posited the Law of Requisite Variety, which simply states that the diversity of an environment can be blocked by a diverse system [6]. In essence, Ashby’s law shows that in order to handle a variety of situations, one must have a diverse system capable of
adapting to those various situations. As a result, it is clear that variety is important for natural systems given the diversity of the environment in which they can exist. In fact, it has been seen that entities within an environment will adapt to create or replace any diversity that have been removed, further enforcing the need and importance of diversity [51]. However, it has also been found that too much variety can be counterproductive to a system because it can grow uncontrollably and be unable to maintain improvements [9]. Therefore, it appears that complex natural systems that exhibit emergent behavior need to have the right balance between order and variety, or positive and negative feedback, which is exactly what a strange attractor does in a chaotic system. By keeping the system aperiodic within definable bounds, chaotic systems show that the battle between order and variety is an essential part of complex natural systems. As a result, strange attractors provide systems with the maximum adaptability.

3.4 Towards Today’s ABM: Complex Adaptive Systems

After learning how to synthesize complex systems and discovering some of their properties, the field of Complex Adaptive Systems (CAS), which is commonly referenced as the direct historical roots of ABM, began to take shape. Primarily, the field of CAS draws much of its inspiration from biological systems and is concerned mainly with how complex adaptive behavior emerges in nature from the interaction among autonomous agents [71]. One of the fundamental contributions made to the field of CAS, and in turn ABM, was Holland’s identification of the four properties and three mechanisms that compose all CAS [51]. Essentially, these items have aided in defining and designing ABM as they are known today [71] because Holland takes many of the properties of complex systems discussed earlier and places them into clear categories, allowing for better focus, development, and research.
The first property of CAS discussed by Holland is Aggregation, which essentially states that all CAS can be generalized into subgroups and similar subgroups can be considered and treated the same. As can be seen, this property of CAS directly relates to the hierarchical structure of complex systems discussed earlier. Furthermore, not only did Simon in 1962 discuss this property of complex systems, he also discussed several other hierarchical ideas about the architecture of complex systems [106] that can be related to two other of Holland’s mechanisms of CAS. The first is Tagging, which is the mechanism that classifies agents, allows the agents to recognize each other, and allows easier observation of the system. Essentially, Tagging is nothing more than a means of putting agents into subgroups within some sort of hierarchy. The second mechanism is Building Blocks, which is the idea that simple subgroups can be decomposed from complex systems that in turn can be reused and combined in many different ways to represent patterns. Besides being related to Simon’s discussion of the decomposability of complex systems, this mechanism also reflects the common theme that simplicity can lead to emergent behavior and the theory behind modeling a complex system. Therefore, it can be seen that the elements of Aggregation, Tagging, and Building Blocks can be related back to Simon’s results when studying the architecture of complexity.

Another property of CAS is Non-linearity, which, as previously discussed, is the idea that the whole system output is greater than the sum of the individual component output. In essence, the agents in a CAS collectively to create a result that cannot be attributed back just to the individual agents. Hopefully, it is now clear that not only is this fundamental property the inspiration behind synthesizing and analyzing complex systems, but that non-linearity can also be the result of dynamic feedback and interactions. These causes of non-linearity can be related to two more of Holland’s CAS elements. The first is the property of Flow, which states that agents in CAS communicate and that this communication can change with time. As was
seen in examples using CA, having agents communicate with each other and their environment dynamically can lead to the non-linearity of emergent behavior. Also, within the property of Flow, Holland discusses several interesting effects that can result from changes made to the flow of information such as the Multiplier Effect and the Recycling Effect. In short, the Multiplier Effect occurs when an input gets multiplied many times within a system. An example of the Multiplier Effect is the impact made on many other markets when a person builds a house. Similarly, the Recycling Effect occurs when an input gets recycled within the system and the overall output is increased. An example of the Recycling Effect is when steel is recycled from old cars to make more new cars [51]. Interestingly enough, both of these effects can be directly related back to Information Theory and Cybernetics. The other element that relates to non-linearity is the Internal Model Mechanism, which gives the agents an ability to perceive and make decisions about their environment. It is easy to think of this mechanism as being the rules that an agent follows in the model, such as turning colors based on its surroundings or moving away from obstacles. As with a CA, simple Internal Models can lead to emergent behavior in complex systems. Therefore, the link between these three elements is the essential nature of complex systems: non-linearity.

The final property discussed by Holland is Diversity. Essentially, Holland states that agents in CAS are diverse, which means they do not all act the same way when stimulated with a set of conditions. By having a diverse set of agents, Holland argues that new interactions and adaptations can develop making the overall system more robust. Of course, the idea that variety creates more robust systems relates directly back to Ashby’s Law of Requisite Variety, which in turn relates back to strange attractors and Chaos Theory.
3.5 Conclusion

For the ABM modeler to successfully defend their model and have it be considered worth any more than a new and trendy modeling technique, the modeler needs to have a fundamental understanding of the many scientific theories, principles and ideas that led to ABM and not just an understanding of the ‘how to’ perspective on emergence and ABM. By gaining deeper understandings of the history of ABM, the modeler can better contribute to transforming ABM from a potential modeling revolution [13] to an actual modeling revolution with real life implications. Understanding that ABMs were the result of the lack of human ability to understand nonlinear systems allows the modeler to see where ABM fits in as a research tool. Understanding the role that computers play in ABM shows the importance of understanding the properties of computers and in turn their limitations. Understanding that the fundamental properties of CAS have their origins in many different fields (Computers, CA, Cybernetics, Chaos, etc) gives the modeler the ability to better comprehend and explain their model. Understanding each of these individual fields and how they are interrelated means a modeler can potentially make new discoveries and better analyze their model. Finally, understanding the history of ABM gives the modeler the ability to discern between and develop new ABM approaches.

As it is often the case, examining history can lead to insightful views about the past, present, and the future. It is the hoped that this chapter has shed some light on the origins of ABM as well as the connections between the many fields from which it emerged. Starting with theories about machines, moving onto synthesis and analysis of natural systems, and ending with CAS, it is clear, despite this chapter being primarily focused on complexity, that many fields played an important role in developing the multidisciplinary field of ABM. Therefore, in accordance with the Law of Requisite Variety, it appears wise for those wishing to be successful in ABM to also be well versed in the many disciplines that ABM encompasses. Furthermore, many
insights can be discovered about the present nature of ABM by understanding the theoretical and historical roots that compose the rules-of-thumb (for example Holland’s properties and mechanisms) used in today’s ABM. For example, knowing the theory behind Cybernetics and Chaos Theory could help a modeler in determining the impact that certain rules may have on the system or in trouble shooting why the system is not creating the desired emergent behavior. Finally, it could be postulated that understanding the history of ABM presents one with a better ability to discern between good and bad ABM approaches as well as in developing new ones. In conclusion, this article has provided an abbreviated look into the emergence of ABM with respect to complexity and has made some new connections to today’s ABM that can hopefully serve as a starting point for those interested in understanding the diverse fields that compose ABM.
Simulation and Agent-Based Modeling Validation Philosophy

Since their introduction, computer simulations have become popular in many scientific and engineering disciplines. This is partly due to a computer simulation’s ability to aid in the decision making and understanding of relatively complex and dynamic systems where traditional analytical techniques may fail or be impractical. As a result, the use of simulations can be found in just about every field of study. These fields range anywhere from military applications [30] and meteorology [63] to management science [93], social science [35], nanotechnology [57], and terrorism [96]. What can be inferred from this widespread use is that not only are simulations robust in their application, but they are also practically successful. Due in large part to this robustness and success, simulations are becoming a standard tool found in most analyst’s toolbox. In fact, proof that simulations are becoming more of a generic analysis tool and less of a new technique can be found in the increasing number of published articles that use simulations but do not mention it in the title [65].

However, despite their increasing popularity, a fundamental issue has continued to plague simulations since their inception [82, 108]: is the simulation an accurate representation of the reality being studied? This question is important because typically a simulation’s goal is to represent some abstraction of reality and it is believed that if a simulation does not accomplish this representation, then information gained from

---

2Packaged with the History Chapter and published as a chapter in the Handbook of Research on Discrete Event Simulation Environments: Technologies and Applications (2009).
the simulation is questionable. Therefore, one can understand why answering the question of simulation validity is so important because having an accurate simulation could improve the knowledge about reality without actually observing, experimenting, and dealing with the constraints of reality [113]. Thus, many articles over the years have been devoted to the topic of simulation validity. In particular they tend to focus on some aspect of the following fundamental questions of simulation validity:

- Can simulations represent reality? If not, what can they represent?

- If a simulation cannot or does not represent reality, then is the simulation worth anything?

- How can one show that a simulation is valid? What techniques exist for establishing validity?

- What roles do or should simulations play today?

Given the considerable amount of time and effort spent on simulation validity, a reasonable question to ask is why is simulation validity still haunting simulationists today? In short, the fundamental reason why it is still an issue, and will continue to be one, is that the question of a simulation’s validity is a philosophical question found at the heart of all scientific disciplines [108]. By considering the above questions, one will notice that they closely resemble some typical philosophy of science questions [58]:

- Can scientific theories be taken as true or approximately true statements of what is true in reality?

- What methods, procedures, and practices make scientific theories believable or true?
Therefore, the philosophy of science can shed light on the nature of simulation validity and the nature of simulation itself as it is known today. It is from this fundamental philosophy of science perspective that this chapter provides insights into the fundamental questions of simulation validity, where current practices in simulation validation fit into the general framework of the philosophy of science, and what role simulations play in today’s world.

This chapter has four sections. The first section discusses how the relationship between reality and simulation is flawed such that all simulations do not represent reality. The second section describes what is currently meant by simulation validation in practice. The third section discusses the usefulness of simulations today and how simulations are becoming the epistemological tool of our time. The fourth section discusses the usefulness, roles, and special issues in validating Agent-Based Models.

4.1 Why All Simulations are Invalid

There are many definitions of simulation. For this document a simulation is generically defined as a numerical technique that takes input data and creates output data based upon a model of a system [69] (for this article the distinction between theory and model will not be made, instead the term model will be used to represent them together). In essence, a simulation attempts to show the nature of a model as it changes over time. Therefore, it can be said that a simulation is a representation of a model and not directly a representation of reality. Instead, it is the model’s job to attempt to represent some level of reality in a system. In this case, it would appear that a simulation’s ability to represent reality depends upon the model upon which it is built [30]. Although this relationship between a real system, a model, and a simulation has been described in many different ways [14, 69, 108, 119], a simplified version of the cascading relationship is shown in Figure 7. Note that commonly simulations today are performed by computers because they are much more efficient at
numerical calculations. Therefore, we assume for this document that a simulation is constructed within a computer and that a simulation is a representation of a model which is a representation of a real system (as shown in Figure 7).

Now that the fundamental relationship between a real system, a model, and a simulation have been defined, the implications of this relationship can be examined. As was already discussed, a simulation’s ability to represent reality first hinges on the model’s underlying ability to represent the real system. Therefore, the first step in determining a simulation’s ability to represent reality is to examine the relationship between a real system and a model of that real system. To begin, it must be recognized that a real system is infinite in its input, how it processes the input, and its output, and that any model created must always be finite in nature given our finite analytical abilities [43]. A model can never be as real as the actual system and that instead all that can be hoped for is that the model is at least capable of representing some smaller component of the real system [7]. As a result, it can be said that all models are invalid in the sense that they are not capable of completely representing reality.
The idea that all models are bad is certainly not a new idea. In fact, it is recognized by many people that this is true \cite{7, 43, 108} and there are even articles written which discuss what can be done with some of these bad models to aid in our understanding and decision making \cite{50}. However, if all models are bad at representing a real system and a model is only capable of representing a small portion of that real system, then how will it be known if a model actually represents what happens in the system? In essence, how can we prove that a model is valid at least in representing some subset of a real system?

The basic answer to this question is that a model can never be proven to be a valid representation of reality. This can be shown by examining several different perspectives. The first perspective can be explained using Gödel’s Incompleteness Theorems \cite{42}. Through his theorems, Gödel showed that all propositions from a theory cannot be proven or disproven from the axioms upon which the theory was based. In essence, this means that because every model must be based upon some set of axioms about the real system, there is no way to prove that any model is correct \cite{43}. Another perspective to consider is that there are an infinite number of possible models that can represent any system and it would therefore take an infinite amount of time to show that a particular model is the best representation of reality. Together these perspectives hearken back to one of the fundamental questions found in the philosophy of science; how can a model be trusted as representing reality?

Although a model cannot be proven to be a correct representation of reality, it does not mean that the second fundamental question of the philosophy of science (what methods and procedures make models believable?) has not been thoroughly explored. There actually exist many belief systems developed by famous philosophers that attempt to provide some perspective on this question \cite{58, 60}. For instance, Karl Popper believed that a theory could only be disproved and never proved (Falsificationism), others believe that a model is true if it is an objectively correct reflection
of factual observations (Empiricism) [93]. However, no matter what one believes to be the correct philosophy, the fundamental idea that remains is that all models are invalid and impossible to validate. A shining example of this idea can be seen by the fact that although both are considered geniuses, Einstein still showed that Newton’s model of the world was wrong and therefore it is likely that eventually someone will come up with a new model that seems to fit in better with our current knowledge of reality [58]. Therefore, regardless of how correct a model is believed to be, it is likely that there will always exist another model which is better.

The analysis from the previous paragraphs on the relationship between a real system and a model system have led to the following conjectures about models:

- Models cannot represent an infinite reality and therefore all models are invalid with respect to a complete reality;

- Models can only hope to represent some aspect of reality and be less incomplete;

- There are infinitely many models that could represent some aspect of reality and therefore no model can ever be proven to be the correct representation of any aspect of reality; and

- A better model than the current model is always likely to exist in the future.

From these conjectures, it appears that a simulation’s capability to represent a real system is bleak based purely on the fact that a model is incapable of fully representing reality. However, there is yet another issue with trying to represent a model with a simulation. As seen graphically in Figure 7, another round of translation needs to occur before the transition from the real system to the simulation is complete. At first glance, translating a model into a computer simulation would seem to be relatively straightforward. Unfortunately, this is not case even when programming (verification) issues are left out of the equation. This conclusion generally arises
from to the limitations of the computer. For example, because computers are only capable of finite calculations, often times truncation errors may occur in the computer simulation when translating input into output via the model. Due to these truncation errors alone, widely different results can be obtained from a simulation of a model with slightly different levels of detail. In fact this result is often seen in chaotic systems such as Lorenz’s famous weather simulations, which would later lead to the idea of the Butterfly Effect [44].

Suppose, however infeasible it may be, that advances in computers would make the issues of memory storage and truncation errors obsolete. The next issue in a computer simulation’s ability to represent a model is the computer’s processing speed. Given that computer processing speeds are getting increasingly faster with time, the question about whether a computer can process the necessary information, no matter how large and detailed the model, within an acceptable time seems to be answered by just waiting until technology advances. Unfortunately, there is a conjecture which states that there is a speed limit of any data processing system. This processing speed limit, better known as Bremermann’s Limit [7], is based upon Einstein’s mass-energy relation and the Heisenberg Uncertainty Principle and conjectures that no data processing system whether artificial or living can process more than $2 \times 10^{47}$ bits per second per gram of its mass [22]. From this conjecture, it can be seen that eventually computers will reach a processing limit and that models and the amount of digits processed in a respectable amount of time will dwarf Bremermann’s Limit. Consider for example how long it would take a computer approximately the size ($6 \times 10^{27}$ grams) and age ($10^{10}$ years) of the Earth operating at Bremermann’s Limit to enumerate all of the approximately $10^{120}$ possible move sequences in chess [22] or prove the optimal solution to a 100 city traveling salesman problem ($100!$ or approximately $9.33 \times 10^{157}$ different routes). Given that this super efficient Earth-sized computer would only be able to process approximately $10^{93}$ bits to date, it would take approximately $10^{27}$ and
9.33 \times 10^{64} times longer than the current age of the earth to enumerate all possible combinations for each problem respectively. From the human perspective it would take too long and be too impractical to attempt to solve these problems using brute force.

A computer’s memory and processing limitations impede its ability to accurately represent a model or provide accurate results in a practical amount of time. Thus, simulationists will often build a simulation of a model that incorporates many assumptions, abstractions, distortions, and non-realistic entities that are not in the model [63, 80, 108, 120, 121, 123]. Such examples include breaking continuous functions into discrete functions [63], introducing artificial entities to limit instabilities [63], and creating algorithms which pass information from one abstraction level to another [122]. The limitations of computing makes translating a model into a simulation unlikely to result in a completely valid representation of that model. Simulation building is often considered more of an art than a science because getting a simulation to reasonably represent a model in a computer may require tinkering with the simulation. As a result of this discussion, it can be seen that not only are there an infinite number of models that can represent some aspect of reality, but there is probably also an infinite number of simulations that can represent some aspect of a model.

The following conjectures concern the ability of a computer simulation to represent a model:

- Computers are only capable of finite calculations and finite storage, therefore truncation errors and storage limitations may significantly impact the ability of a computer to represent a model;

- Computers can only process information at Bremermann’s Limit, making it is impossible for them to process large amounts of information about a model in a practical amount of time;
• Representing a model with a computer simulation either requires sacrificing accuracy to get results or sacrificing time to get better accuracy;

• The limitations of computing and the trade off between accuracy and speed means there are many ways to represent a model with a simulation; and

• There are many possible simulations that can represent an aspect of a model meaning it is impossible to have a completely valid simulation of a model.

The conjectures above show why translating a model into a computer simulation is not always easy. Many times a simulationist is simply trying to obtain a simulation that is partially valid but useful to the analytical task at hand.

The following conjectures can be made about simulation validity:

1. A real system is infinite.

2. A model cannot represent an infinite real system and can only hope to be one of any infinite possible representations of some aspect of that real system.

3. A model is an invalid representation of the real system and cannot be proven to be a valid representation of some aspect of the real system.

4. There are many possible computer simulations that can represent a model and each computer simulation has trade offs between the accuracy of the results and time it takes to obtain those results.

5. A simulation cannot be said to be a completely valid representation of a model.

6. A computer simulation is an invalid representation of a complete real system and at the very best cannot be proven to be a valid representation of some aspect of a real system.

The above conjectures lay out the issues with a simulation’s ability to represent reality. Furthermore, it can be seen why simulation validation continues to be a major issue.
If simulations cannot be proven to be valid and are generally invalid representations of a complete real system, then what value to they serve? However, this question is not the primary source of research in simulation validation. Instead, much of the focus still remains on how one can validate a simulation. Given the conjecture that all simulations are invalid, or impossible to prove to be valid, what do all of these authors mean when discussing simulation validation?

4.2 What Does Simulation Validation Really Mean in Practice?

Although all simulations are invalid with respect to a real system, there is still a significant amount of literature that shows how a simulation can be validated. It may initially appear that those involved in simulation building are unaware of the downfalls facing simulation’s ability to represent reality, but this is not the case [14, 69]. So what are these articles and books discussing when they are focused on simulation validation? Insight into what practitioner’s really mean by simulation validation can be seen from the definitions of validation:

“Validation is the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study” [40, 69]

“Model Validation is substantiating that a model within its domain of applicability, behaves with satisfying accuracy consistent with the models and simulations objectives...” [12, 101]

“Validation is concerned with building the right model. It is utilized to determine that a model is an accurate representation of the real system. Validation is usually achieved through the calibration of the model, an iterative process of comparing the model to actual system behavior and using the discrepancies between the two, and the insights gained, to improve the model. This process is repeated until the model accuracy is judged to be acceptable.” [14]
“Validation is the process of determining the manner in which and degree to which a model and its data is an accurate representation of the real world from the perspective of the intended uses of the model and the subjective confidence that should be placed on this assessment.” [29]

These definitions clearly indicate that, in practice, simulation validation takes on a subjective meaning. Instead of validation being the process of determining the accuracy of a simulation to represent a real system, the clause “with respect to some objectives” provides the caveat that a simulation can never accurately and completely represent a real system. By adding this caveat, simulationists have inserted some hope that a model is capable of being classified as valid for a particular application or purpose. This clause gives validation a completely new meaning. No longer is the issue of absolute validity the problem, the problem is now proving the partial validity, or relative validity, of a simulation model with respect to some set of objectives.

Many articles have been published which provide a different perspective of this relative validity problem. One of these perspectives is to attempt to evaluate the relative validity of the simulation by treating it not as a representation of a model/theory but as a miniature scientific theory and then to use the principles from the philosophy of science to aid in proving/disproving its validity [60, 61]. As first introduced by Naylor and Finger in 1967 [82], many authors have thoroughly examined the many beliefs from the philosophy of science and have related them to simulation validity [15, 37, 59, 60, 93, 103]. Often these philosophical works provide insightful views into simulation validation because the philosophy of science has been actively discussing the validity of theories long before the inception of simulation [60].

Although the introduction of scientific philosophy has certainly provided new perspectives and points of view on the subject of validity to simulationists [60, 103], the philosophy of science has brought with it more questions than answers. There are several key reasons for this. The first is that every belief system in the philosophy of science has both advantages and disadvantages. For example, simulation validation is
often favorably compared to Falsificationism because it states that a simulation can only be proved false and that in order to consider a simulation scientific, it must first undergo scrutiny to attempt to prove that it is false [59]. However, under this belief system it is difficult to determine whether a model was rejected based on testing and hypothesis errors or whether the model is actually false [58]. Another reason of concern for using the philosophy of science is that, as discussed earlier, it is impossible to prove that a model/theory is valid. Therefore, using the philosophy of science to aid in simulation validity is more applicable in providing insights into the fundamental philosophical questions stemming from validation as well as potential validation frameworks than in actually proving the validity of a simulation.

Another perspective on simulation validation considers methods and procedures to aid simulationists in proving the relative validity of their simulation given the assumption that it can be proven. This assumption is by no means a radical one. If one defines the objective of the simulation to include the fact that it cannot completely represent the real system then it is possible for a simulation to meet the needs of a well defined objective and therefore have relative validity. A plethora of techniques have been developed within systematic frameworks to aid simulationists in validating their models [12, 101]. Even the idea of validation itself has been reduced to many different types of validation such as replicative, predictive, structural, and operational validity [36, 125].

A lot of research effort summarizes and defines how one can go about validating a simulation given some objectives. A common theme in research is subjectivity. Whether the validation technique is quantitative, pseudo-quantitative, or qualitative, each technique has its advantages, disadvantages, and is subjective to the evaluator. This subjectivity is found in beliefs of system similarity, levels of statistical significance, and reasonableness of representations or results.
Since no technique can prove the relative validity of a simulation, the fundamental question remains: what does simulation validation really mean in practice? In practice simulationists attempt to validate the simulation according to some objective, which cannot be systematically proven true. So what is really occurring when a simulationist is trying to validate their model according to some objective? Simulation validation, in practice, is really the process of persuading the evaluators to believe that the simulation is valid with respect to the study objective; how well can the simulationist "sell" the simulation’s validity using the appropriate validation techniques that best appeals to the evaluator’s sense of accuracy and correctness.

The idea that simulation validation in practice is really the process of selling the simulation to the evaluator may not appeal to scientists, engineers, and simulationists, but there is a fair amount of evidence supporting this conclusion. First of all, any simulation book or article focused on validation frequently stresses the importance of knowing the evaluator’s expectations and getting the evaluator to buy into the credibility of the simulation [14]. Some works even explicitly state that one must sell the simulation to the evaluator [69]. Others indicate that validating a simulation is similar to getting a judicial court system to believe in the validity of the simulation [37]. Generally those practicing simulation understand that validation is more about getting the evaluators to believe in the simulation’s validity and less about getting a truly valid simulation (which is impossible). Therefore, simulation validation is not completely removed from society and other social influences. In fact, it appears that simulation validation in practice requires the simulationist to actively interact with the community of evaluators to persuade that community to accept the simulation as correct. As a result, some have argued that simulation validation in practice is similar to how any social group makes a decision [93].

In trying to determine what simulation validation really means in practice, several fundamental points have been made:
• In practice, a simulation is validated based on some objective and not on being a true representation of the real system;

• All of the techniques developed to prove the validity of a simulation in practice are subjective to the evaluator and therefore cannot systematically prove the relative validity of the simulation; and

• Validating a simulation in practice depends upon how well the simulationist sells the validity of the simulation by using the appropriate validation techniques that best appeals to the evaluator’s sense of accuracy and correctness. Simulation validation in practice is susceptible to the social influences permeating the society within which the simulation exists.

Simulation validation in practice seems to have little to do with actual validation, where validation is the process of ensuring that a simulation accurately represents reality. If simulation validation in practice is more concerned with getting approval from evaluators and peers of a community relative to some overall objective for the simulation, then simulation validation, in practice, is really the process of getting the simulation sanctioned [119]. As a result, simulationists should consider adopting the term simulation sanctioning instead of simulation validation since sanctioning implies a better sense of what is actually occurring while validation implies that the truth is actually being conveyed in the simulation. However, it is unlikely that this transition will occur given the fact that simulation validation today is mainly concerned with getting evaluators to buy into the results of the simulation, the current paradigm in simulation has been established, and saying a simulation is valid sounds much better to a seller than does saying a simulation is sanctioned. This brings up an interesting dilemma for simulationists because if simulations cannot represent reality, then what good are they?
4.3 What Good are Simulations?

Since simulations in practice are sanctioned and not validated, the next logical question to ask is if simulations are incapable of representing reality and therefore are incapable of providing true results with respect to the system, then what good are simulations? This question is posed fully understanding that simulation is growing in popularity based on noting its continuing widespread use in practice, the number of commercial simulation software packages, and the number of academic publications using simulation. These are indications that simulation is robust enough to be considered useful and indeed successful [65].

Even though all simulations are invalid with respect to an absolute real system, a simulation can, at some abstraction level, get relatively “close” to representing a small portion of an absolute real system. For example, a simulation of a manufacturing system may come very close to representing the outcome of a process. Simulations get results close enough to true to become practically useful [62, 64]. Simulations are useful because they are often capable of providing reliable results without having to be a true representation of a real system [123].

The reliability and predictability of simulation results depends upon how well the real system is understood and studied; a well understood system provides better underlying theories that form the foundation of the simulation. A simulationist using a simulation to represent a well understood system, really just takes advantage of the processing power and memory of the computer as a computational device. The simulation is likely to produce reliable results because it is simply being used to express the calculations that result from a well-established theory. A typical example of this can be found in a queuing simulation, which has been extensively studied and has well-established theories [56]. For such well understood systems, the simulation can provide predictive power.

As less is understood about the real system, a simulation becomes research in-
strument acting as a mediator between theory and the real system [80]; the theories
about the real system are not developed enough to provide reliable predictions about
future states of that system. The simulation in this case is a research tool in the same
sense, for example, that a microscope is a research tool [63]. While the microscope can
provide insight into the real system, it does not directly reflect the nature of the real
system and cannot directly provide reliable predictions about the real system. Instead
the microscope provides a two-dimensional image of a dynamic three-dimensional real
system. The microscope mediates between existing theories about the real system and
the real system itself. Experiments are designed and hypotheses tested based on in-
formation gained from the microscope. Similarly, a mediating simulation is capable
of providing insight into the real system and the theory on which it was built without
being a completely valid representation of that real system. Although only formally
recognized recently [80], the idea that computers can be used to facilitate experiments
and mediate between reality and theory has existed for a long time. In the early years
of computing, John von Neumann and Stanislaw Ulam espoused the heuristic use of
the computer, which is an alteration of the scientific method to replace real system
experimentation with experiments performed within a computer [113].

An interesting aspect of mediating simulations is the interplay between the so-
phistication of the simulation and the real system. As the real system becomes better
understood, the simulation of that real system is improved thereby allowing new in-
sights into the real system. Examples of this mediating role of simulation is seen in
many different fields. One example can be found in the field of nanotechnology where
without computer simulations to aid in the complex and difficult experiments, certain
advances in nanotechnology would not be possible [57]. Another example is found in a
complex production system, where the simulation provides insights into how the real
system might behave under different operating circumstances. In the world of ABM,
“toy models” such as ISAAC (Irreducible Semi-Autonomous Adaptive Combat) have
been used as to explore and potentially exploit behavior that emerges in battlefield scenarios [55]. A final example can be seen in the field of physics where some “think of sciences as a stool with three legs - theory, simulation, and experimentation - each helping us to understand and interpret the others” [68].

While experiments on real systems are often preferred, experiments on sanctioned simulations of real system often benefit. In simulations, errors are controllable and in fact repeatable. Simulation experiments offer greater control than in real systems since simulation parameters are fixed. Thus, the simulation can mediate theories about the real system.

The use of simulation to mediate despite the use of the real system abstraction has led to black-box evaluations of the simulation. Generally, scientists prefer models to structurally resemble the real system. A simulation has many assumptions and falsification and will generally not structurally resemble the real system. However, a recent trend is to assess how well the simulation translates realistic input into realistic output, as the fundamental benchmark in determining the usefulness of that simulation [62, 63]. Indeed, many of the technical validation techniques proposed today emphasize the use of this black-box paradigm [11, 14, 69, 101].

In general, this shift away from white-box evaluation (structural representation) towards black-box evaluation (output is all that matters) [21] can lead to several interesting conclusions. The first is that this shift indicates the general acceptance of the idea found in Simon’s *The Sciences of the Artificial*. Essentially, Simon argues that artificial systems (ones man-made such as simulations) are useful because it is not necessary to understand the complete inner workings of a real system due to the fact that there are many possible inner workings of an artificial system that could produce the same or similar results [107]. One way to think about this is to consider whether the differences between the inner workings of a digital clock and an analog clock really matter if they both provide the current time for the user. Another conclusion
that can be drawn by the shift towards black-box evaluation is that simulations are beginning to catch up and pass the theoretical understanding of the systems that they are being built to represent. The question now becomes, what possible usefulness can a simulation be when little to nothing is known about the underlying principles of the real system of interest?

At first glance, the usefulness of a simulation for a system that is not well-understood appears nonexistent. However, it is from this lack of underlying theory and understanding of the real system that the usefulness of this type of simulation becomes evident. Consider what a simulationist would encounter if asked to build a simulation of a poorly understood system. The first steps they would probably be to observe the system and then attempt to generate the system behavior within the simulation. The simulation that generates reasonable system behavior can then be exercise to generate insights. Systematic input changes tot he simulation can generate resulting system outputs from which the simulationist might derive explanatory theories.

This ability of a simulation to act as a medium in which new theories about a real system are generated is the third role of simulation, which is that of a theory or knowledge generator. Using a simulation as a medium to generate new theories and ideas about the real system is no different from using pencil and paper or simply developing mental simulations about the real system [7]. One could observe a system and attempt to test the implications of a theory by using pencil and paper or develop elaborate thought experiments as those made famous by Einstein. Alternatively, one could use a simulation to test whether a theory is capable of representing the real system. Examples of simulations being used for this role abound in the new simulation paradigm of Agent-Based Modeling (ABM), where simulationists are typically trying to understand problems that are difficult for us to grasp due to the large amount of dispersed information and the high number of interactions that occur in these systems.
(more about the special case that ABM simulations present to the world of simulation is discussed in the next section) [35, 76, 83].

There are several advantages for using simulations as generators, the most important of which is the ability of a simulation to create “dirty” theories of systems where the simplicity of the real system eludes our grasp. Typically, scientific theories are often idealized for a particular case and do not allow for deviations from these idealizations. It could be thought that these idealizations are partly the result or desire of humans to make simplifications and elegant equations to represent the complex world. However, simulations allow theorists to build a representation of a system within a simulation using less-elegant mechanisms, such as ad-hoc tinkering, engineering, and the addition of logic controls such as if-then statements [64, 122]. This flexibility, means that as more problems of interest fall into the realm of Organized Complexity (medium number of variables that organize into a highly interrelated organic whole) [115] the use of simulation to generate new “dirty” theories about the real system will increase. These new systems of interest are irreducible and typically hard to understand to the point that often simulationists are surprised about the results obtained from these systems [25].

Simulationists using simulations as theory generators should not consider themselves disconnected from science, because there are no implications of using a simulation as a generator to the philosophy of science [41]. Instead, they should ascribe to the practices, rigor, and roles taken on by scientists to make true progress in the practitioner’s field of interest. Furthermore, as simulationists and scientists continue to push the limits of simulations beyond that of the current knowledge of some system of interest, it can be seen why some researchers consider simulation the epistemological engine of our time [54].

Figure 8 connects the roles of a simulation with the simulationist’s knowledge level of the system of interest. Figure 8 shows that when much is known about the system,
the simulation tends to take on more of a predictor role. As less is known about the real system, the simulation begins to take on the role of being a mediator between the system and the theory. Finally, when the understanding of the system is low, the simulation can act as a generator of potential theories about the nature of the system. The use of simulations as generators is of particular interest in this research. Since ABM directly fits into this role, the deeper issues involved with generator simulations is discussed in the next section.

4.4 What Good is ABM?

Despite the fact that any simulation paradigm can be used in a generator role, probably the most popular paradigm used today is ABM. Emerging from Cellular Automata, Cybernetics, Chaos, Complexity, and Complex Adaptive Systems, ABM helps to understand and explore complex, nonlinear systems where typically independent and autonomous entities interact together to form a new emergent whole. An example of such a system is the flocking behavior of birds [70]. Although each bird is independent, somehow they interact together to form a flock, and seemingly without any leading entity, manage to stay in tight formations. With this in mind, simulationists using ABM attempt to discover the rules embedded in these individual entities that could lead to the emergent behavior and eventually attempt to make in-
ferences about future states of these systems based on the simulations they developed. ABM is often used as a generator of hypotheses for these type of complex systems.

ABM is often used to investigate problems where no micro-level theory exists (it is not known how the individual entities operate) and where it is often very difficult to measure and collect macro-level data (the emergent behavior) from a real system and compare it to the data generated from the simulation [13, 70, 76, 83]. Ultimately, this characteristic of these complex problems means that the current traditional and accepted quantitative sanctioning techniques which promote risk avoidance based on performance and comparing outputs are not applicable [105], since too little is known about these systems. From this statement, several interesting conclusions arise about ABM and generator simulations.

First, since ABM is a relatively new paradigm, either accepted techniques to sanction these simulations have not yet been created to match the current sanctioning paradigm or a new sanctioning paradigm with new sanctioning techniques is needed specifically for generator simulations. In order for the first statement to be the case, the underlying theory behind the real system being studied by these generator simulations needs to be known to the point that the simulation is no longer a generator but instead is a predictor; the current sanctioning paradigm has a majority of its interest in predictability and has created sanctioning techniques that are mainly focused on this predictability. Therefore, it is impossible for generator and ABM simulations, by their nature, to fit into the current predictive sanctioning paradigm. Furthermore, if a ABM simulation ever became predictable it would no longer be a generator simulation and traditional quantitative sanctioning techniques could be used. As a result, simulationists using ABM today as a generator should shift their focus to creating a new generator sanctioning paradigm and developing new techniques to match. Attempting to create this new sanctioning paradigm will certainly not be easy, but is necessary if ABM simulations as generators are to become acceptable. Only after this
sanctioning paradigm has been created can both simulationists and evaluators come to firm conclusions about whether a generator simulation should be sanctioned as a scientific research tool or an engineering alternative analysis tool.

Until the complex systems simulated by ABM are well understood, ABM simulations should be viewed as a research tool capable of providing insight into the real system and identifying what needs to be understood about the real system in order to develop a theory of the real system [13]. For the knowledge gained from ABM simulations to be viewed as reliable, a new sanctioning paradigm is needed based on precision and understanding as it relates to the more traditional methods employed by scientists [105]. As this new sanctioning paradigm expands, new sanctioning techniques can be created which provide value to the generator simulationist such that the real system is understood to the point that generator simulation paradigms, such as ABM, can evolve into mediator or predictor simulations.

4.5 Conclusion

As simulation continues to grow in popularity in scientific and engineering communities, it is valuable to reflect upon the theories and issues that serve as the foundation for simulation. This chapter added context and reconciled the practices of simulation with the theory of simulation. In particular, this chapter built a framework describing the crucial relationships that exist between simulation as a medium and real systems. A fundamental conclusion is that simulations are not really validated in practice but are instead sanctioned, which brings into question the usefulness of simulations in general.

A simulation does not need to be a complete representation of some aspect of a real system to be useful. Therefore, a general framework was developed that related the role of simulation based on the level of understanding of the real system of interest. In this continuous framework, a simulation can take on the role of generator, mediator,
or predictor as the level of understanding increases with regards to the real system. Of particular interest are generators and how the epitome of this new use of simulation as a theory generator and research tool has emerged in the form of ABM, because ABM aids in the understanding of complex, nonlinear systems. However, because ABM is a relatively new simulation paradigm, the current sanctioning practices are not applicable. Therefore, the ABM community needs to develop a new sanctioning paradigm for generator simulations, focused on understanding and accuracy of less understood real system.
A Survey of Agent-Based Modeling Practices

5.1 Introduction

Emerging from the fields of Complexity, Chaos, Cybernetics, Cellular Automata, and Computers, the Agent-Based Modeling (ABM) simulation paradigm began gaining popularity in the 1990's and represents a departure from the more classical simulation approaches such as discrete-event simulation. A primary reason for the popularity of ABM and its departure from other simulation paradigms is that ABM can simulate and help examine organized complex systems (OCS). This means the ABM paradigm can represent large systems consisting of many subsystem interactions. These systems are typically characterized as being unpredictable, decentralized, and nearly decomposable. Although computer simulation as an analytical tool has been around since the advent of computers, the ability of the ABM paradigm to simulate complex systems has moved to fields such as social science and economics where for the first time they can utilize simulation to analytically explore these complex systems at a level of detail that was difficult or impossible previously.

5.1.1 What is Holding Back ABM?

Its characteristics and abilities have led some to claim that ABM represents a revolution in modeling and simulation. However, this statement is based primarily on the potential of ABM rather than results [13]. One reason for the lack of meaningful

---

results from ABM studies, in general, is due to the type of complex systems that ABM is used to simulate and explore. Traditionally these types of systems are very difficult to analyze given their non-linear behavior and size [25]. Nevertheless, there is no reason why analyzing these complex systems using ABM should not eventually produce meaningful, model-based results. Systems that are large and difficult can be understood. History gives many examples of problems that seemed nearly impossible to solve, but when given the proper tools scientists found solutions. For example, at one point we did not understand why an apple fell to the ground from a tree. Newton, and others, were able to develop theories and tools that helped them not only explain but also predict the behavior of the falling apple. By extension meaningful results regarding these complex systems will be gained when the proper tools and models are in place, and ABM is, at least for the moment, the most suitable tool for analyzing these types of the complex systems.

ABM as a modeling technique and paradigm is still a work in progress. This statement is generally supported by the relatively recent development and popularity of the paradigm, its departure from traditional simulation paradigms, and the “new to simulation” fields that are using ABM to study OCS. Whenever a new tool or technique emerges, time is needed to flush out the details of its application, capability, and limitations. For ABM, researchers must determine what simulation techniques/philosophies are appropriate and what new techniques/philosophies are needed specifically for ABM. Since ABM is being used in fields that traditionally have not used simulation, it will take some time for these researchers to hone their simulation skills and to effectively develop appropriate analytic models for their domain.

Two key things are needed to mature the ABM paradigm. First, techniques, philosophies, and methods need to be developed specifically for ABM and distinguished from other simulation techniques, philosophies, and methods. A fair amount
of research in this area has already been done (for a few examples see [10, 18, 34, 35, 51, 71, 76, 83]). Second, the teaching of ABM techniques, philosophies, and methods must improve so those using ABM can build effective models. These key things are independent of the specific scientific domain of interest.

5.1.2 What is the Current State of ABM?

Specifically what do ABM researchers need to focus on? What specific problems exist in the ABM paradigm domain that are keeping ABM from reaching its full potential? To help answer this question, we present a comprehensive review of the state of ABM to determine research directions, needs, and opportunities. We surveyed 279 published articles in which agent-based models were built and used for analysis. The survey helps to describe the last 10 years of the field’s development as well as its current state of the art.

The remainder of the chapter is divided into four sections. Section 2 discusses the general survey methodology and provides justification for the categorization strategy employed. Section 3 discusses the results from the survey. Section 4 discusses the implications the survey results have on identifying the research opportunities in the ABM paradigm. Finally, Section 5 summarizes and concludes the chapter.

5.2 Methodology

Throughout the survey process every attempt was made to obtain ABM articles in an unbiased manner. However, the ABM literature is vast and covers many scientific domains of interest. Thus, it is quite likely that this survey will miss some domains using ABM. However, the issues and challenges associated with ABM are likely quite domain independent. Thus, our survey provides a starting point in determining the state of the art and the common research challenges.
5.2.1 Collection of the Sample

The survey methodology involved obtaining a large sample of published works where the authors built some agent-based models and reported their analytical findings. There are several advantages to this approach. The first is that it more accurately reflects what simulationists are concerned with, the techniques they are using, and what the publication outlets and reviewers deem acceptable practice. This type of information directly represents the main thoughts, feelings, and techniques used by those constructing “acceptable” agent-based models. This approach can also help capture trends by tracking when the works were published. Finally, this approach is less subjective to author opinion and bias. A good representative sample of works can be collected and a well defined categorization scheme can be implemented to objectively capture the techniques used by the simulationists. Focus on articles discussing specific techniques or methods would yield limited information on ABM trends, issues, and challenges.

The works included in this survey discuss development of an agent-based model, the results they produced, were published by a peer-reviewed outlet, and were published within an approximate 10 year time frame (January 1, 1998 to July 20, 2008). Using this criteria, 279 works were obtain from a variety of outlets. The primary source used to collect the samples was OhioLINK’s Electronic Journal Center. OhioLINK is a consortium of 89 Ohio colleges and universities as well as the State of Ohio Library. Specifically, the Electronic Journal Center (EJC) is one service of OhioLINK that was established in 1998 and is a online full-text collection of over 7,750 journals from many different disciplines [1]. Using the EJC, the keyword search “agent-based” provided the links to the works obtained.

In addition to the EJC, other sources were used to obtain samples from fields that are not as well represented within the EJC. One such source is the Journal of Artificial Societies and Social Simulation (JASSS). JASSS is one of the few journals dedicated
to society and social computer simulations. All JASSS articles that met the search criteria were also included in the survey sample. One field that was noticeably missing in the original EJC sample was military applications of ABM. To incorporate some of the military work involving ABM, Master’s Theses from the Naval Postgraduate School in Monterey, CA and the Air Force Institute of Technology in Dayton, OH were also included into the survey. Although not published journal articles, they are reviewed and deemed to be acceptable enough to award students with a Master Degree. These works not only meet the survey criteria but often provide much more detail about their models since they are not restricted by page limits. Appropriate articles from the Winter Simulation Conference (WSC) were included to capture ongoing work in simulation since WSC is one of the primary simulation conferences in the world. Note, WSC articles are also reviewed before being published in the proceedings. Finally, duplicate works were excluded. Duplicate works included papers using a common model but for differing purposes. Removing duplicates helped avoid skewing the survey results.

Altogether, a total of 279 samples were collected from 92 unique publication outlets from the 10 year sampling period. The distribution of the number of articles per year is shown in Figure 9. In general, this distribution appears appropriate; it reflects what is intuitively expected. Since ABM has become more popular over time, there should be an increasing trend in the number of articles per year. Clearly the sample reflects this. Thus, this sample appears to be a relatively decent representation of the population. Note 2008 data only includes article available before July 20, 2008. Projections of final 2008 number are not made since the survey focus is not on projecting ABM growth but on capturing ABM trends and research challenges.

The breakdown of the number of articles per publication outlet is shown in Figure 10. Figure 10 indicates that the majority of the samples come from publication outlets with four or less articles in the sample. This means that many different
Figure 9: Number of Articles per Year in the Sample

![Histogram of Article Publication Year](image)

Figure 10: Articles per Publication Outlet in the Sample

![Histogram of Top Number of Articles per Journal](image)
outlets are accepting ABM articles, a nice trend for the field. Figure 10 also shows that the sample represents a wide variety of topics including military applications, biology, economic, social science, business, complexity theory, and simulation. This topic diversity in the range of outlets further supports our claim that this sample is a meaningful representation of the ABM field. A complete list of the 279 works included in this sample is found in the Appendix.

5.2.2 Categorization and Data Collection Strategy

With a reasonable sample of literature, the next step was determining an appropriate categorization and data collection strategy that would give insight into the progression and current state of ABM. Some data is standard. For example, the author(s), publication outlet, general topic, and year of publication were easily recorded from each sample. These data do not provide the insight needed into many of the techniques, methods, and philosophies of the field. Therefore, other data were employed.

5.2.2.1 Software

Software data included whether general software packages or native languages were used to realize the agent-based model. If authors mentioned a software package, for example the ABM was built using Java or C++, the software package name used was recorded. If the authors said they programmed their model directly, for example by using Java or C++, then the programming language was recorded. This type of information gives insight into the popularity of particular software packages and helps to determine how modelers are creating their agent-based models.

5.2.2.2 Field of Study

Accurate information regarding the author’s domain or field of study helps infer whether different fields of study have different ABM practices. Each article was deemed from a field of study such as economics, social science, military, biology and
public policy. The field applied was judged to best describe the topic of the model. Naturally, there were instances where a model could exist in multiple fields of study; only the best describing field was used. This categorization strategy gives insight into the differences and similarities between and within domains that are using ABM.

5.2.2.3 Reference to the Complete Model

Science and engineering is the process of developing models/theories of real systems for particular purposes. ABM is just a technique that aids science and engineering in gaining insight into the real world and how the real world behaves. As with any science or engineering model, ABM results must be independently replicated for the results to be considered scientifically valuable. Each article was reviewed to see if they provided some reference to the complete model, or at least some way to obtain a complete description of the model.

5.2.2.4 Validation Technique(s)

To gain insight into a real system, a model must be an accurate representation of that real system. Since all models are incorrect representations of reality [7, 108], the emphasis of simulation validation is ensuring the model is an appropriate representation of the real system of interest for a given set of objectives [12, 14, 69, 101, 125].

There are two aspects when considering the validation of ABM, or any simulation model. The first aspect is the piece of the simulation model being validated. There are many pieces of a simulation. For simplicity this survey examined validation of the two most basic pieces: the conceptual model and the simulation output. There are many different representations of how to build a good simulation model [12, 14, 69, 101, 125]. Figure 11 shows a simplified simulation development process. Notice there are two rounds of validation, each validating different parts of the simulation. The first round validates the conceptual model. The conceptual model is the abstracted model of the real system, it relies upon known system theories, it drives
model development, and dictates the variety of assumptions required in any model abstraction process [14, 79, 98, 97, 101]. The conceptual model forms the foundation of an ABM model; an invalid conceptual model indicates the model may not be an appropriate representation of reality. The second round validates results of the simulation against results from the real system. For a model to be completely valid, it must be validated both conceptually and operationally. For the survey, each article in the sample was examined to check whether conceptual and operational validation of the model occurred.

The second aspect is the techniques used to validate each piece of the simulation model. Within the simulation domain are many different validation techniques (for several examples see [12]). This survey partitions these techniques into statistical and non-statistical techniques. Statistical techniques are defined as techniques that use formal statistical hypothesis tests to check the validity of some piece of the model. Non-statistical techniques are techniques that do not use formal statistical hypothesis
tests, but rely instead on more qualitative assessments such as expert opinion. For the survey, each piece (conceptual and operational) of a model was examined to determine if a statistical technique, a non-statistical technique, some mixture, or no validation technique was performed on that piece of the model.

All validation techniques involve the evaluator subjectivity in determining the simulation is a valid representation of reality. Some say that validation, which implies truth, should really be called sanctioning [119], which implies more of a process in which evaluators agree that a model is close enough for useful purpose. For the survey, an article was reviewed and data recorded when a validation technique was performed within the framework established. No measure was assigned pertaining to the quality of the validation process as such a measure would be inherently biased based on the author’s like or dislike of the technique.

5.2.2.5 Purpose of the Simulation

Defining the purpose of the model can be subjective and ambiguous. However, knowing a model's purpose allows conjectures regarding how different ABM techniques and model philosophies support differing ABM purposes. To reduce subjectivity and ambiguity another framework describing the different purposes simulation is established. This framework is based upon the level of understanding associated with the system of interest and more recent research concerning the role that simulation and modeling plays in modern science [65].

Figure 12 relates the three defined roles or purposes of the simulation (Generator, Mediator, and Predictor) with the level of understanding known about the real system. When the system is well understood the simulation is called a Predictor; it is used like a calculator to provide clear and concise predictions about the system. An example of this could be a simple queuing system or a very well understood assembly line activity. As less is understood about the real system, the simulation moves toward a Mediator role. In this role the simulation provides insight into the system, but
is not a complete representation of how that system actually behaves. When using a simulation as a Mediator, theories can be put forth and tested and the simulation can be subsequently improved. For more about simulations and models as Mediators see [80]. When little is known about the real system of interest, the simulation takes on the role of a Generator; the simulation acts as a generator of hypotheses and theories about how the real system behaves. As a Generator, a simulation serves the same purpose as other mediums where theories and hypotheses are proposed [7].

These three roles are not mutually exclusive. Figure 12 shows that these roles exist on a continuum. Thus, simulations can exist between two different roles. For this survey, the model was recorded into the dominant role. For example, if a model was 40% mediator and 60% generator, the model was classified as a Generator. For the survey, the following definitions were used:

- A Generator is a simulation where little is known about the system of interest and it is used primarily to determine if a given conceptual model/theory is capable of generating observed behavior of the system.

- A Mediator is a simulation where the system is moderately understood and it is used primarily to establish the capability of the conceptual model to represent the system and to then gain some insight into the system’s characteristics and behaviors.
5.3 Results

This section provides the main results compiled from the survey. A further analysis of each topic is discussed in the next section.

5.3.1 Software

Figure 13 displays a summary of the software packages or programming languages used. Overall, 68 unique software packages or programming languages were referenced with many of them (22.6%) being referenced less than three times overall. It is clear that both ABM specific software packages and generic programming languages are being used and that the most popular software packages are ones that are public domain. In fact, only AnyLogic and Matlab are commercial packages listed in Figure
13. A striking result is that 104 articles (37.3%) did not provide any details on what package or programming language was used to construct and execute the simulation.

5.3.2 Fields of Study

The breakdown of the articles by domain is displayed in Figure 14. In the sample the three most popular fields of study using ABM are economics, social science, and biology. In general, the fields of study in the survey show ABM being used by fields whose systems involve many interacting autonomous entities. This supports the fundamental belief that ABM is good at modeling and analyzing these systems. Although the majority of the fields of study in the survey are not traditional scientific disciplines, there are still a significant number of traditional disciplines using ABM. This supports the wide appeal of ABM as a methodology.

5.3.3 Purpose of the Simulation

In terms of model purpose, 111 (39.8%) of the models surveyed were Generators, 168 (60.2%) were Mediators, and 0 (0.0%) were Predictors. This confirms the belief that agent-based models are used primarily to gain insight into the system of interest. It is interesting to note an almost equal number of generators and mediators.
Simulationists are using agent-based models to generate theories about a system's behaviors and as a mediating instrument to capture certain behaviors of the system and to characterize how the system may behave under certain scenarios. This general characteristic is relatively constant over the last 10 years, as shown in Figure 15.

There does appear to be differing model purposes by domain of interest. As shown in Figure 16, the only domains where the majority of the models were generators are social science (66.2%) and economics (65.8%). The domains with the lowest number of generator models are business (0.0%), public policy (4.3%), and the military (5.6%).
These differences are reasonable. Social science and economics are still new and in the process of developing theories about how their systems of interest operate. Thus, using agent-based models as generators allows them to explore hypotheses and ideas that are not easily manipulated using other theory generating techniques. Conversely, it makes sense that business, public policy, and the military are more interested in mediating models that can be used to gain insight into the system in order to exploit some aspect of the system’s characteristics.

5.3.4 Reference to the Complete Model

Only 44 (15.8%) of the articles surveyed gave a reference for the reader to access or replicate the model. This indicates that the majority of the authors, publication outlets, and reviewers did not deem it necessary to allow independent access to the models. This trend appears consistently over the last 10 years as shown in Figure 17.

Figure 18 depicts model references by domain. The domains with the most references to the complete model are social science (26.5%) and economics (19.0%), while those with the least are the military (2.8%) and business (0.0%). These results are again reasonable. Social science and economics are scientific fields interested in theory
development, so they are more likely to provide their model to others. The military and business fields are secretive (e.g., security, competitive advantage) so less they are less willing to share their complete model.

The defined purpose of the simulation generally has little impact on the whether the complete model is referenced. Figure 19 indicates that only 21.6% of generator models and only 11.9% of mediator models gave references to the complete model. It may seem that this is a significant difference, but the correlation between purpose and domain better explains the difference depicted in Figure 19.
5.3.5 Validation (Not Considering Technique)

We next focus on whether a model was conceptually validated, operationally validated, conceptually and operationally validated, or not validated at all. Figure 20 indicates that 29% of the models were not validated, 17% only had their conceptual model validated, 19% only operationally validated their model, and 35% both conceptually and operationally validated their model. A reasonable position is that a model is only validated, or sanctioned, when it is both conceptually and operationally validated. In this case, at least 65% of the models in the survey were incompletely validated. This is alarming since most outlets for scientific publication insist on some level of model validation.

Emphasis on model validation does seem to be changing. As seen in Figure 21, the percentage of models not completely validated is declining. The difference between the beginning and the end of the 10 year period is distinct and shows that the field is improving in terms of completely validating their models. However, between 2005 and 2008 the number of articles that both conceptually and operationally validate their model remains relatively constant and averages to just under 43%.

Breaking down model validation by domain reveals that some fields are more con-
cerned with validation than others. As shown in Figure 22, the fields with the highest percentage of completely validated models are ecology (77.8%) and biology (70.0%) and the fields with the lowest percentage of validated models are military (16.7%), economics (20.3%), and social science (27.9%). A reasonable conjecture regarding the differences is their scientific tradition. However, while military, economics, and social science are relatively new fields and not as well connected to the classical scientific tradition the military has a long history of using computer simulation and their issues with simulation validation are well documented [30]. Thus, this aspect of validation
for military agent-based models is somewhat surprising.

There does appear to be a relationship between the purpose of the simulation and whether it has validation efforts. In Figure 23, 11.7% of generator models were completely validated while 51.2% of mediator models were completely validated. Since generator models are based on systems that are less understood, these models would be harder to validate because there is less information available about the system. Conversely, more “validation activities” should occur for mediator models because more information is known about the system being modeled.

5.3.6 Validation Techniques

Of the models validated in some way, 0.5% used only statistical validation techniques, 95.0% used only non-statistical validation techniques and 4.5% used a combination of statistical and non-statistical validation techniques. Thus, it appears that in ABM the primarily validation techniques employed are expert opinion and qualitative comparisons of behaviors. The statistical validation techniques often taught in basic simulation courses are not as popular. This result may be due in part to difficulties capturing statistics from the ABM simulation as well as the system. Furthermore, it can be more challenging to analyze due to nonlinear output. When examining
validation techniques by year, as shown in Figure 24, a trend shows a decreasing number of models not using any validation technique. For the most part the use of non-statistical validation techniques are being employed.

Figure 25 breaks out the validation technique used by field and again the most commonly used are non-statistical validation techniques, but with no strong relationship between validation technique and the field of study. Figure 26 displays validation techniques by model purpose. The most popular validation techniques are non-statistical techniques while for mediator models there is a slightly higher use of
statistical validation techniques; this is expected since more is understood about the real system.

5.4 Discussion

These survey results provide information about the development and current state of ABM. From this data research directions, needs, and opportunities are identified. While there are many different implications these results may have depending upon a researcher’s interest, in this section just some of the most important implications of these results on developing and maturing the field of ABM are discussed.

5.4.1 Software and Verification

With 68 unique software packages or programming languages used to build and execute the surveyed simulations it is clear that there are many ways that a model can be represented in a computer simulation. This variety is likely attributed to the background of the simulationist, programmer, or non-programmer. Thus, no software package or programming language will likely ever become the standard in building agent-based models. This means that tools developed to aid in constructing
and documenting agent-based models as well as teaching techniques, should not be specifically geared towards a software package or programming language. Instead, development and documentation tools and teaching techniques should be independent of software and programming languages. Also, they should be focus on the issues involved in the construction and execution of an agent-based model while emphasizing the fundamental methods and issues of building a simulation.

There are also implications for reviewers and evaluators of agent-based models when there is a lack of common software packages. ABM evaluators must understand basic simulation programming techniques. Since agent-based models can address a wide range of problems it is essential that researchers provide sufficient discussion of their application for the evaluator to assess the realization of the system abstraction into the simulation. Publication outlets, and their reviewers, do not seem to be requiring such detail.

5.4.2 Addressing the Many Fields of Study or Creating a New One

ABM is connecting diverse fields. The fields of biology, business, ecology, economics, the military, public policy, social science, and traffic, among others, all use ABM. These diverse fields are trying to understand complex systems and are using ABM as a common tool. If it holds that complex systems generally have similar properties, then these diverse fields should be actively sharing insights about their complex systems. Naturally, ABM publications promote sharing. However, after reviewing the surveyed articles it is clear that each field has developed their own ABM terminology to describe techniques, applications, and results, have their own ABM standards, and their own ABM philosophies.

Observing the growth of multiple ABM theories points to a fundamental need for ABM to be studied as an independent discipline, a subset of simulation, such that standard ABM techniques, practices, philosophies, and methodologies can be
established. A common ABM theory means all disciplines could speak the same ABM language and develop techniques and models based on proven and accepted approaches. To gage the depth of this division one only needs to realize that even the definition of an agent is not clear, depends upon who is the author, and can vary widely. Bringing together the field of ABM will result in a better analysis tool for every field of study. It is important that some standards be established when considering that some believe that ABM and simulation is becoming the epistemological engine of our time [65].

5.4.3 Redefining the Meaning of Results by Purpose

Those considering ABM, as a simulationist or evaluator, must re-consider how they define the results of the model. ABM naysayers argue the models do not produce results while this survey found otherwise. This contradiction is likely the result of different definitions of "results" and the various expectations associated with simulation. There is a general belief that simulations should produce clear predictions and estimations of system behaviors to be considered successful. This expectation fits well with the long standing ability of discrete-event simulations, but it does not necessarily fit well with the kind of systems that an agent-based models simulate.

It could be conjectured that the majority of simulations developed throughout history are of fairly well understood systems and that their general purpose was to provide some estimation or prediction about the behaviors of a particular system. In other words, the majority of past simulations are held up to predictor expectations. But from the survey it is clear that agent-based models are being used as mediators (60.2%) and generators (39.8%). This survey finds that ABM is living up to its potential as a revolution in modeling and simulation by extending the applicability of simulation to new fields of studies and complex system abstractions. As the use of ABM expands, and complex systems become more understood, it is conjectured that
eventually the ability of an agent-based model to provide predictions will improve as more is understood about the complex systems they are simulating.

5.4.4 Providing a Reference to the Complete Model

A low value of 15.8% of the surveyed articles provided a reference to the complete model. If the reader or evaluator does not have access to a complete model, how can they verify the results produced? In other sciences, such shortfalls would give the article little or no chance of publication. This prompts the question of why such limited model descriptions are allowed?

There are probably several main reasons why references to the complete model in ABM are not considered an important part of the article. The first is that simulationists may not be willing or able, due to propriety issues, to provide their complete model to the public. This is unlikely to change. However, a potential remedy to this problem is to require authors to provide enough of a description of the model such that independent evaluators can reconstruct the model. Such detail could allow others to quickly review the logic and execution of the model and reproduce it in their choice of software package or programming language. For this to occur some model describing tools or diagrams from the fields of systems engineering or computer science may help by providing rich and complete descriptions of these models sufficient for independent evaluation and replication.

An ABM developmental tool offers other benefits to the ABM community. First, methods could help enforce good simulation programming practices by emphasizing particular aspects of the model that must be described. This information aids those building the model and provides evaluators a way to evaluate and validate every model. The tool could also be used as a teaching aid to help researchers build more effective models. This could mean more effective ABM employment resulting in improved understanding of modern complex systems.
5.4.5 *Complete* Validation is Required for *Every* Model

It could be argued that validation is one of the most important aspects of model building because it is the only means that provides some evidence that a model can be used for a particular purpose. Without validation a model cannot be said to be representative of anything real. However, 65% of the surveyed articles were not completely validated. This is a practice that is not acceptable in other sciences and should no longer be acceptable in ABM practice and in the publications associated with ABM. One of the other potential reasons why models are not being completely validated is that the authors may consider that just conceptually or operationally validating their model is good enough. This survey found that overall 36% (the majority) of the articles only validated one aspect of the model. Our position is that both conceptual and operational validity are required for complete validity.

If a model is only conceptually validated, then it unknown if that model will produce correct output results. For example, consider a scientific experiment. In this experiment a hypothesis about some macro-level behavior is made based on some conceptual model that appears valid based on what is known about the system. However, when the experiment is performed the hypothesis is rejected because it did not properly predict the macro-level behavior. The operational-level hypothesis based on the conceptual model is invalid even though the conceptual model of the hypothesis appears valid prior to the experiment.

Conversely, if a model is only operationally validated, then it is unknown whether that model is based on any appropriate representation of reality. For example, consider a simulation of a standard single server queuing model where the objective is to achieve the theoretical performance [56]. Typical performance measures are the average time in the queue or system throughput. The standard approach to build this simulation is to observe the real system, measure arrival rates, measure server processing times, and then build a realistic representation of the system using some discrete
event simulation packages. It would be expected for the simulation to behave like the real system and therefore the simulation would be both conceptually and operationally validated. Now consider using an ABM simulation with reproducing bugs to model the queue. In this simulation, the bugs move about their environment looking for food and reproduce with other bugs, much like those of the Sugarscape agent-based model [35]. Key measures about the bugs, such as lifespan and birthrate, are mapped to the goal performance measures of the single server queuing model. Parameters concerning the bugs and their environment are adjusted using some algorithm until the simulation's performance measures match the expected queuing performance measures. The bug model is then deemed useful for queuing analysis, even though it is unlikely that anyone would accept this conceptual construct as a queuing system construct. Although this is an extreme example, without complete validation the effectiveness and ability of the model to represent a system is unknown.

The importance of validation in science and simulation cannot be overstated. Not enough scientists using ABM as an analysis tool are properly validating and documenting their model. It is absolutely essential that all models be completely validated and that the articles associated with them clearly document the validation techniques used and their results. Likewise, publication outlets and reviewers should be stringent in their validation requirements in order to produce better models and to advance not only their field of interest but also the field of ABM.

5.4.6 Statistical vs. Non-Statistical Validation Techniques

It is surprising that so few of the articles surveyed used statistical validation techniques given the widespread use of statistical validation techniques in other simulation paradigms. Two conjectured reasons are that ABM is used to simulate systems whose output are not conducive to statistical analysis and that those building and evaluating these agent-based models have validation criteria that differs from validation criteria
used in other simulation paradigms. The surveyed models are primarily being used in non-traditional simulation fields that may not be as influenced by the statistical validation standard of other simulation paradigms. Further, the surveyed models generally reflect using a simulation for generator and mediator purposes, as opposed to predictor purposes that are more focused on matching system outputs and therefore more conducive to statistical analysis.

The popularity of non-statistical validation techniques in ABM, highlights potential research opportunities. First, the effectiveness of statistical validation techniques for ABM needs to be further explored and evaluated. Second, there is a need for new statistical validation and data collection techniques specifically for ABM. Unlike non-statistical techniques, which requires evaluator knowledge of the domain modeled, statistical techniques do not require domain knowledge about the system or field for the evaluator to judge the validity of the model. Finally, the field must develop more standardized and comprehensive non-statistical validation techniques specifically for ABM. Fundamentally, by developing and discussing the use of both statistical and non-statistical validation techniques for ABM, the resulting models will be validated to a higher standard, yielding more robust models that can advance the knowledge of the system being modeled and the field of ABM.

5.5 Conclusion

It has been conjectured that ABM is an immature method and that standard practices promoting effective ABM modeling are neither clearly established nor accepted. This survey supports that conjecture. The lack of maturity and standard practices in the ABM field is reflected by the lack of models that were completely validated, the lack of references to the complete model, and what is accepted as publishable. A remedy is that techniques, philosophies, and methods need to be adopted from other simulation paradigms, or developed specifically for ABM, and that these techniques,
philosophies, and methods need to be taught to those using ABM such that they can build more effective models. Based on a survey of 279 published articles this article portrayed the state-of-the-art in ABM and identified key research directions.

Six specific research directions, needs, and opportunities for ABM were identified in the survey. First, development and documentation tools for ABM need to be independent of software and published articles should detail the software package or programming language used in to build and execute the simulation. Second, since ABM is a departure from other simulation paradigms, it needs to be studied as an independent discipline yet also as a subset of the simulation discipline. From this standard techniques, practices, philosophies, and methodologies are needed to extend ABM as a functional analysis tool. Third, since ABM is used for different purposes, simulationists should have different expectations for ABM. Fourth, articles need sufficient information about the model so other researchers can independently develop and evaluate the effectiveness of these models. The fifth, and most significant, conclusion reached from the survey is that reviewers and publication outlets must require that the model be completely validated and documented in the article. Finally, both statistical and non-statistical validation techniques specifically for ABM need to be developed and conveyed effectively to those building these models.

These six research directions, needs, and opportunities represent just some of the things needed to mature and help establish standard practices for ABM. If ABM is to reach its full potential as a modeling and simulation paradigm, these fundamental opportunities must be addressed. This is especially true as simulation takes on new roles and begins to extend our limited ability to comprehend and mentally analyze modern complex systems. By establishing clear research goals and standards, the field of ABM will continue to mature and progress and every field exploring complex systems is better equipped to understand, evaluate, and predict these systems through the exploitation of more appropriate and effective agent-based models.
II. The Development of the Conceptual Model for Simulation Diagram
The Sanctioning Solution Concept and Why It is Required

Based upon the six identified needs of the ABM survey and the philosophical and historical foundations developed in Part I, it is clear that there are opportunities to develop a diagramming technique that will have a significant impact on the way agent-based models are constructed, validated, and reported. The major reason for considering a diagramming technique as a potential solution concept is that diagrams are graphical languages that can describe entities and processes, provide documentation, communicate ideas, and emphasize important aspects of the artifacts being described [23]. These general capabilities accurately describe the kinds of needs identified in Part I. A sufficient diagramming technique for ABM has the potential to meet many of the identified needs and thereby satisfy the objective of helping to advance ABM as an analysis tool. However, developing a new diagramming technique, particularly one suited for ABM, should be based on more specific requirements. In this chapter I further investigate/re-investigate the process of constructing simulations, examine the appropriate emphasis when sanctioning agent-based models, and consider what types of systems are being simulated using ABM. By examining these topics, a series of detailed requirements for a diagramming technique are identified.


6.1 Exploring Requirements

6.1.1 How Simulations are Developed

There are a wide variety of step-by-step instructions, and associated figures, that authors propose as a guide for the simulationist to create a good simulation and conduct a simulation study. Some guides are fairly linear in nature with iterative steps to ensure that the model meets the objectives of the project and is properly sanctioned [14, 69]. Other guides are more complicated in structure, emphasizing the need for continuous sanctioning and they attempt to convey the complex task that is required if one wishes to build a good simulation [12, 101]. Even though all of these simulation development processes differ, they contain similar fundamental elements of simulation building. Based upon these similarities, a simplified simulation development process is shown in Figure 27. This simplified process emphasizes the role of sanctioning (both conceptual and operational) in simulation building.

The first step, as shown in Figure 27, is to formulate the problem and set the objectives to be achieved by the simulation study. In this step, arguably the most important step, the overall idea is to determine whether the simulation paradigm is a good fit for the problem, determine the proper abstraction level for the simulation, and clearly define the expectations of the simulation project. The second step is to build the conceptual model. There is currently no clear and concise definition of what exactly is a conceptual model [97]. However, a conceptual model can be described as “the process of abstracting a model from a real or proposed system” [97] and it is typically a mathematical, logical, and/or verbal representation of the real system of interest [101]. A conceptual model is the abstracted model intended to mimic the desired behaviors of the real system. This is often referred to as the “art” portion of the model building process [14, 79]. This second step relies heavily upon any known system theories driving the model development and the variety of assumptions required in the model abstraction process. Sanctioning in this step is
Upon successfully sanctioning the conceptual model, the next part of building a simulation is to translate the conceptual model into a computerized model. This is where verification issues come into consideration [12, 14, 69, 101]. Once the computerized model is verified, the final step of this process is to run the simulation and obtain results which can then be used to gain insights into the real system. However, before the simulation can reasonably be used as a proxy of the real system, it must first undergo another round of sanctioning called Operational Sanctioning. For the simulation to be operationally sanctioned, its output behavior must sufficiently match the real system’s output behavior at the desired abstraction level and intended purpose [101]. Although this simulation development process appears simple, it highlights the major steps involved in building a good simulation.
### 6.1.2 Sanctioning Emphasis

Some key aspects about simulation building with respect to sanctioning arise by examining the simulation development process. First, there are two types of sanctioning processes that occur during the building of any simulation. Second, the two sanctioning processes occur at different points in the simulation development process and as a result have very different objectives. While Conceptual Sanctioning occurs at the beginning of development process, and is concerned with how well the conceptual model matches the theory and assumptions of the real system, Operational Sanctioning occurs at the end of the simulation building process and is concerned with how well the output of the simulation matches the output of the real system.

One can compare Conceptual versus Operational Sanctioning with white-box versus black-box evaluation, respectively. Both Conceptual Sanctioning and white-box evaluation place more emphasis on understanding the details of how the system works while both Operational Sanctioning and black-box evaluation place more emphasis on matching results (performance) rather than the internal structure of the system. This comparison highlights where Conceptual and Operational Sanctioning fit into the simulation framework. If the major emphasis of a simulation study is on performance (Operational Sanctioning), and not on understanding the theories of the system, then one could assume that the simulation is based upon a well understood system (otherwise the simulation study would have failed the Conceptual Sanctioning phase). Conversely, if the emphasis of a simulation study is on understanding the system (Conceptual Sanctioning) and not on how well the outputs match reality, then one could assume the simulation is based upon a less understood system. It is unlikely that a simulation would be sanctioned based purely on performance when the entire simulation is built upon a soft and assumption-laden conceptual model (this has been called the Base of Sand Problem [30] in a military modeling context). The framework in Figure 8 on page 73 is modified to relate the appropriate emphasis
Figure 28: Relationship between System Understanding and Simulation Sanctioning Emphasis

![Diagram](image)

of sanctioning of a simulation based upon how much is understood about the real system and is shown in Figure 28. It still holds that regardless of the level of system understanding, any simulation should undergo both types of sanctioning to provide reasonable confidence in ABM results. Figure 28 indicates which sanctioning type is most crucial in the simulation development process with specific emphasis dictated by the criticality of the ABM component.

To re-enforce this idea, consider the emphasis within a typical scientific article. Even though the literature review is always a crucial part of any article, the time spent critiquing and testing past work depends upon how well the foundational issues concerning the system of interest are generally understood. It is unlikely that a physicist using Newton’s Laws for an experiment will spend much time on the conceptual validity of Newton’s Laws if they are already accepted as applicable for the system, especially if the experimental results contributes to the field. As Hooker states with respect to the analysis of algorithms [52], and as extended to ABM for my case, emphasis should be to the degree dictated by the critically of the ABM component.

A conclusion from Figure 28 is that simulations of poorly understood systems require more Conceptual Sanctioning emphasis than Operational Sanctioning emphasis. The previous survey of current ABM practices found that most systems being simulated using the ABM paradigm are not well understood. Therefore, it appears that
the natural sanctioning emphasis for the ABM community is Conceptual Sanctioning. Thus, the next section further explores the process of conceptual modeling and what Conceptual Sanctioning techniques currently exist.

### 6.1.3 Conceptual Modeling

Despite the fact that conceptual modeling is probably one of the most important aspects of building an effective simulation, there is actually little literature that specifically address how to build a conceptual model [98, 97], particularly for an agent-based model. There seems to be two main reasons for this. The first is that the guideline for building a conceptual model come in literature that discuss how to generally build a model; a conceptual model is traditionally an assumed part of the model building process. A conceptual model defines what is going to be modeled and how it is going to be modeled. While there is often no direct discussion of conceptual modeling, there is a fair amount of literature that discusses the essence of conceptual modeling as part of how one should build a model [98]. These model building articles also touch on the second point of why there are not many of articles discussing conceptual modeling; conceptual modeling is more of an art than a science [14, 79, 98, 97]. Conceptual modeling cannot be detailed into a step-by-step process that guarantees some particular result. Instead, all that can be offered to those attempting to build a conceptual model are general guidelines, such as keeping things simple and creating analogies to other developed structures [79]. The fundamental conclusions that can be drawn about conceptual modeling is that “conceptualizing a model requires system knowledge, engineering judgment, and model-building tools” [95]. Thus, examining the process of conceptual modeling reveals some important considerations for a new ABM sanctioning technique: the need to understand and have system knowledge, engineering judgment, and access to model-building tools.

In a 2005 article, Sargent identifies two focuses that together encompass the idea
of Conceptual Sanctioning [101]. The first part is ensuring that “the theories and assumptions underlying the conceptual model are correct” by using mathematical analyzes and statistical methods as well as ensuring that the theories are properly applied [101]. Sargent suggests using empirical sanctioning techniques to ensure that all assumptions and theories of the conceptual model match that of the real system. Thus, there are certain aspects of any conceptual model that are quantitative in nature and therefore the more traditional Operational Sanctioning techniques focused on quantitative sanctioning can be used.

The next part of Conceptual Sanctioning is ensuring that “the model’s representation of the problem entity and the model’s structure, logic, and mathematical and causal relationships are ‘reasonable’ for the intended purpose of the model” which are primarily evaluated using face validation and program traces [101]. Face validation and program traces involve subject matter experts to examine all of the logic of the conceptual model, typically via some flowchart or other graphical device, in order to sanction the conceptual model. Note that program traces often require the simulation code, which emphasizes the need for both conceptual and operational sanctioning. This means there is a qualitative aspect of Conceptual Sanctioning that requires an expert to subjectively review the structure and logic of the conceptual model.

To summarize, three key ideas regarding Conceptual Sanctioning and conceptual modeling were identified. First, Conceptual Sanctioning should be emphasized when little is understood about the system. Second, conceptual modeling is not a straightforward process but requires system knowledge, engineering judgment, and model-building tools. Finally, Conceptual Sanctioning involves both quantitative evaluation and a qualitative evaluation, with each of these types of evaluations having long-standing sanctioning techniques. The next section section considers ABM application in more detail.
6.1.4 Difficulties in Modeling Organized Complex Systems

The ABM paradigm is a relatively new simulation paradigm that emerged out of need to understand Organized Complexity Problems, or problems with a medium number of highly interrelated variables causing the system to be highly nonlinear [115]. With these new types of problems in mind, consider the following general conditions that make modeling a system easier and in turn make developing a conceptual model easier [95]:

- Physical laws are available that pertain to the system;
- A pictorial or graphical representation can be made of the system; and
- The uncertainty in system inputs, components, and outputs is quantifiable.

With Organized Complexity Systems the above conditions do not always hold. Although general progress is being made in defining laws and theories governing Organized Complex Systems (such as found in the fields of Chaos, Cybernetics, and Complexity), these types of systems are not yet so well understood that there are solid physical laws available from which to build a model. Since this is a new type of problem, there are not many pictorial or graphical representations one can use to represent a Organized Complex System. Attempting to use traditional two dimensional graphs with arcs and nodes to represent nonlinear, complex, and highly interrelated states can get cumbersome, even infeasible, and too often increases confusion and complexity, the opposite intention of having the graphical representation [46, 47]. Attempting to quantify the uncertainty of inputs, components, and outputs can be a challenging task simply because Organized Complex Systems are not well understood. Not having the tools or methods to build a conceptual model of these systems makes sanctioning of the entire simulation extremely difficult and can immediately bring into question how well the simulationist understands the system being modeled.
6.1.5 Re-visiting Peer-Level Sanctioning

Simulations are becoming the epistemological engine of our time because they can be used to represent complex nonlinear systems and show the implications of those systems. As an epistemological engine, simulations have almost become the theories of the systems they are intended to mimic and therefore should be treated the same way as other scientific theories. This means the simulation must survive peer scrutiny and be reproducible for scientific progress to be made.

This conclusion that ABM simulations of real systems need to be independently peer evaluated is not new. Several articles discuss the need for independent evaluation or peer sanctioning [8, 10, 118]. Hindrances to independent peer sanctioning include ambiguity in published papers, gaps in published descriptions or unclear descriptions, and technical difficulties related to simulation [8]. Since these simulation-based theories cannot be represented in simple equations or descriptions, attempts to describe them completely in words in a journal paper is either impossible or extremely difficult. This may be the reason why only 15.8% of the surveyed articles gave reference to the complete model in their article. However, this should not be a surprise given the difficulty of representing the complex non-linearity of these systems and the fact that journals are probably not willing to publish an article long enough to completely describe a simulation.

One natural solution to this problem is for the authors to provide their peers access to their simulation model; this, however, raises several issues. First of all obtaining a copy of the simulation model may be difficult due to proprietary issues. Even if the simulation is obtained, there are many simulation languages and packages that the simulationist could have used. If the evaluator is not familiar with the simulation language of the simulation, then understanding the simulation can be a problem. Even if the evaluator is familiar with the simulation language and can run the simulation independently, the evaluator may be able answer the “how” questions
of the simulation but they still cannot effectively answer the “why” questions. For example, the evaluator knows how an entity A behaves when it encounters an entity B, but the evaluator does not know why or with what justification entity A’s behavior was defined.

To evaluate simulation, an evaluator must understand the micro-level details of the simulation. This means the evaluator must have simulation and system domain knowledge. The evaluator must abstract their domain knowledge into a simulation paradigm and use this abstraction to understand the conceptual model for the simulation under evaluation.

A diagramming technique describing an ABM simulation could be effective in this capacity. A diagramming technique could provide information on such things as initial conditions, all logic associated with the micro-level entities and justification behind the logic, how the entities interact and the justification, variables, parameters, probability distributions, random number generators, and terminating conditions. The diagramming technique could also be independently evaluated by experts of varied simulation experience levels and provide both the hows and justification at various levels of detail.

6.1.6 Summary of the Key Requirements

The following are key requirements for a diagramming technique for ABM:

1. Aids in learning and conveying system knowledge

2. Incorporates proper engineering judgment

3. Aids in translating the conceptual model into a computerized model

4. Emphasizes the development and sanctioning of the micro-level behaviors

5. Displays the theories and assumptions built into the model for quantitative analysis
6. Conveys the conceptual model’s logic and structure for qualitative analysis

7. Completely represents the simulation so it can be reproduced by independent evaluators

8. Provides justification for all structures and actions in the simulation

9. Reviewable by evaluators of varied simulation and domain expertise levels

10. Can represent Disorganized and Organized Complex Systems

Complete systematic design maps requirements to the identified needs to keep the design focused and to maintain design traceability. The mapping of these design requirements to the needs in ABM are shown in Figure 29. In Figure 29 each line type is connected to a need and to the requirements derived from that need. The goal of this figure is to demonstrate exactly how the needs are connected to the requirements. With these requirements the next step is to reviewing diagramming concepts, investigate appropriate existing diagrams based on the needs and requirements, evaluate their capabilities, and identify gaps between the capabilities of those diagrams and the derived requirements. The following chapter discusses this in more detail. The derived requirements indicate that what is needed to improve ABM is a holistic methodology that incorporates the needs of building a conceptual model of complex systems with the needs to conceptually sanction models. The next section details how this methodology fits into the current practice of ABM, and simulation in general, to provide further insight into and justification concerning this sanctioning methodology.

6.2 Further Justification and Insight into a New ABM Sanctioning Methodology

In general, there are two main sanctioning foci for those currently building ABM simulations: model fitting and model testing (this terminology and fundamental concept
Figure 29: Current Practice Needs Mapped to Diagramming Technique Requirements

- Requirements
  1. Aids in learning and conveying system knowledge
  2. Incorporates proper engineering judgment
  3. Aids in translating the conceptual model into a computerized model
  4. Emphasizes the developing and sanctioning of micro-level behaviors
  5. Displays the theories and assumptions built into the model for quantitative analysis
  6. Conveys the conceptual model's logic and structure for qualitative analysis
  7. Completely represents the simulation so it can be reproduced by independent evaluators
  8. Provides justification for all structures and actions in the simulation
  9. Reviewable by evaluators of varied simulation and domain expertise levels
  10. Must be able to represent Organized and Disorganized Complex Systems

- Needs
  1. Development and documentation tools for ABM that are independent of software
  2. To be studied as an independent simulation discipline/tool
  3. A change in the appropriate expectations of ABM
  4. A descriptive modeling tool for independent replication
  5. More stringent and ABM appropriate validation review

Solution Concept

ABM Diagramming Technique

Needs Mapped to Requirements

Derived Requirements
is borrowed from [109], however there is no direct correlation with Stasser’s definitions and the ones used here). For both foci the ultimate goal is to have a simulation that appropriately mimics the real system macro behavior; matching the macro behavior indicates that the simulation could be operationally effective, however each focus has a different way to achieve this. In the model fitting focus, the parameters and theories that compose the micro-level of the model are “optimized” via some algorithm. The optimization is not based on observations from the real system. In essence, the micro-level portion of the model is systematically changed until the macro-level results are achieved. Conversely, in the model testing focus, the parameters and theories that compose the micro-level of the model are based on observations and experiments performed on the real system.

Even though these foci are at opposite ends of the spectrum, and certainly hybrids of these focuses exist, each extreme focus addresses the fundamental problem with ABM today; not much is understood about the real system. Model testing directly attacks the lack of knowledge by using the more traditional scientific method. Model fitting synthetically generates a feasible model to produce macro-level behavior. An advantage of using model fitting is potentially obtaining novel micro-level theories about how the real world operates. However, an infinite number of models could represent a real system and thus it is impossible to prove that any model is an accurate representation of the real system; in fact one can only disprove that a model does not represent the real system of interest. Therefore, even if the model fitting focus results in a good representation of the macro-level system behavior it does not mean that the model accurately reflects the real system. Instead, the simulation is only a proposed theory that needs to be thoroughly tested to determine if it is feasible in reality, which means that model testing will still be required.

If the simulation exhibits the intended macro level behavior then what does it matter if the micro-level behavior is correct? There are several examples that highlight
why strict model fitting without empirical evidence can be problematic. One example comes from professional car racing where prior to the race crews can make adjustments to their car to attempt to optimize the car’s performance for that particular track. After considering the layout of the track, road conditions, and after making several trial runs, suppose the crew outfits their car such that maximum performance is achieved during the race and the car ends up winning. As a result, for the next race the crew decides to use the same car adjustments because they would expect to see the same performance. However, each track and race condition is different. Thus, using the same car adjustments may result in poor performance for the next race. In the same way, adjusting a simulation until it matches one particular performance measure from one particular real system may only be a good result for that one real system. The problem here is ensuring that the extendability and robustness of the simulation exists in order to explore and extrapolate implications of other real systems in the same domain [14, 30, 69]. Having a micro-level model that is not properly sanctioned can ultimately lead to unreliable simulation results beyond the particular performance measures obtained for that particular real system modeled.

Another example to consider is a standard single server queuing model where the objective of the simulation is to achieve the theoretical queue performance [56]. For example, typical performance measures are the average time in the queue or system throughput. The standard approach to build this simulation is to observe the real system, measure arrival rates, measure server processing times, and then build a realistic representation of the system using some discrete event simulation packages. Now consider utilizing a model fitting focus using an ABM simulation with reproducing bugs. In this simulation, the bugs move about their environment looking for food and reproduce with other bugs, much like those of Sugarscape [35]. Key measures about the bugs, such as lifespan and birthrate, are mapped to the goal performance measures of the single server queuing model. Then, in the spirit of model fitting,
parameters concerning the bugs and their environment are adjusted using some algorithm until the simulation’s performance measures match the expected queuing performance measures. The bug model is then deemed useful for queuing analysis. Although these are extreme examples, both approaches can meet the expected theoretical performance measures. However, when it comes to sanctioning, the ABM bug simulation does not really represent the real system and therefore it is unlikely that anyone should sanction this simulation.

What emerges from analyzing these two sanctioning foci is two conditional statements about sanctioning ABM simulations. The first condition comes from model fitting: if the macro behavior is sanctioned, then the micro behavior may be sanctionable. However, taking this approach requires one to also ensure that the micro behavior is sanctionable or else the simulation as a whole may not be sanctionable. The second condition comes from model testing: if the micro behavior is sanctioned, then either the macro behavior will be generated by the simulation or the appropriate micro behavior has not been captured by the simulation. This is a strong condition particularly if macro behavior, sometimes referred to as emergent behavior, is unexpected and surprising by definition.

ABM emergent and macro-level behavior is only surprising or counter intuitive to us because of the way in which we can generate that behavior and not in the fact that the phenomena exists in first place. Within the short history of ABM, the initial belief was that to generate complex and emergent behaviors a complex model was required. The true surprise came when it was found that very simple models could generate emergent behavior. Since this discovery, discussion of emergent behavior has proliferated and has resulted in some terminology confusion [34]. However, the idea of emergent behavior is not new and it can be observed in every scientific discipline in the form of abstraction levels. For example, from atoms emerge molecules, from molecules emerge cells, from cells emerge organisms, and etc. Therefore, the issue with
this statement should not be with the fact that emergent behavior is surprising, but with how one can say that the macro-level behavior will appear when the appropriate micro-level behavior is included in the model.

Micro-level behavior naturally leads to macro-level behavior. In any system, entities can be identified that exhibit micro-level behavior that when examined together create macro-level behavior. In fact, the fundamental difference between micro-level behavior and macro-level behavior is the abstraction level of interest, even though both belong to the same system. Every macro-level behavior is the result of some micro-level behavior. If one can appropriately mimic the micro-level behavior then the macro-level behavior should follow. Even though emergent behavior appears unpredictable, it is not unexplainable [34]. Thus, each ABM simulation study should begin by coming up with some statement similar to: we want to model these entities and their interactions in this environment to get appropriate macro-level behavior.

While model fitting is a useful model generating tool, it is not a proper sanctioning technique because it can generate one of the infinite models that does not represent the real system. Model testing should be the focus of any sanctioning methodology because it emphasizes the need for the model to represent some abstraction of reality. Furthermore, sanctioning should focus on the micro-level behavior; having an appropriate micro-level behavior helps ensure that the complete model is sanctionable. If only the macro-level behavior is sanctioned, then considerable effort is still be required to ensure that the micro-level behavior is sanctionable. Up front, effort in micro-level sanctioning pays off in the long run as the complete model is sanctionable. If it turns out that the macro-level behavior is not emerging from the micro-level behavior, then there is some fundamental part of the puzzle missing from the micro-level model. This missing result will encourage designers to further research the micro-level behaviors, which is more in line with the spirit behind the scientific revolution. Since, simulations of less understood systems are becoming scientific theories in themselves, it is
vitally important they be based on some representation of reality. Therefore, more emphasis needs to be on conceptual modeling (micro-level) of these less understood systems. A survey of literature regarding how ABM simulations are sanctioned today indicates that there is too much focus is on exclusively obtaining macro-level behaviors that match those of the real system. In fact, the survey discussed in the previous chapter indicated that 48% of the articles did not consider micro-level sanctioning at all. Although 52% of the articles indicate some micro-level sanctioning, this number should be at 100% given all of the systems being simulated are not well understood.
An Exploration of Diagramming Techniques and Their Capabilities

This chapter explores existing diagramming techniques and matches their capabilities with requirements to identify if any individual technique, or permutations of individual techniques, can meet these requirements. The following section defines diagrams, relates them to models, and discusses their limitations and roles. Next this chapter develops a classification scheme for techniques based on what they objectives and capabilities. This development supports the idea that behavioral diagramming techniques have capabilities that match closely with ABM needs and requirements. The final section identifies capability gaps of current diagramming techniques and concludes that a new diagramming technique is needed to fill this gap.

7.1 An Overview of Diagrams

A diagramming technique is a graphical language that communicates features of an object or concept of interest. As with any language, each specific diagramming technique has a set of semantics (symbols used to form expressions) and syntax (rules defining the ways symbols combine to form concepts) that allow people to share their thoughts with others in standard ways [23, 24]. This is a primary reason why a diagramming technique has the potential to advance ABM; many of the needs and requirements are related to communication. Another characteristic of diagramming techniques is that they are designed to emphasize particular aspects of the concept or object being described. For example, a typical flow chart diagramming technique
emphasizes when and under what conditions activities execute, but there is often no
information concerning who or what executes the activity or how long the activity will
eexecute. Similarly, a diagramming technique can be designed to emphasize important
ABM features such as conceptual sanctioning.

The types of diagramming techniques of particular interest are those capable of
describing features of model systems to be simulated. This important capability is
used to screen candidate techniques based on the defined requirements. In support
of this goal it is important to carefully define and differentiate diagrams and models.
In the simplest sense diagrams are graphical descriptions of features of models and
models are abstractions of real or soon-to-be real systems. Distinguishing between a
model and a diagram can become difficult when models resemble diagrams such as
when models are graphic abstractions of systems. For example, an ARENA simula-
tion model visually looks like a diagram describing the flow of entities. However, a
complete ARENA model has enough detail specified in addition to its visual compo-
nents to be translated and executed into Siman code. Here the distinction is that a
model is a representation of a system, it translates input into output, and a diagram
of that model describes the process of translating input into output; diagrams are
graphical descriptions of features or information. Thus, models can be associated
directly with an "engine" that allows it to be "run". The goals of a diagram are
less ambitious, and may only describe select model features, rather than the more
complete description of a "model".

In working with different diagramming techniques, it can be difficult to clearly
prove that one technique is better than another; there is a lot of subjectivity involved.
For example, comparing diagramming techniques (graphical languages) can be like
comparing English and German (verbal languages). Individuals may prefer one over
the other, but saying that English is better than German is subjective. Thus, to ob-
jectively examine different diagramming techniques one must compare their intended
purpose, capabilities, and limitations. Examining diagramming techniques in this manner will effectively eliminate personal biases and allow a technique's capabilities determine how well the technique can satisfy the defined requirements.

7.2 Capabilities of Diagramming Techniques that Describe Model Systems

For this research, a key requirement of a diagramming technique is the ability to effectively describe the conceptual model of a simulation. Thus, the diagramming technique must be capable of describing model systems as they move through time. This requirement has two important features. The first is that the diagramming technique must describe dynamic relationships. Here dynamic relationships are defined as conditional and time dependent relationships such as those seen in a part flowing through a manufacturing facility. The second is that the technique must have a formalism that allows elaboration of how the model system is executed by a computer. These factors point to the diagramming techniques developed in the fields of Systems Engineering and Computer Science because both utilize diagramming techniques and both are concerned with designing and documenting dynamic systems. In particular, the techniques developed in Computer Science are concerned with designing systems that are computer executable. Therefore, diagramming techniques used in these fields are likely candidates capable of effectively describing a conceptual model of a simulation.

7.2.1 Organizational Diagramming Techniques

Systems Engineering and Computer Science have two general categories of diagramming techniques used to describe model systems. The first set of techniques describe the organizational structure of a model system. These organizational diagramming techniques (ODTs) capture static relationships and highlight the structure of various
components of the model system. Some common OTDs include:

- Entity-Relationship Diagrams [27]
- Higraphs [46]
- Data Flow Diagrams [110]
- IDEF0 Diagrams [23, 24]
- $N^2$ Charts [67]
- UML 2.0 Structural Diagrams (including SysUML equivalents) [4, 5, 24, 19, 94, 116]
  - Class Diagrams (Block Diagrams)
  - Component Diagrams
  - Composite Structure Diagrams (Internal Block Diagrams)
  - Package Diagrams
  - Object Diagrams
- UML 2.0 Use Case Diagrams (including SysUML equivalent) [4, 5, 19, 24, 94, 116]
- SysML Requirement Diagrams [24, 116]
- SysML Parametric Diagrams [24, 116]

While these diagramming techniques are effective at describing static relationships and organizational structure, they do not capture the dynamic behavior of a model system. Since this is a key requirement in ABM, OTDs are eliminated as possible candidate techniques. Despite this, understanding these diagrams is valuable since
some of their best practices and unique structures can be utilized in the development of a new diagramming technique. For example, naming conventions could be adopted from IDEF0 as well as other “standard” and “accepted” practices such as arcs representing relationships.

7.2.2 Behavioral Diagramming Techniques

A second set of diagramming techniques in Systems Engineering and Computer Science describe the dynamic behavior of the model system. These behavioral diagramming techniques (BDTs) capture the dynamic control and execution of activities or functions of a model system and describe the desired high-level capabilities sought after in candidate diagramming techniques. The remainder of this section explores specific types of BDTs and evaluates their individual capabilities against the detailed requirements. These techniques have:

- The capability to aid in learning and conveying system knowledge (Req. #1);
- The capability to incorporate proper engineering judgment (Req. #2);
- The ability to convey the conceptual model’s logic and structure for quantitative analysis (Req. #6);
- The capability to be reviewable by evaluators of various simulation and domain expertise levels (Req. #9); and
- The ability to represent Organized and Disorganized Complex Systems (Req. #10).

BDTs are separated into two categories and evaluations in each category occur separately. This is more efficient because the specific techniques in each category have very similar objectives and components and evaluating each technique individually would be repetitive. The two categories of BDTs further clarifies a framework that
describes diagramming techniques, their capabilities, and their differences. The two categories of BDTs are process flow behavioral diagramming techniques (PFBDTs) and machine behavioral diagramming techniques (MBDTs).

### 7.2.2.1 Process Flow Behavioral Diagramming Techniques

PFBDTs describe the flow of control of activities or functions of a model system. They describe the order and conditions in which activities or functions are performed. They do not describe how the model system actually executes. PFBDTs do not contain the semantics and syntax to relate the process flow activity to the model system’s “engine,” how the model system transitions, and what impact that action/activity has on the current status of the model system. Some common PFBDTs include the following:

- **Functional Flow Block Diagrams (FFBDs)** were originally developed in the 1950’s and 1960’s by TRW to describe the order that functions of the system are executed by adding semantics in conjunction with ODTs. FFBDs emphasize functional decomposition of a system [23, 24, 116].

- **Enhanced Functional Flow Block Diagrams (EFFBDs)** are similar to FFBDs, but with added syntax to describe iterations, loops, and replications [23, 24].

- **Behavior Diagrams (BDs)** were developed in conjunction with the Distributed Computer Design System for the Department of Defense and have similar features and capabilities as FFBDs (in some cases they have the exact same semantics). BDs are also focused on functional decomposition of a system and utilize ODTs [3, 23, 24].

- **Control Flow Diagrams (CFDs)** incorporate similar semantics and syntax of Data Flow Diagrams (an ODT) with the addition of broken arcs to show some functions being turned on or off to indicate changes in the operating mode of
the model system [23, 24]. CFDs show where the flow of control can change, but they do not describe what causes these changes.

- **UML 2.0 Activity Diagrams** (UADs) and **SysML Activity Diagrams** (SADs) are similar to flow charts except with well defined semantics and syntax. UADs and SADs capture the flow of control or the data passing between components of a model system and often represent high-level business and operational processes [4, 5, 19, 24, 94, 116].

- **UML 2.0 Interaction Diagrams** (UIDs) and **SysML Interaction Diagrams** (SIDs) [4, 5, 19, 24, 94, 116]
  - **Communication Diagrams** describe the relationships between components in terms of the order that messages are passed between them. These diagrams are primarily used in UML 2.0 and SysML to describe the relationships between other diagramming techniques within the UML 2.0 and SysML paradigm.
  - **Sequence Diagrams** describe the flow of logic between various components of the model system as it executes.
  - **Timing Diagrams** are a variation of Sequence Diagrams that focus specifically on the order and duration of activities occurring within a set period of time. While time is explicitly captured, Timing Diagrams do not capture what causes these changes as needed when replicating the model system.
  - **Interaction Overview Diagrams** are closely related to UADs and describe the flow of control of other UIDs.

A shortcoming of PFDBTs is that they do not provide structures for indicating more detailed operations in translating input into output for the model system. This capability is critical in order to simulate a model system. For example, consider a
manufacturing system that converts raw materials into a finished product with two machines in series. A PFBDT would describe this system in terms of the order required to produce finished products. First, raw material arrives, then machine one processes it, then machine two processes it, and finally finished goods are produced. In this example there is no description of how the manufacturing system’s operations are changing. A PFBDT does not describe the changes each machine makes to a piece, the time to execute these machine actions, the events that cause a piece to go from one machine to another, and many other simulation “engine” related details. All that is given is the order of activities required to produce finished goods from raw material. More information would be needed for a person or machine to mimic this process.

There are two requirements that PFBDTs cannot satisfy. The first is it cannot aid in translating the conceptual model into a computerized model (Req. #3). A PFBDT lacks enough information to capture the conceptual model that is to be mimicked by a computerized model. The second is its inability to completely represent a model and its simulation so it can be reproduced by independent evaluators (Req. #7). Not being able to satisfy these important requirements indicates that PFBDTs are not effective candidate diagramming techniques.

7.2.2.2 Machine Behavioral Diagramming Techniques

MBDTs describe the dynamic modes of a model system and the events that cause the model system to transition between modes. MBDTs have the additional syntax and semantics to describe the “engine” of a model system that is missing in PFBDTs. This additional information provides the capability to mimic the model system’s translation of input into output and how it changes over time. Various MBDTs are discussed in the following paragraphs.
Petri Nets

Petri Nets (PNs) were first formally introduced by Carl Petri in 1962 and are used to describe systems in terms of places, transitions, tokens, and arcs. Here places are where tokens can reside and describe the modes of the system, transitions represent events or activities, tokens describe the current state of the system, and arcs represent the relationships between places and/or transitions. PNs are capable of representing systems that exhibit concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic characteristics. However, PNs do not have any structures to modularize large and complex systems. This makes PNs challenging to construct and understand when the system is large and complex. Since PNs are based on strict mathematical definitions and formulations, PNs can be both simulated and analyzed mathematically [23, 24, 81, 90, 92]. Figure 30 displays a simple PN diagram of a chemical reaction.

State Diagrams

State Diagrams (SDs) or State-Transition Diagrams (STDs) were introduced by Taylor Booth in 1967 to represent Finite-State Machines and are used to describe the states and events of systems in terms of boxes and arcs. Here boxes represent a mode of the system and arcs represent the transition between modes of the system. Arcs are often accompanied by events that trigger the transition of the system to a new
mode and the action taken by the system when that event occurs [20, 23, 24, 32].Although they are technically capable of representing large and complex systems, one criticism of SDs is that they do not effectively display these systems very well because, similar to PNs, they have no structures to represent modular hierarchies [46]. Figure 31 displays a simple example of a SD of a basic telephone system.

**Statecharts**

Statecharts were created by David Harel in 1987 in response to criticisms that SDs cannot represent large and complex systems effectively. Statecharts extend SDs by adding syntax and semantics to better handle issues of hierarchy, concurrency, and communication seen in large and complex systems. A key idea in Statecharts is that states (blocks with rounded corners and representing modes of the system) can exist within each other to display concurrency and hierarchies. Statecharts also allow activities to be performed not only when arcs are activated but also when entering, exiting, and/or residing within a state. Another feature of Statecharts is that an arc emanating from a black dot indicates the initial state when a state containing that black dot is initialized. Although this is not a complete description of Statecharts, these additional features allow Statecharts to be a powerful and easy-to-understand diagramming technique when attempting to describe abstract complex systems that exhibit concurrency and hierarchies [23, 24, 46, 47]. Figure 32 displays an example of a Statechart Diagram for a ceiling fan with two speeds and a light.
UML 2.0 and SysML State Machine Diagrams

UML 2.0 State Machine Diagrams (USMDs), and equivalently SysML State Machine Diagrams (SSMDs), are based upon the fundamental elements and ideas of SDs and Statecharts but have more formally defined syntax and semantics that fit into the overall UML 2.0 or SysML frameworks. Thus, USMDs and SSMDs have the same general capabilities of SDs and Statecharts, but are positioned to address the particular domains of each framework. For UML 2.0 and SysML these domains are software development and systems development, respectively [4, 5, 19, 24, 94, 116]. While the specific and formalized semantics and syntax of USMDs and SSMDs are valuable for their domains, selecting either as a candidate ABM technique would require users become familiar with UML 2.0 and SysML rather than the fundamental diagramming semantics and syntax that these techniques are based on. This requires that a user learn additional and potentially unnecessary elements to utilize USMDs or SSMDs. Furthermore, the requirements and needs do not specify that a diagramming technique need to develop a software or systems specific architecture. In some cases this may make USMDs and SSMDs inappropriately geared for the purposes of the user.
or model they are developing. Based on these factors USMDs and SSMDs are not considered as candidate diagrams, but are represented under the SD and Statechart banners with their formal semantics and syntax noted for potential adoption in a new diagramming technique.

While composed of many individual diagramming techniques, UML 2.0 and SysML diagramming techniques are intended to be utilized as an integrated diagramming technique. This means that as a whole UML 2.0 and SysML diagramming techniques are capable of representing a specific behavior, characteristic, or relationship where individual diagramming techniques fail. This ability highlights one of the key design goals of UML 2.0 and SysML, which is to describe in detail the architecture of any system. However, this ability adds a significant amount of complexity to these techniques and can take the focus away from building a model or system and place it more on building the diagram. While capable of representing many if not all systems, UML 2.0 and SysML were not designed for the ABM analyst.

Since MBDTs have the syntax and semantics to describe the execution of the model system there are additional requirements they can satisfy beyond those satisfied by PFBDTs. The first is the ability to aid in the translation of a conceptual model into a computerized model (Req. #3). The second is the ability to completely represent a simulation so that it can be reproduced by independent evaluators (Req. #7).

### 7.3 Capability Gap Analysis and Conclusions

A summary of the requirements that each specific BDT satisfy is shown in Table 3. Table 3 indicates that MBDTs satisfy more requirements than PFBDTs, but overall there are still three requirements that have not been satisfied by any diagramming technique. These unsatisfied requirements are:

- The ability to emphasize the development and sanctioning of micro-level behaviors (Req. #4);
Table 3: Diagramming Technique Capability to Requirement Analysis

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Behavioral Diagramming Technique</th>
<th>Process Flow</th>
<th>Machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Aids in learning and conveying system knowledge</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2. Incorporates proper engineering judgment</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3. Aids in translating the conceptual model into a computerized model</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4. Emphasizes the developing and sanctioning of micro-level behaviors</td>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>5. Displays the theories and assumptions built into the model for quantitative analysis</td>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>6. Conveys the conceptual model’s logic and structure for qualitative analysis</td>
<td></td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>7. Completely represents the simulation so it can be reproduced by independent evaluators</td>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>8. Provides justification for all structures and actions in the simulation</td>
<td></td>
<td>✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>9. Reviewable by evaluators of varied simulation and domain expertise levels</td>
<td></td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
<tr>
<td>10. Must be able to represent Organized and Disorganized Complex Systems</td>
<td></td>
<td>✓ ✓ ✓ ✓</td>
<td>✓ ✓ ✓ ✓</td>
</tr>
</tbody>
</table>
• The ability to display the theories and assumptions built into the model for quantitative analysis (Req. #5); and

• The ability to provide justification for all structures and actions in the simulation (Req. #8).

The commonality of these requirements is that they all are concerned with the sanctioning and documenting of the theories and assumptions built into the simulation. This leads to the conclusion that current diagramming techniques focus primarily on describing the “whats” and the “hows”. They do not provide the “whys,” which have been identified as being a very important aspect of modeling and scientific evaluation. To fill this capability gap, a new diagramming technique is needed that not only describes the model system in a similar manner as MBDTs but also includes valuable conceptual sanctioning information. Only then will a diagramming technique be able to satisfy the requirements in an effort to advance ABM.
The Conceptual Model for Simulation Diagram

This chapter presents a new diagramming technique, the Conceptual Model for Simulation (CM4S) Diagram, that fulfills the requirements for a ABM simulations. This chapter contains four main sections. First the basic motivation behind the CM4S Diagram is discussed. Next the key structures, semantics, and syntax of the diagramming technique are presented. Then a CM4S Diagram of the well known Sugarscape Simulation [35] is presented to demonstrate how the diagramming technique is used and to demonstrate its overall effectiveness at representing the conceptual model and the appropriate validation technique. Finally, the capabilities of the CM4S Diagram are discussed and it is shown that the CM4S Diagram meets all of the requirements identified in chapter seven.

8.1 The Need for a New Diagramming Technique

From chapter seven, it is clear that a new Machine Behavioral Diagramming Technique (MBDT), that includes important conceptual sanctioning information, is needed. In particular the new MBDT needs the following abilities:

- The ability to emphasize the development and sanctioning of micro-level behaviors (Req. #4);

- The ability to display the theories and assumptions built into the model for quantitative analysis (Req. #5); and
• The ability to provide justification for all structures and actions in the simulation (Req. #8).

There are two key ways to develop a new MBDT with these capabilities. The first is to construct a completely new MBDT to satisfy these three conceptual sanctioning requirements as well as the other seven requirements discussed in the previous chapter. The key advantage of this approach is a diagramming technique designed specifically for the purposes identified in the requirements. The main disadvantage of this approach is appropriately defining a completely new set of semantics and syntax when only three of the ten requirements were not satisfied. The second approach is take an existing MBDT and extend that MBDT to satisfy the conceptual sanctioning requirements. Selecting this approach minimizes new developments while re-using fundamental concepts from an already proven MBDT. The second approach was selected based on these advantages.

There are three main MBDT candidates upon which to develop a new MBDT: Petri Nets, State Diagrams, and Statecharts (Note: UML 2.0 and SysML State Diagrams are based on State Diagrams and Statecharts). Upon examining each of these MBDTs it is clear that Statecharts has several advantages over Petri Nets and State Diagrams that make it a logical choice to begin developing a new MBDT. First, Statecharts have the same capabilities of State Diagrams, but State Diagrams cannot represent large and complex systems as effectively as Statecharts. Similarly, Statecharts have many of the same capabilities of Petri Nets, but Petri Nets do not have the structures to help modularize large and complex systems as do Statecharts. Based on these features to easily represent large and complex systems, Statecharts were selected to provide the foundational starting point of a new MBDT that focuses on the conceptual sanctioning requirements.
8.2 Adapting Statecharts to Satisfy the Conceptual Sanctioning Requirements

A new MBDT, based on Statecharts, must convey more information than previous MBDTs. Statecharts and the other MBDTs primarily convey “how” information and the conceptual sanctioning requirements define the need for conveying “why” information. To adapt the Statechart diagramming technique key features need to be added that describe the “why” information that is associated with the “how” information; the new MBDT needs to include another dimension of information.

This new dimension of information can exist in many forms. For example, the “why” information could be a written explanation, a reference to an experimental study, academic work, accepted theory, or a mathematical proof. Therefore, a feature capable of capturing this information must accommodate many possible inputs. A set of properties associated with each diagramming element that convey that element’s “why” information can satisfy the conceptual sanctioning requirements without sacrificing the “how” information. Furthermore, each property’s field can allow variable input to handle the various needed inputs.

There are issues with adding properties to each element in any MBDT. First, all of the properties needed for each element must be clearly defined to completely describe the needed information. This issue is easily addressed through the defined requirements and through testing and development. The second is to effectively create and retrieve the associated properties from an element since these properties cannot be efficiently represented visually. These two issues are addressed through the utilization of modern drawing and diagramming software products such as MS Visio or SmartDraw. Each product can associate properties to shapes and elements. By creating the appropriate elements and properties using one of these products the issues with property entry, management, and reporting as well as diagram construction, storing, and sharing are minimized.
The other major issue with adding properties to each element is ensuring that each individual state, condition, event, and action has a distinct set of properties. Currently, Statecharts have only three major shapes: rounded rectangles to represent states, arcs with an arrow to represent transitions between states, and circles for pointing to initial states. Each of these shapes describe the flow of control through the system and are associated with events, conditions, and/or actions. The close connection between the visual flow of control with activities or conditions to be executed or evaluated is one of the hallmarks of diagramming techniques. However, it presents a problem when trying to include “why” information because there are times when the justification of an action may not be the same as the justification of the state that contains that action. For example, deciding to decompose a system in different states with different grouping of activities may have significant consequences on the conceptual model and would require justification in the new diagramming technique. Thus, several additional features and changes are needed to effectively differentiate between flow of control and activities.

The key additional features included are a rectangle shape to represent actions, a diamond shape to represent conditions/events, circles shapes to represent important variables in the simulation, and a pentagon shape to represent how, when, and why data is collected. These new shapes clearly distinguish elements and their justification. They also provide fundamental building blocks for constructing the conceptual model and they provide a more familiar abstract representation of the conceptual model for those not familiar with the syntax and semantics of Statecharts.

The Conceptual Model for Simulation (CM4S) Diagram was created by adapting the fundamental components of Statecharts with new information properties and creating new shapes to clearly distinguish elements.

While the CM4S Diagram is based on Statecharts, it is not intended to be used as a replacement of Statecharts or any software design diagramming technique. The
CM4S Diagram is best utilized in the early development of a simulation because it emphasizes validation and highlights the issues commonly encountered in building computer simulations. Furthermore, the CM4S Diagram provides vital information concerning assumptions and purpose of the simulation and can be used throughout the life-cycle of a simulation. Therefore, the CM4S Diagramming Technique fits into the niche of agent-based modeling efforts and frameworks, early simulation prototyping and experimentation, and as a reference point throughout the life of the simulation.

8.3 Description of the CM4S Diagram

In this section the key aspects of the CM4S Diagram are described. This includes a review of the diagramming technique’s syntax and semantics, shape naming conventions, and its use in conjunction with drawing/diagramming software. An example of a CM4S Diagram is shown in Figure 34.

8.3.1 Basic Syntax and Semantics

The CM4S Diagram consists of arcs, history pointers, and initial pointers. Initial pointers indicate what block is initially active whenever the block containing that initial pointer is activated. History pointers work in a similar fashion except they indicate that the current active block is the one that was last visited when the block containing the history pointer was last exited.

Arcs in the CM4S Diagram only show the flow of control between pointers and blocks; they do not contain any information concerning events, triggers, and/or guards. The CM4S Diagram arcs act the same ways as flow chart arcs. Arcs can only emanate from pointers, blocks, and decision shapes connected to block shapes and arcs can only point to history pointers and blocks. Arcs emanating from pointers and blocks are followed once all of the activities associated with the emanating shape are complete. However, arcs emanating from decision shapes are followed based the
Figure 33: Sugarscape CM4S Diagram Example

1 Sugarscape Conceptual Model

1 Build the Environment
A Assign Environment Coordinates
B Assign Lattice Max & Current Sugar Levels
C Assign Lattice Sugar Growth Rates
D Create the Agents

2 Assign Agent Parameters
A Assign Start Coordinates
B Assign Metabolism
C Assign Life Span
D Assign Vision
E Randomly Assign Agent Number
F Assign Initial Sugar Level

3 Time Progression

2 Prepare for Next Time Step
A Grow Lattice Sugar
B Increment Time Step
C Reset Agent Numbers

4 End Simulation
A Stop?
condition in the decision shape. The only way to transition between blocks is via arcs.

There are five other shapes in the CM4S Diagram: Blocks, Actions, Decisions, Recorders, and Variables. The Block Shape is a rounded rectangle that represents an abstraction of the system that is composed of actions, decisions, recorders, and variables. Blocks can exist within each other to represent hierarchies of a system’s abstraction and blocks can be partitioned with dotted lines to represent concurrent abstractions within a higher level abstraction. The Block Shape has seven properties. The From property describes the blocks that point to the current block. The To property describes the blocks that are pointed to by the current block. The Actions property describes actions that are members of the block. The Decisions property describes the decisions that are members of the block. The Variables property describes variables that are members of the block. The Recorders property describes the recorders that are members of the block. Finally, the Basis property describes the rationale for the block for clarification and sanctioning purposes.

The Action Shape is a rectangle that describes the general actions and activities that occur within the system. Actions can only exist within a block and have many properties to describe the activities executed as well as when and how often to execute them. The Action Shape has nine properties. The Member Of property describes the block where the action is directly residing. The Behavior property is the written description of the behavior that the action is to achieve. The Pseudo Code - Function property is the pseudo code that represents the key behavior of the action. The Pseudo Code - Update property is the pseudo code of how often or under what conditions the Pseudo Code - Function updates. The Pseudo Code - Start property is the pseudo code of when the Pseudo Code - Function begins execution. The Pseudo Code - Stop property is the pseudo code of when the Pseudo Code - Function stops execution. The purpose of these pseudo code family of properties is to provide details how the
behavior is achieved computationally. The *Variables* property describes the variables utilized or changed by the action. The *Sequence within Block* property is the order that the action is executed within the block and is used primarily for breaking ties with other shapes. Finally, the *Basis* property describes the rationale for the behavior of the action for clarification and sanctioning purposes.

The Decision Shape is a diamond that describes the conditions for transitioning between blocks. Decisions are associated with blocks and are outside their associated block, but touch it. Decisions have properties describing the conditions to evaluate, when and how often to evaluate them. The Decision Shape has nine properties. The *Member Of* property describes the block where the decision resides. The *Behavior* property is the written description of the conditional behavior that the decision is trying to achieve. The *Pseudo Code - Condition* property is the pseudo code that represents the key conditions of the decision. The *Pseudo Code - Update* property is the pseudo code of how often or under what conditions the *Pseudo Code - Condition* is re-evaluated. The *Pseudo Code - Start* property is the pseudo code of when the *Pseudo Code - Condition* is evaluated. The *Pseudo Code - Stop* property is the pseudo code of when the *Pseudo Code - Condition* stops being evaluated. The *Variables* property describes the variables utilized or changed by the decision. The *Sequence within Block* property is the order of decision execution within the block and is primarily used for breaking processing ties with other shapes. Finally, the *Basis* property describes the basis for the conditional behavior of the decision for clarification and sanctioning purposes.

The Recorder Shape is a pentagon that describes how and when data is collected within the system being simulated. Recorders only exist within a block. The Recorder Shape has nine properties. The *Member Of* property describes the block where the recorder resides. The *Behavior* property is the written description of the collection behavior the recorder achieves. The *Pseudo Code - Function* property is the pseudo
code that represents the key behavior of the recorder. The \textit{Pseudo Code - Update} property is the pseudo code of how often or under what conditions the \textit{Pseudo Code - Function} is updated. The \textit{Pseudo Code - Start} property is the pseudo code of when the \textit{Pseudo Code - Function} begins execution. The \textit{Pseudo Code - Stop} property is the pseudo code of when the \textit{Pseudo Code - Function} stops execution. The \textit{Variables} property describes the variables used by the recorder. The \textit{Sequence within Block} property is the execution order within the block and is primarily used for breaking processing ties with other shapes. Finally, the \textit{Purpose} property describes the purpose for collecting the data for clarification and sanctioning purposes.

The Variable Shape is a circle that describes a key variable of the system. Variables only exist within a block. The Variable Shape has four properties. The \textit{Member Of} property describes the block where the variable resides. The \textit{Behavior} property is the written description of what the variable represents. The \textit{Value} property is the initial value of the variable. Finally, the \textit{Basis} property describes the basis for the variable for clarification and sanctioning purposes.

A summary of these shapes is shown in Figure 34. See [46, 47] for a more complete and formalized description of the syntax and semantics of the fundamental structure of Statecharts.

\subsection*{8.3.2 Shape Naming Conventions}

The naming convention for the shapes in the CM4S Diagram ensures the uniqueness of each shape, provides hierarchical information, and further determines how execution ties are broken. Every shape in a diagram has an alpha-numeric name along with a descriptive name that appropriately describes the shape. The alpha-numeric name of a shape is determined by its location within and relationship to other shapes in the diagram. The descriptive name should concisely describe the shape’s high-level purpose.
Figure 34: CM4S Diagram Shape Descriptions

<table>
<thead>
<tr>
<th>Name</th>
<th>Shape</th>
<th>Properties</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td></td>
<td>From</td>
<td>Represents an abstraction of the system that is composed of actions, decisions, recorders, and variables. Blocks can exist within each other to represent hierarchies of a system's abstraction and blocks can be partitioned with dotted lines to represent concurrent abstractions within a higher level abstraction.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>To</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decisions</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Recorders</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Basis</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td></td>
<td>Member Of Behavior</td>
<td>Describes the general actions and activities that occur within the system. Actions can only exist within a block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Update</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence within Block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Basis</td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td></td>
<td>Member Of Transitions To</td>
<td>Describes the conditions to transition between blocks. Decisions can only be associated with blocks and must be outside their associated block but still be touching it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Condition</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Update</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence within Block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Basis</td>
<td></td>
</tr>
<tr>
<td>Recorder</td>
<td></td>
<td>Member Of Behavior</td>
<td>Describes how and when data is collected within the system being simulated. Recorders can only exist within a block</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Function</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Update</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Start</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pseudo Code - Stop</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Sequence within Block</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td></td>
<td>Member Of Behavior</td>
<td>Describes a key variable of the system. Variables can only exist within a block.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Basis</td>
<td></td>
</tr>
</tbody>
</table>
The alpha-numeric conventions for the CM4S Diagram are similar to standard outlining formats. Shapes have the same alpha-numeric name as the shape that it exists within, but with the addition of an extra alpha-numeric character. For example, if the block shape “Move” is the first block shape that exists within the block shape “1.2 Agent” then the appropriate name for the block shape would be “1.2.1 Move”. It should be pointed out that block shapes can only be given numeric characters. All other shapes are given alphabetic characters. For example, if the action shape “Scan Ahead 3 Spots” is the first action shape to appear in the “1.2.1 Move” block shape then action shape should be named “1.2.1.A Scan Ahead 3 Spots”. Each Action, Decision, Variable, and Recorder shape has its own independent naming sequence, but Action, Decision, Variable, and Recorder shapes that exist in the same block will have similar alpha-numeric names. If only one of all four of these shapes existed within the the “1.2.1 Move” block shape, then each would have the same alpha-numeric name of “1.2.1.A”. In the event of an execution tie the shapes are sorted in alpha-numeric order to determine execution order. A complete example of these naming conventions follows.

8.3.3 Constructing a CM4S Diagram with Drawing/Diagramming Software

The CM4S Diagram was designed for use with computer drawing/diagramming software such as SmartDraw and MS Visio. A MS Visio template for the CM4S Diagramming Technique is available for download at http://www.CM4SDiagram.com.

There are several things to keep in mind when using software tools to construct a CM4S Diagram. The first is consider how the diagram will be displayed on paper or in a presentation. While the computer can show the diagram on one page, it is often valuable to break the diagram into multiple pages to allow for larger font sizes and to highlight distinct components of the conceptual model. To accommodate this several syntax and semantic components have been added to the diagramming technique.
First, new pages are only needed to show the contents of block shapes. Second, if the full contents of a block shape is shown on another page the name of that block shape is underlined. Finally, all arcs leaving and entering a block shape are shown on all of the pages where the block shape is present. Furthermore, to help shorten the length of alpha-numeric names only the “lowest” block shape on a page needs the complete alpha-numeric name. The remaining shapes only need the last character of their alpha-numeric name displayed for each page.

Another consideration when using software tools to construct a CM4S Diagram is balancing the visual construction with the detailed information construction. When constructing a diagram to focus first on building the shapes and signifying the flow of control and then to focus on filling in the detailed information. Also, when filling in the detailed information it is important to be consistent in descriptions and level of detail. Following these suggestions will allow for an effective development of a CM4S Diagram using software tools.

8.4 A CM4S Diagram of the Sugarscape Simulation

To demonstrate the functionality and effectiveness of the CM4S Diagramming Technique a CM4S Diagram of the Sugarscape ABM Simulation [35] is constructed. The Sugarscape Simulation is a good choice for a number of reasons. First, it is a well-known, fairly basic, and a purely notional simulation that utilizes fundamental concepts found in the ABM paradigm. Using Sugarscape provides a clear example of how the CM4S Diagram captures common conceptual ABM paradigm behaviors while showing how these behaviors are executed within a computer simulation without requiring extensive domain-specific knowledge. Constructing a CM4S Diagram of the Sugarscape Simulation also provides the modeler and evaluators with concise, detailed and easily accessible written documentation of the simulation; one could read an entire book on this simulation [35]. This allows for modelers and evaluators to
see the direct translation between the written conceptual model and the constructed CM4S Diagram. Thus, a Sugarscape Simulation example of the CM4S Diagram is both instructive and informative.

8.4.1 Overview of the Sugarscape Simulation

In the book *Growing Artificial Societies: Social Science from the Bottom Up* the authors present a notional ABM simulation called Sugarscape to demonstrate the usefulness of the ABM paradigm in social science [35]. In the most basic scenario of Sugarscape (found in chapter 2) agents exist in a two-dimensional lattice environment (50 by 50 discrete cells in a torus shape with two “mountains” of sugar) where sugar grows as the only source of food. In each time step, agents are randomly selected to take turns visually searching for an unoccupied neighboring lattice position, containing the most sugar that they can move to. Once at the new position the agents consume the sugar and, based on their time step metabolism, either survive to the next time step or die and are removed from the simulation. After each time step the environment grows new sugar at some set rate.

The version of the Sugarscape Simulation in chapter two of the book adds several other agent features. Agents are provided randomly assigned values for movement direction, vision range, metabolism range, initial starting position, initial endowment of sugar, and life span. If the agent reaches its life span, then it is removed from the simulation and a new agent with similarly random attributes enters the simulation. There are several other attributes and features in the actual Sugarscape simulation but they are not utilized in the current example for simplicity.

8.4.2 Walk-through of the Sugarscape CM4S Diagram

The CM4S Diagram of the Sugarscape Simulation is composed of two main pages. The first page describes the high-level execution and construction of the simulation. The visual representation of the first page of the CM4S Diagram is shown in Figure 149.
The corresponding data for the Action, Block, Decision, Recorder, and Variable Shapes on the first page are shown in Figures 36, 37, 38, 39, 40, respectively.

Figure 35 depicts several important parts of the simulation. At the highest level of abstraction is the block called Sugarscape Conceptual Model. Everything within this block represents the conceptual model and how the simulation executes. The *Basis* property data associated with this block (Figure 37) provides a high-level justification for building this model and documents its purpose. Within the Sugarscape Conceptual Model block are 13 variables (circles) that are used throughout the simulation. Beyond these variables, no other shapes belong to the Sugarscape Conceptual Model block.

Four blocks are a single abstraction level lower within the Sugarscape Conceptual Model block: Build the Environment, Assign Agent Parameters, Time Progression, and End Simulation. The block executed first is indicated by the initial pointer (dark circle) at that level of abstraction. Thus, the Build the Environment block is executed first. There are four actions that belong to the Build the Environment block. Each action helps in initializing the sugarscape environment. The execution sequence is defined by their *Sequence within Block* property (see Figure 36). The defined sequence of these actions is as follows: Assign Environment Coordinates, Assign Lattice Max & Current Sugar Levels, Assign Lattice Sugar Growth Rates, and Create the Agents. Once all of these initialization actions are executed, control passes to the Assign Agent Parameters block, where all of the agent parameters/variables are appropriately assigned, after which control passes to the Time Progression block.

The Time Progression block introduces several new features into the Sugarscape simulation. First, the block is divided into two areas. This indicates that while the Time Progression block is active both areas of the block are executed “concurrently”. The upper area of the block shows simulation time step management and the agents in the model shown in the lower area of the block. Together these areas represent model
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Pseudo Code + Function</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variables</th>
<th>Sequence within Block</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A</td>
<td>Action A</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the x and y coordinates of each lattice node (500 in total)</td>
<td>On entry</td>
<td>A. Latt X cord B. Latt Y cord</td>
<td></td>
<td></td>
<td>1</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21</td>
</tr>
<tr>
<td>Action B</td>
<td>Action B</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the maximum and current sugar levels to the appropriate values</td>
<td>On entry</td>
<td>C. Latt Max Sugar D. Latt Current Sugar</td>
<td></td>
<td></td>
<td>2</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21</td>
</tr>
<tr>
<td>Action C</td>
<td>Action C</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the current rate of sugar growth based on the particular experiment</td>
<td>On entry</td>
<td>E. Latt Growth Rate</td>
<td></td>
<td></td>
<td>3</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996</td>
</tr>
<tr>
<td>Action D</td>
<td>Action D</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the x and y coordinates of each lattice node (500 in total)</td>
<td>On entry</td>
<td>F. Latt X cord G. Latt Y cord</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action E</td>
<td>Action E</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the current rate of sugar growth based on the particular experiment</td>
<td>On entry</td>
<td>H. Latt Growth Rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action F</td>
<td>Action F</td>
<td>Member Of</td>
<td>Build the Environment</td>
<td>Sets the x and y coordinates of each lattice node (500 in total)</td>
<td>On entry</td>
<td>I. Latt X cord J. Latt Y cord</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 36: Sugarscape CM4S Diagram Page 1 - Action Shape Data**

1. **Display Text**: Displays the text that is used to display the action in the Sugarscape CM4S diagram.
2. **Member Of**: Indicates the member of the action.
3. **Behavior**: Describes the behavior that the action performs.
4. **Pseudo Code + Function**: Specifies the pseudo code and function that is used to implement the action.
5. **Pseudo Code - Update**: Indicates the pseudo code that is used to update the action.
6. **Pseudo Code - Start**: Specifies the pseudo code that is used to start the action.
7. **Pseudo Code - Stop**: Indicates the pseudo code that is used to stop the action.
8. **Variables**: Lists the variables that are used in the action.
9. **Sequence within Block**: Indicates the sequence within the block.
10. **Basis**: Specifies the basis for the action.
### Figure 37: Sugarscape CM4S Diagram Page 1 - Block Shape Data

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>To</th>
<th>From</th>
<th>Actions</th>
<th>Decisions</th>
<th>Recorders</th>
<th>Variables</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>1 Sugarscape Conceptual Mode</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Build the Environment</td>
<td>2 Assign Agent Parameters</td>
<td></td>
<td>A Assign Environment Coordinates B Assign Lattice Max &amp; Current Sugar Levels C Assign Lattice Sugar Growth Rates D Create The Agents</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Time Progression</td>
<td>4 End Simulation</td>
<td>2 Assign Agent Parameters</td>
<td>A Step?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Assign Agent Parameters</td>
<td>3 Time Progression</td>
<td>1 Build the Environment</td>
<td>A Assign Start Coordinates B Assign Metabolism C Assign Life Span D Assign Vision Randomly Assign Agent Number F Assign Initial Sugar Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Wait For Step</td>
<td>2 Prepare for Next Time Step</td>
<td>2 Prepare for Next Time Step</td>
<td>A Step Done?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Prepare for Next Time Step</td>
<td>1 Wait for Step</td>
<td>1 Wait for Step</td>
<td>A Grow Lattice Sugar B Increment Time Step C Reset Agent Numbers</td>
<td>A Call Number of Agents</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 End Simulation</td>
<td>3 Time Progression</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This conceptual model is based on the Sugarscape simulation in chapter 2 of “Growing Artificial Societies” by Epstein & Axtell, 1996.

Here all of the environmental arrays are assigned and the agent constructs are created.

Here the agent parameters are assigned according to chapter 2 of “Growing Artificial Societies” by Epstein & Axtell, 1996.

The block waits for all of the agents to execute before advancing the clock to the next step.

This state conducts time step and simulation maintenance and collects data.

This describes the behavior of the agents. See the Agent page for more information.

This represents the end of the simulation’s execution.
### Figure 38: Sugarscape CM4S Diagram Page 1 - Decision Shape Data

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Transitions To</th>
<th>Behavior</th>
<th>Pseudo Code - Condition</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variable</th>
<th>Sequence within Block</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision A</td>
<td>Step Done? 1</td>
<td>Wait For Step</td>
<td>2 Prepare for Next Time Step</td>
<td>Full of the agents currently running in the simulation have executed then prepares for the next time step</td>
<td>$(K \cdot \text{Current Active Agent} &gt; \text{Number Agents})$ then transitions to 2 Prepare for Next Time Step</td>
<td>EVERY change in $K \cdot \text{Current Active Agent}$</td>
<td>On Entry</td>
<td>$K \cdot \text{Current Active Agent}$</td>
<td>$\text{Number Agents}$</td>
<td>1</td>
<td>Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996</td>
</tr>
<tr>
<td>Decision A</td>
<td>Step? 3 Time Progression</td>
<td>4 End Simulation</td>
<td></td>
<td>Full stopping line has been reached then stops simulating the simulation</td>
<td>$(L \cdot \text{Current Time} = \text{Stop Time})$ then transition to 4 End Simulation</td>
<td>Change of $L \cdot \text{Current Time}$</td>
<td></td>
<td></td>
<td>$\text{1 L Current Time 1 N Stop Time}$</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Displayed Text</td>
<td>Master Name</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Pseudo Code - Function</td>
<td>Pseudo Code - Update</td>
<td>Pseudo Code - Start</td>
<td>Pseudo Code - Stop</td>
<td>Sequence within Block</td>
<td>Variables</td>
<td>Purpose</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------</td>
<td>------------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>----------------------</td>
<td>-----------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>A. Count Number of Agents</td>
<td>Recorder</td>
<td>2. Prepare for Next Time Step</td>
<td>Count the number of agents for each time step.</td>
<td>Count(1-33, Agent)</td>
<td>On Entry</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Obtain performance data concerning the number of agents to be analyzed later.</td>
<td></td>
</tr>
<tr>
<td>Displayed Text</td>
<td>Master Name</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Basis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Lattice X-coord</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array describing the x-coordinates (1-50) of each lattice location in the environment. Size=2500.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B Lattice Y-coord</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array describing the y-coordinates (1-50) of each lattice location in the environment. Size=2500.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C Lattice Max Sugar</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array describing the maximum sugar level (0-4) of each lattice location in the environment. Size=2500.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21-22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D Lattice Cur Sugar</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array describing the current sugar level of each lattice location in the environment. Size=2500.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21-22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E Lattice Growth Rate</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>A value describing the growth rate of the sugar per time step.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21-22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F Agent Metab. Range</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array of values describing the range of metabolisms for all of the agents.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H Agent Life Span Range</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array of values describing the range of life spans for all of the agents.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I Number Agents</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>The number of agents in the simulation.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996 p. 21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K Current Active Agent</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>A number describing the agent that is currently active. Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M Agent Ini Sugar Range</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>An array of values describing the range of initial sugar endowed to the agents.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N Stop Time</td>
<td>Variable</td>
<td>1. Sugarscape Conceptual Model</td>
<td>The value of how long the simulation is to run before being stopped.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
progression through time. The CM4S Diagram indicates the Sugarscape Simulation approach that advances the model time step once all of the agents have executed in the current time step. Recall that when constructing and/or reviewing a CM4S Diagram is that there are many ways to represent a system. This is just one representation. As a modeler, creating a good abstraction and representation using the CM4S Diagram is the art of modeling. As an evaluator of a CM4S Diagram, the goal is to determine the sanctionability of the abstraction and representation of the conceptual model.

Note the series of stacked blocks in the Agent block of the lower area in the Time Progression block. This represents multiple copies of a block that has the same basic internal structure. All agents in the Sugarscape simulation are structurally the same but differentiated by the unique variable values associated with each agent. Since the CM4S Diagram is a representation of a conceptual model of a simulation, not a simulation, the CM4S Diagramming Technique takes advantage of variables and similar structures to reduce the amount of documentation required. Furthermore, when a block’s title is underlined as in the Agent block, it means that its internal structure is represented on another page. The details of the Agent block is discussed later.

There are two other features of not within the Time Progression block. The first is an instance of the decision shape called Stop. This decision belongs to the Time Progression block and updates after every time step (see Figure 38) to determine if it is time to pass control to the End Simulation block. The End Simulation block represents the end of the simulation. The second feature is the first instance of the recorder shape within the Prepare for Next Time Step block called Col Number of Agents. The primary function of this recorder is to collect data on the agents in the model at a particular time step and to document exactly when this data are collected. Too often simulation designs fail to specify when data are collected. Thus, recorder shapes are very important in any CM4S Diagram.
The second page of the CM4S Diagram describes the activities, characteristics, and behaviors of the agents in the Sugarscape Simulation. The visual representation of the second page of the CM4S Diagram is shown in Figure 41. The corresponding data for the Action, Block, Decision, Recorder, and Variable Shapes on the second page are shown in Figures 42, 43, 44, 45, and 46, respectively.

As described earlier, all of the agents in the Sugarscape Simulation are structurally similar but have different variable values making each agent unique. Hence, only one page is needed to conceptually describe all of the agents in the simulation. Note the Sugarscape CM4S Diagram name of the highest level block on the page. In this case the Block’s name is Agent, but its number is “1.3.3.-”. The dash indicates that there are multiple instances of the Agent block. Each shape lower than, or contained within, the Agent block has a number starting with “1.3.3.-” and followed by their unique number/alphabet character. While executing this model an agent’s unique number would replace the dash to act as a place holder and determine how execution ties are broken. Following this convention means that the action shape named Scan North has the number “1.3.3.-.2.B”. Because all of the remaining shapes, features, and flows of control displayed on Page 2 have been introduced, the remainder of this section reviews the contents of the Agent block.

There are many elements within the Agent block that describe the conceptual model of the agents in the Sugarscape Simulation. First, there are many variables that uniquely define the agent. These variables capture key components of the agent including location, capabilities, and status. All of the shapes within the Agent block reference and update these variables.

The Sugarscape Simulation is turn based; each agent must wait their turn within a time step before executing. This design element is reflected in the initial block of the agent called Wait for Turn. In this block, data on the vision and metabolism of each agent is collected and the agent does not proceed further until the 1.K Current
Figure 41: Sugarscape CM4S Diagram Page 2 - Visual Representation

1.3.3. - Agent

1. Wait for Turn
   A. Vision at step
   B. Metab at step

2. Look for Sugar
   A. Scan This Location
   B. Scan North
   C. Scan South
   D. Scan East
   E. Scan West
   F. Temp Sugar Level

3. Move to Sugar
   A. Jump to Goal Loc

4. Collect Sugar
   A. Collect the Sugar

5. Consume Sugar
   A. Adjust sugar level
   B. Collect Sugar Level

6. Check Life Span
   A. Sugar Level >= 0?
   B. Can't Living

7. Make Replacement
   A. Randomly Reset Variables
   B. Jump to New Empty Location

8. Agent Turn Over
   A. Ready for the Next Agent

9. Agent Dies
   A. Remove Agent
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Pseudo Code - Function</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variables</th>
<th>Sequence within Block</th>
<th>Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Act1</td>
<td>A Scan This Location</td>
<td></td>
<td>Check the current position of the agent to determine if the agent needs to move</td>
<td>A Temp Sugar Level = UL Latte Left Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>B Scan North</td>
<td></td>
<td>Check the current position of the agent and determine if the agent needs to move</td>
<td>A Temp Sugar Level = UL Latte Left Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>C Scan South</td>
<td></td>
<td>Check the current position of the agent and determine if the agent needs to move</td>
<td>A Temp Sugar Level = UL Latte Left Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>D Scan East</td>
<td></td>
<td>Check the current position of the agent and determine if the agent needs to move</td>
<td>A Temp Sugar Level = UL Latte Left Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Jump to Goal Left</td>
<td></td>
<td>Move the agent to the goal position</td>
<td>A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Collect the Sugar</td>
<td></td>
<td>Remove the sugar from the current position</td>
<td>A Sugar Level = A Sugar Level + UL Latte Left Coord X Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Drink the Sugar</td>
<td></td>
<td>Consume the sugar</td>
<td>A Sugar Level = A Sugar Level + UL Latte Left Coord X Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Remove Agent</td>
<td></td>
<td>Remove the agent from the current position</td>
<td>A Agent Number = A Agent Number - 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Randomly Reset Variables</td>
<td></td>
<td>Perform random resets on the agent’s variables</td>
<td>A Variable = A Variable + UL Latte Left Coord X Coord X Coord + A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>A Ready for the Next Agent</td>
<td></td>
<td>Signal the next agent</td>
<td>A Signal Next Agent = I Signal Next Agent Value + A Signal Next Agent Value + A Signal Next Agent Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Act1</td>
<td>B Jump to New Empty Location</td>
<td></td>
<td>Move the agent to the new empty location</td>
<td>A Cur X Coord = I Cur X Coord Value + A Cur X Coord = I Cur X Coord Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>Based on “Growing Artificial Societies” by Epstein &amp; Axtell, 1996.</td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>From</td>
<td>To</td>
<td>Actions</td>
<td>Decisions</td>
<td>Recorders</td>
<td>Variables</td>
<td>Basis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>------</td>
<td>----</td>
<td>---------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td>-------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1.3.3.- Agent</td>
<td>1.3.3.- Agent</td>
<td>From</td>
<td>To</td>
<td>Actions</td>
<td>Decisions</td>
<td>Recorders</td>
<td>Variables</td>
<td>Basis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1 Wait for Turn</td>
<td>1. Wait for Turn</td>
<td>2. Look for Sugar</td>
<td>A. My Turn?</td>
<td>A. Vision at step, B. Metab at step</td>
<td>Record the vision and metabolism of the agent at that step and wait for its turn before executing further. Time step is based on “Growing Artificial Societies” by Epstein &amp; Axelrod, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 2 Look for Sugar</td>
<td>1. Wait for Turn</td>
<td>3. Move to Sugar</td>
<td>A. Scan This Location, B. Scan North, C. Scan South, D. Scan East, E. Scan West</td>
<td>A. Temp Sugar Level</td>
<td>Look in all four lattice directions to determine which possible lattice location has the most sugar. Based on “Growing Artificial Societies” by Epstein &amp; Axelrod, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 3 Move to Sugar</td>
<td>2 Look for Sugar</td>
<td>4. Collect Sugar</td>
<td>A. Jump to Goal Loc</td>
<td>After determining the best sugar lattice position, move to that position.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 4 Collect Sugar</td>
<td>3. Move to Sugar</td>
<td>5. Consume Sugar</td>
<td>A. Collect the Sugar</td>
<td>After moving to the new position, remove the sugar from the current lattice position.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 5 Consume Sugar</td>
<td>4. Collect Sugar</td>
<td>6. Check Life Span, 7. Agent Dies</td>
<td>A. Adjust sugar level</td>
<td>A. Sugar Level ≥ 0, B. Sugar Level &lt; 0</td>
<td>Reduce the sugar level due to metabolism, record the current level of sugar, and proceed accordingly based on the agent's current sugar level.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 6 Agent Dies</td>
<td>5. Consume Sugar</td>
<td>9. Agent Turn Over</td>
<td>A. Remove Agent</td>
<td>Remove the agent once it has died.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 7 Check Life Span</td>
<td>5. Consume Sugar</td>
<td>8. Make Replacement, 9. Agent Turn Over</td>
<td>A. Past Life Span, B. Compete Winning</td>
<td>Checking if the life span of the agent is exceeded and proceeding appropriately.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 9 Agent Turn Over</td>
<td>6. Check Life Span</td>
<td>A. Ready for the Next Agent</td>
<td>This represents the end of the agent turn and the start of the next agent’s turn.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Displayed Text</td>
<td>Master Name</td>
<td>Member Of Transitions To</td>
<td>Behavior</td>
<td>Pseudo Code - Condition</td>
<td>Pseudo Code - Update</td>
<td>Pseudo Code - Start</td>
<td>Pseudo Code - Stop</td>
<td>Variable</td>
<td>Sequence within Block</td>
<td>Basis</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>--------------------------</td>
<td>----------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>----------</td>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>A</td>
<td>My Turn?</td>
<td>Decision 1 Wait for Turn</td>
<td>Look for Sugar</td>
<td>When it is the agent's turn to execute based on its number then begin executing</td>
<td>If (1.K.Current Active Agent = J.Agent Number) then transition</td>
<td>Every change of 1.K.Current Active Agent</td>
<td>On Entry</td>
<td>1.K.Current Active Agent J.Agent Number</td>
<td>3</td>
<td>This turn-based strategy is based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell 1996</td>
</tr>
<tr>
<td>B</td>
<td>Can't Move?</td>
<td>Decision 6 Check Life Span</td>
<td>9 Agent Turn Over</td>
<td>If the agent has not exceeded their life span then the agents turn is over</td>
<td>If (1.L.Current Time - F.Start of Life ≤ E.Life Span) then transition to 9.Agent Turn Over</td>
<td>On Entry</td>
<td>1.L.Current Time F.Start of Life E.Life Span</td>
<td>2</td>
<td>Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell 1996</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Past Life Span</td>
<td>Decision 6 Check Life Span</td>
<td>8 Make Replacement</td>
<td>If the agent has exceeded their life span then insert a replacement agent for them</td>
<td>If (1.L.Current Time - F.Start of Life &gt; E.Life Span) then transition to 8.Make Replacement</td>
<td>On Entry</td>
<td>1.L.Current Time F.Start of Life E.Life Span</td>
<td>1</td>
<td>Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell 1996</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Sugar Level ≥ 0?</td>
<td>Decision 5 Consume Sugar</td>
<td>6 Check Life Span</td>
<td>If the sugar level of the agent is greater than or equal to zero then check whether the agent has exceeded their life span</td>
<td>If (G.Sugar Level ≥ 0) then transition to 6.Check Life Span</td>
<td>On Entry</td>
<td>G.Sugar Level</td>
<td>3</td>
<td>Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell 1996</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Sugar Level &lt; 0?</td>
<td>Decision 5 Consume Sugar</td>
<td>7 Agent Dies</td>
<td>If the sugar level of the agent is less than zero then the agent dies</td>
<td>If (G.Sugar Level &lt; 0) then transition to 7.Agent Dies</td>
<td>On Entry</td>
<td>G.Sugar Level</td>
<td>4</td>
<td>Based on &quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell 1996</td>
<td></td>
</tr>
<tr>
<td>Displayed Text</td>
<td>Master Name</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Pseudo Code</td>
<td>Variables</td>
<td>Sequence within Block</td>
<td>Purpose</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------</td>
<td>-------------</td>
<td>-----------</td>
<td>----------------------</td>
<td>---------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Collect Sugar Level</td>
<td>Record</td>
<td>5. Consume Sugar</td>
<td>Record the amount of sugar for each agent at each time step.</td>
<td>Record the value of G. Sugar Level</td>
<td>G. Sugar Level</td>
<td>2</td>
<td>Obtain performance data concerning the sugar level of agents to be analyzed later.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Vision at step</td>
<td>Record</td>
<td>1. Wait for Turn</td>
<td>Record this agent's current vision at each time step.</td>
<td>Record the value of C. Vision</td>
<td>C. Vision</td>
<td>1</td>
<td>Obtain performance data concerning the vision of agents to be analyzed later.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B Metab at step</td>
<td>Record</td>
<td>1. Wait for Turn</td>
<td>Record the agent's current metabolism at each time step.</td>
<td>Record the value of variable D. Metab</td>
<td>D. Metab</td>
<td>2</td>
<td>Obtain performance data concerning the metabolism of agents to be analyzed later.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Displayed Text</td>
<td>Master Name</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Basis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Cur X-coord</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current x-coordinate of this agent</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B Cur Y-coord</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current y-coordinate of this agent</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C Vision</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current vision of the agent, which defines how far it can see and move.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D Metab</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current metabolism of the agent. This is how much sugar they consume in one time step.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E Life Span</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current life span of the agent. Exceeding these number of steps will result in the agent dying.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F Start of Life</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is when the agent's life began.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G Sugar Level</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is current sugar level of the agent.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H Goal X-coord</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is goal x-coordinate of the agent.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I Goal Y-coord</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is goal y-coordinate of the agent.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J Agent Number</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the agent's unique number.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A Temp Sugar Level</td>
<td>Variable</td>
<td>2. Look for Sugar</td>
<td>This is a temporary variable to store the maximum amount of sugar encountered so far.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K Cur Lattice Position</td>
<td>Variable</td>
<td>1.3.3.- Agent</td>
<td>This is the current lattice position of the agent.</td>
<td>&quot;Growing Artificial Societies&quot; by Epstein &amp; Axtell, 1996.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Active Agent variable equals the Agent Number variable.

When allowed to proceed, the agent executes a series of activities defined by the Sugarscape Simulation rules. First, the agent proceeds to the Look for Sugar block where a series of actions are performed to look for a lattice position (within their visual range) that has the most sugar. These series of actions could be represented in one action shape, but are detailed here to better highlight the sequence of these actions. This block not only represents a "state" of the agent, but also defines the algorithm and temporary variable used by the agent to find the most sugar. First, the agent scans their location and then the agent sequentially scans north, south, east, and west, as per the Sequence within Block property of these actions. The Pseudo Code - Function property of each action effectively and easily conveys how the relatively complex scanning task is executed without being overly technical or formally complete.

After all of the actions in the Look for Sugar block have executed, control is passed to the Move to Sugar block where the agent moves to the neighboring location with the most sugar. The agent collects and consumes the sugar dictated by their metabolism; the net level of sugar obtained by the agent is recorded. At this point two related conditions are evaluated: is the sugar level of the agent greater than or equal to zero or is it less than zero? If it is less than zero, then the agent dies and is removed from the simulation. If the sugar level of the agent is greater than or equal to zero, then another series of activities occur before the agent’s turn is over.

Evaluating two decisions determines how to proceed based on the agent’s life span and current length of life in the Check Life Span block. If the agent is past its life span then a replacement agent is created and the old agent is removed. Once the replacement agent is created or the agent is within its life span the agent’s turn is over and the agent signals the next agent’s turn via adjusting the 1.K Current Active Agent variable.
This section highlighted some of the key aspects of the CM4S Diagram representation of the Sugarscape simulation. One should spend time reviewing the diagram to completely understand the Sugarscape scenario, the Sugarscape Simulation, and how the CM4S Diagram represents the simulation. A key purpose of the diagram is to provide a wealth of information that is often either not conveyed or not conveyed well with traditional model description techniques. Thus, any written description is likely to be inherently incomplete when compared to the CM4S Diagram. Walking through this example is the best way to fully understand the CM4S Diagram's functionality.

8.4.3 The Effectiveness of the Sugarscape CM4S Diagram

The CM4S Diagramming Technique can represent the conceptual model of an ABM simulation. It captures and documents the key activities, behaviors, timing, statistics, and characteristics of the conceptual model of the simulation and documents the “why” information of each shape for sanctioning purposes. However, note that even for this relatively simple simulation, a significant amount of work was required to actually build the CM4S Diagram. This is not a drawback of the CM4S Diagram, but is a reflection of the difficulty and complexity required to truly document the execution and sanctioning criteria of a simulation model. A simple written description only begins to capture the simulation and building an effective simulation that executes in a computer is not a simple task.

Further evaluation of the effectiveness of the CM4S Diagram is discussed in Part III: The Evaluation of the CM4S Diagram, where two ABM simulation projects (military and supply chain examples) are constructed and analyzed using the CM4S Diagram.
8.5 Capabilities and Conclusions

The fundamental goal of the CM4S Diagram is to advance ABM as an analysis tool. The CM4S Diagram satisfies all 10 requirements of the ABM diagramming technique solution concept. The basic syntax of the CM4S Diagram was described and used to build a conceptual model of Epstein and Axtell’s Sugarscape Simulation. The CM4S Diagram represents the first diagramming technique designed specifically for the effective representation, construction, and sanctioning of ABM computer simulations based on identified needs in the ABM modeling field and simulation modeling philosophy.
III. The Evaluation of the Conceptual Model for Simulation Diagram
In Chapter 8 of Part II, the CM4S Diagram is presented in its final version. However, as with any newly developed product, the CM4S Diagram went through several revisions and evaluations prior to the final version. This part of the document reviews the evolution of the CM4S Diagram through two rounds of improvement studies where a version of the CM4S Diagram was developed, used to construct an ABM simulation, evaluated for flaws and overall effectiveness, and then improved. The revisions resulting from these studies represents a significant improvement and refinement of the CM4S Diagram. Furthermore, constructing unique ABM simulations from various domains and levels of complexity highlights the robustness of the CM4S Diagram as well as the successful application of the theories, principles, and philosophies developed and discussed in Part I.
In this chapter the initial prototype of the CM4S Diagram is described and evaluated by constructing an ABM simulation that replicates Scenario 1 of Champagne [26].

Bay of Biscay ABM Simulation as a proof of concept study. One of the primary reasons for selecting this scenario is that it can be represented as a complex ABM Simulation and has had real implications on past, present, and even future military strategies. The Bay of Biscay Scenario took place in WWII when German Uboats, or submarines, would cross the Bay of Biscay (as shown in Figure 47) from captured French Ports to disrupt logistical forces in the North Atlantic that were supporting the Allied war effort. To combat this the Allies sent Airplanes to search the Bay for surfaced Uboats and destroy them. Although on the surface this scenario may seem relatively simple, it demonstrates complexity due to technological developments, possible searching strategies, and various tactical policies. Uboat search operations helped develop the field of Operations Research and since that time several books were written and models developed to gain further insight into this scenario [75, 114].

Another reason for selecting this simulation was the amount of available documentation. Many aspects required to reproduce the simulation are included because a significant portion of Lance Champagne’s dissertation was spent describing the Bay of Biscay proof-of-concept.
of Biscay ABM Simulation. As discussed earlier, this amount of documentation is rarely provided for published simulation projects. With abundant details, one could more effectively develop an example within the CM4S Diagram Prototype as a proof of concept, describe the functionality of the prototype, more effectively show the usefulness and application of the CM4S Diagram Prototype, and further improve the CM4S Diagram Prototype and identify shortcomings. Ultimately, deciding to reproduce another simulation puts more emphasis on the CM4S Diagram Prototype and less emphasis on creating an entirely new simulation.

Finally this ABM Simulation leads to a more meaningful discussion regarding the sanctionability of the reproduced ABM Simulation using the CM4S Diagram Prototype. In his dissertation, Champagne provides both conceptual and operational benchmarks that can help gauge the sanctionability of the ABM Simulation. For example, Champagne provides justification and documentation regarding the logic
of how Aircraft searched for Uboats and how the Uboats responded to the Aircraft. Using this documentation a better conceptual model can be constructed and some claims about the sanctionability of the conceptual model can be made. Also, Champagne provides key operational measures such as Number of Sightings and Number of Kills that can be used to operationally sanction our simulation. Being able to make these comparisons will further support the effectiveness and justification of the CM4S Diagram Prototype. It should be made clear that this chapter does not directly compare this simulation with the real system. It compares a new Bay of Biscay model with another well documented and sanctioned simulation.

9.1 The Initial Design of the CM4S Diagram Prototype

Before proceeding with the description of the simulation it is worthwhile to describe the initial design of the CM4S Diagram Prototype. In the following subsections the fundamental characteristics of the CM4S Diagram Prototype are described along with the basic structure. The purpose of describing these elements is to aid in the understanding of the CM4S Diagram Prototype as it existed during this simulation development effort.

9.1.1 Fundamental Characteristics

A prototype design of a new ABM modeling tool was developed to have the appropriate capabilities to meet the defined needs. This new ABM tool, the Conceptual Model for Simulation Diagram Prototype or CM4S Diagram Prototype, is based primarily on the structure of Statecharts because of their ability to visually represent structures and relationships of complex systems [46]. However, the CM4S Diagram Prototype only borrows the fundamental structural pieces from Statecharts for visual representation purposes and does not incorporate many of the definitions that
compose and define a Statechart. For example, the CM4S Diagram Prototype uses arcs to represent the movement from one block of activities to another, but unlike Statecharts arcs, CM4S Diagram Prototype arcs do not determine when to transition among blocks or states.

The CM4S Diagram Prototype also incorporates some of the basic shapes and properties of Flowcharts for the purpose of simplifying and providing flexibility to those using the CM4S Diagram Prototype. For example, within each block or state, there are a set of actions, interactions, or decisions shapes that define the activities occurring when the modeled entity is in that block or state. Without requiring the users to specifically define events and the transitions to new states, the CM4S Diagram Prototype allows some flexibility to redefine the activities without needing to completely build a new model. Having simple shapes representing different activities within a block allows evaluators or builders to quickly determine what is occurring in that block without the need for dealing with the detailed logic of the model. Incorporating these simple shapes allows users to spend more time focusing on behaviors and activities and less time worrying about how best to design the logic of the computer simulation. Even with these changes, the CM4S Diagram Prototype remains formal and provides enough information to translate the diagram into a simulation.

The CM4S Diagram Prototype incorporates visual aspects to aid the user in defining and understanding the hierarchical structure, relationships, and activities that occur in each block or state. The CM4S Diagram Prototype also incorporates a database of properties for each shape to provide details necessary to properly sanction and build the simulation. For example, an Action shape has five properties that define its relationship to other shapes, the behavior of the real system it is trying to mimic, pseudo code to help to translate that real system behavior into a simulation, and a reference to the source that describes that behavior occurring in the real system. With these properties, the CM4S Diagram Prototype contains much of the
“why” information that is often lacking in other modeling techniques. This defines why the CM4S Diagram Prototype can be translated into a simulation and provide enough information for evaluators to review the sanctionability of the simulation.

9.1.2 Basic CM4S Diagram Prototype Structure

There are five shapes that are used to construct the CM4S Diagram Prototype. The first shape is the Block, visually represented as a rectangle, which defines a collection of shapes. The Block is the fundamental shape that describes the hierarchical relationship and structure of the system being simulated and it can be used to describe that state of the system at any point in time. For example, a Block called Environment could describe the collection of shapes that describe the environment within which the agents operate. Activities or behaviors within a Block could be defined to occur “synchronously” or “concurrently,” which is a fundamental property of Statecharts. Actions within a Block could define the creation of more instances of Blocks. For example, in a Build Agents Block there could be an action that creates 40 agents.

The CM4S Diagram Prototype has shapes to describe the micro-level behaviors that occur within the model. The Action shape describes an activity that changes an entity or variable and is visually represented with a block arrow. An example of an Action may be the consumption of food or moving to a new location. A special case of the Action shape is called the Interaction Shape, visually represented by a circle. The Interaction Shape defines the exchange of information between agents. For example, an Interaction could be the communication of the location of an important resource between two agents. There are two fundamental reasons the existence of the Interaction shape. First, when evaluating or building an ABM simulation it is important to understand the interactions that occur between agents; making it a special shape makes the identification of this behavior easier. Second, the interaction
between agents within a simulation often requires special programming attention; explicitly capturing this behavior can aid in translating the CM4S Diagram Prototype into a simulation. The final shape that describes micro-level behavior is the Decision shape, visually represented with a diamond. A Decision shape examines a set of conditions and determines whether to transition to another block. For example, a Decision could be to transition from a Move Block to a Stop Block if an obstacle is encountered by some moving agent.

The final shape used in the CM4S Diagram Prototype is the Information Shape, visually represented with an horizontally elongated circle. The Information shape provides information referenced by the Behavior Shapes. For example, an Information shape could represent the simulation clock that a Decision shape references to see if the simulation should terminate. An Information shape could represent the schedule by which Agents are created or destroyed. The Information shape is included in the CM4S Diagram Prototype to convey key information that the simulationist does not want to represent using Blocks, Actions, Decisions, and Interactions.

These shapes, also have specific properties. A Block shape has five properties. The From property describes which blocks transition to it. The To property describes blocks that can transition to it. The Decisions, Actions, and Interactions properties describe the respective shapes that are directly contained within the block. The fundamental properties of the Block shape describes its relationships to other shapes.

The Behavior Shapes (Action, Decision, and Interaction) also have five properties that are closely related to each other. The first property is the Member Of property, which describes the direct block that the shape belongs to. The next property describes the activity being performed. For the Action shape, the Impacts Decision property describes any Decision shape that is being impacted by the action. For the Decision shape, the Transitions To property describes blocks that can be transitioned to as a result of the decision. For the Interaction shape, the Interacts With property
Table 4: CM4S Diagram Prototype Shape Descriptions

<table>
<thead>
<tr>
<th>Shape</th>
<th>Name</th>
<th>Properties</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>From To Decisions Actions Interactions</td>
<td>A collection of actions, decisions, interactions, and/or information. Can also be a collection of several blocks. Describes the state of the simulation or entity at a point in time.</td>
<td>Initialization, Building, Searching, Crossing</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Member Of</td>
<td>Impacts Decision, Behavior</td>
<td>An activity describing the change of an entity or variable.</td>
<td>Set Coordinates, Travel Home, Destroy Agent</td>
</tr>
<tr>
<td>Decision</td>
<td>Member Of</td>
<td>Transitions To Behavior</td>
<td>A condition that describes when to transition to another Block as well as what Block to transition to.</td>
<td>Interrupt?, Go Home?, Begin Simulation?</td>
</tr>
<tr>
<td>Interacts</td>
<td>Member Of</td>
<td>Interacts With Behavior</td>
<td>A special Action describing the exchange of information between agents.</td>
<td>Look for Neighbor, Kill Enemy, Calculate Distance to Neighbor</td>
</tr>
<tr>
<td>Information</td>
<td>Member Of</td>
<td>Behavior Source</td>
<td>A parameter or constant</td>
<td>Clock, Initial Number of Agents, Schedule</td>
</tr>
</tbody>
</table>

describes the blocks or shapes that the interaction impacts. The final three properties for the Behavior Shapes are exactly the same. The Behavior property describes a behavior of the real system that the shape is attempting to mimic. The Pseudo Code property describes the pseudo code intended to facilitate translating the behavior into the simulation. Finally, the Source property is a reference to a source that provides the justification for that behavior.

The final shape, the Information shape, has three properties; Member Of, Behavior, and Source. Each of these properties are as previously defined for Behavior Shapes. A complete summary of the shapes, their visual representation, properties, definition, and some examples are shown in Table 4. Although the CM4S Diagram Prototype is still in the early stages of development, the fundamental shapes and the structure associated with Statecharts can effectively provide rich descriptions of most ABM simulations. As a proof of concept, to further describe the functionality of the CM4S Diagram Prototype, and to show CM4S Diagram Prototypes effective-
ness at sanctioning, a CM4S Diagram Prototype will be developed for the Bay of Biscay ABM Simulation Scenario 1. Furthermore, a template of the CM4S Diagram Prototype was developed using Microsoft Visio 2007 that will have all of the CM4S Diagram Prototype shapes and properties.

9.2 Scenario 1 of the Bay of Biscay ABM Simulation

In Champagne’s Bay of Biscay ABM Simulation there are two scenarios that approximately correspond to the developments that occurred in WWII. For simplicity only Scenario 1 is replicated. However, the changes presented in Scenario 2 could easily be incorporated in the CM4S Diagram Prototype and the resulting ABM Simulation. In the next few paragraphs, a general description of the modeled Scenario 1 is provided as given in Champagne’s Dissertation. For more detailed information see [26].

Scenario 1 takes place over the six month period from October 1942 to March 1943 and is proceeded by a 12 month warm-up period where only the Uboats travel between the North Atlantic and French Ports. After the 12 month warm-up period the Aircraft take off from England and search for the Uboats in a 50 NM by 50 NM search area using the Modified Barrier Search Pattern. The Aircraft are randomly assigned to search 200 NM x 350 NM area that is not within a 100 NM of the French coast and are schedule to take off uniformly throughout a 24 hour period. Furthermore, they must be at the base for a 12 hour period before taking off again. Throughout the simulation there are 19 Aircraft such that the number of sortie hours approximately correspond to the historical number of sortie hours flown during that six month period. Aircraft travel at 120 knots and search for Uboats for 7 hours and then return to the base. If at anytime the Aircraft detects a Uboat and fires, the Aircraft will immediately return to the base.

During this Scenario the Uboats observe the maximum submergence and nighttime surfacing policy. Day is defined as the time between Nautical Dawn and Nautical Dusk.
and Night is the time between Nautical Dusk and Nautical Dawn. While on the surface the Uboats will scan for Aircraft and upon detection will submerge until it is nighttime. On the surface the Uboats travel at 10 knots and while submerged they travel at 2.5 knots. They can only travel 3 hours submerged before they must surface to recharge their batteries if they wish to continue moving. For every hour traveled on the surface, the batteries have one more hour of submerged travel time up to a total of 3 hours of submerged travel. When they are within a 100 NM of the French coast they will travel on the surface because they will have German Aircraft support. During the warm up period the 70 Uboats are randomly distributed across the Bay of Biscay and are set to either head towards their home port or the North Atlantic. Once the warm up period is over, at the beginning of each month a scheduled number of Uboats enters the simulation at their home port and heads toward the North Atlantic. While in the North Atlantic the Uboats have 30 days worth of provisions and have a 25% chance of extending their time in the North Atlantic by another 30 days. Once the Uboats reach their port they will uniformly depart again after 25 to 40 days.

For detection both the Aircraft and the Uboats used the Inverse Cube Law, which incorporates the use of multiple sensors to create a positive probability of detection regardless of the distance away from the target. Although the sensors and their range was not directly provided by Champagne for the Aircraft and the Uboats, Champagne did reference [75, 114] which gave the breakdown of the sensors. Based on those references, the sweep width and detection probability was calculated. Furthermore, when an Aircraft detects a Uboat it has a 0.02 probability of killing the Uboat regardless of the time of day.

Although many details of the simulation that were given by Champagne are beyond what is typically encountered in the modeling literature, there were still many aspects of his simulation that were not completely clear. There are several reasons for this lack of clarity. First, Champagne primarily relies on written descriptions to
convey the logic and activities of the simulation supplemented with logic flowcharts. This means of communication can be fairly confusing if the sentence is not worded effectively. For example, saying Uboats submerge after detecting an Aircraft and then resurface at night could be interpreted several different ways. Do they resurface immediately if it is night time? Do they resurface after 3 hours and it is still night time? Unless the written description is in a clear structure and formalism there is ambiguity. This problem with written descriptions further indicates that some sort of formal description of the model is required to support attempts to reproduce a model or even interpret the model’s sanctionability as part of the scientific or engineering process.

Closely related to the lack of formal description is that the written description provides no structure that allows the author to describe all of the necessary details in order for others to reproduce their work. After spending a significant amount of time building a simulation it is likely that small, yet critical, pieces of the simulation are left out of written descriptions. For example, Champagne did not include critical information regarding the detailed sensor data in his descriptions of the model. In this particular case I was able to find the sensor data that I believe he used, however I cannot be 100% certain. It could be conjectured that if a more formal description was supplied, then some of these unintentional omissions would be averted.

Overall, the description of the simulation given in Champagne’s Dissertation provided most of the information needed in order to reproduce both the conceptual and operational aspects of the simulation. When something was encountered that was unclear I attempted to review any references given for clarification. However, if no reference were given reasonable assumptions were made and were recorded in the CM4S Diagram Prototype that was developed.
9.3 Bay of Biscay CM4S Diagram Prototype Description

Based on Champagne’s Dissertation, a CM4S Diagram Prototype was developed to serve as a medium between the real world system and the simulation model. The CM4S Diagram Prototype is the descriptive and definitional abstraction of the real system that serves as a translator between the infinitely complex real world and the finitely defined simulation. This section describes the CM4S Diagram Prototype’s functionality and explains how the CM4S Diagram Prototype aids in building better simulation models.

A CM4S Diagram template was built using Microsoft Visio 2007. The CM4S Diagram Prototype of the Bay of Biscay ABM Simulation was built using this basic template. This Visio file is available to download at www.cm4sdiagram.com. The reason for selecting Visio to develop the CM4S Diagram Prototype was the ease of use, the ability to run reports to obtain the properties of all the shapes, and the general industry knowledge of Visio.

There are four different sheets or views of the simulation within the CM4S Diagram Prototype file. The first view is the “Bay of Biscay Model” shown in Figure 48. This view shows the model-level abstraction of the simulation and includes initialization blocks and execution blocks. Also, this view shows the major actions taken to initialize and run the simulation. The second view called “Environment” shows the environment level of abstraction and is in Figure 49. This includes the Uboats and the Aircraft that exist in the environment, their basic interaction, and the progression between night and day. Note that the “Environment” view is part of the “Bay of Biscay Model”. Having separate views aids the user in viewing different levels of abstraction in the CM4S Diagram Prototype. The final two views are the “Uboat” (Figure 50) and “Aircraft” (Figure 51) views, which display the detailed behaviors of each agent.

Collectively these views display hierarchical structure of the actions that take
Figure 48: CM4S Diagram Prototype - Bay of Biscay Model View
Figure 49: CM4S Diagram Prototype - Environment View
Figure 50: CM4S Diagram Prototype - Uboat View
Figure 51: CM4S Diagram Prototype - Aircraft View
place in the real world and the simulation. For example, in the Uboat view (Figure 50) there are many actions that can happen while the Uboat is crossing the Bay, such as they can be surfaced, submerged and moving, submerged and stopped, or be destroyed. Furthermore, the boundaries of the real world abstraction and the simulation are clearly seen. One can observe in the Environment view (Figure 49) that the agents involved are the Aircraft and Uboats and that the time of day plays a role. In the Model view (Figure 48) one can see the simulation details, the building of the environment, the creation of new agents, and other important simulation aspects that define when the simulation begins and ends.

Although the visual aspects of the CM4S Diagram prototype provides ample information, it does not provide enough information for the purposes of sanctioning this conceptual model or building the simulation. This is where the database of information/properties related to each shape fills in the details. For each shape in the CM4S Diagram Prototype there are a series of properties that further define the details of that shape. In the Visio file of the CM4S Diagram Prototype, whenever a shape is selected, a Shape Data Window opens and prompt the user to fill out the properties associated with that shape. For example, a screen shot of the CM4S Diagram Prototype in Visio shows the Shape Data Window for the “Depart Port” decision shape in the Uboat block in Figure 52. In the “Depart Port” shape there are five associated properties: Member Of, Transitions To, Behavior, Pseudo Code, and Source. The associated value of the Member Of properties is In_Port because this describes the block that the decision shape resides in. Furthermore, the Transitions To property is set to Cross_Bay because this describes the block transitioned to as the result of the decision. The Behavior property describes the behavior that the decision shape executes and in turn the Pseudo Code property lists the pseudo code that more precisely describes the behavior. Finally, the Source property describes the source of the behavior and in this case Champagne’s 2003 Dissertation is referenced.
For more details on all of the properties associated with each shape, see the earlier section discussing the CM4S Diagram Prototype design.

The remainder of this section describes in detail just the Uboat block of the CM4S Diagram Prototype. This detailed discussion illustrates how to read and interpret a CM4S Diagram Prototype. Although ideally done dynamically within the Visio file, all of the necessary information pertaining to the Uboat block is provided in this chapter. The visual of the Uboat block is shown in Figure 50. The list of action properties, block properties, decisions properties, and interaction properties of the shapes within the Uboat block are shown in Figures 53, 54, 55, and 56, respectively.

Based on the real world description of the Bay of Biscay and the provided behavior of the Uboats, a conceptual model was developed of the Uboat behavior that captures the desired level of abstraction, provides documentation of that conceptual model, and is designed for simulation. Notice in the description that there are 70 Uboats. However, in the CM4S Diagram Prototype there is only one Uboat block. This does not mean that all of the Uboats are homogeneous or that there is only one Uboat. In fact, the Uboats are heterogeneous because each Uboat is assigned to have different home ports and destinations in the North Atlantic (this aspect is defined in the Initialization block). However, while some of the parameters of the Uboats are
187

Figure 53: Uboat Action Shape Properties


### Figure 54: Uboat Block Shape Properties

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>1 From</th>
<th>2 To</th>
<th>3 Decisions</th>
<th>4 Actions</th>
<th>5 Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>In_Port</td>
<td>Cross_Bay</td>
<td>Cross_Bay</td>
<td>Set Departure</td>
<td>Depart Port</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>Cross_Bay</td>
<td>In_Port, In_Atlantic</td>
<td>In_Port, In_Atlantic</td>
<td>Goal Reached</td>
<td>Set Coal</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>On_Surface</td>
<td>Cross_Bay</td>
<td>Submerged</td>
<td>Surface Policy</td>
<td>Travel, Charge Battery</td>
<td>Scan for Aircraft, Bomb by Aircraft</td>
</tr>
<tr>
<td>Block</td>
<td>Submerged</td>
<td>On_Surface</td>
<td>On_Surface</td>
<td>Surface Policy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>Stopped</td>
<td>Moving</td>
<td></td>
<td>Travel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>Moving</td>
<td>Submerged</td>
<td>Stopped</td>
<td>Battery Life</td>
<td>Travel, Deplete Batteries</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>Destroyed</td>
<td>On_Surface</td>
<td></td>
<td></td>
<td>End Uboat</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>In_Atlantic</td>
<td>Cross_Bay</td>
<td>Cross_Bay</td>
<td>Set Departure</td>
<td>Depart Atlantic</td>
<td></td>
</tr>
</tbody>
</table>
### Figure 55: Uboat Decision Shape Properties

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>1 Member Of</th>
<th>2 Transitions To</th>
<th>3 Behavior</th>
<th>4 Pseudo Code</th>
<th>5 Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>Depart Port</td>
<td>In_Port</td>
<td>Cross_Bay</td>
<td>After 35 to 40 days in port, depart to North Atlantic</td>
<td>If Set_Departure==Now then [Cross_Bay].</td>
<td>Champagne 2003</td>
</tr>
<tr>
<td>Decision</td>
<td>Surface Policy</td>
<td>On_Surface</td>
<td>Submerged or Destroyed</td>
<td>If aircraft is detected then submerge. If bombed by aircraft then the Uboat is destroyed. If there is daylight then submerge.</td>
<td>If aircraft==detected or daylight==true then [Submerged]; if destroyed==true then [Destroyed];</td>
<td>Champagne 2003</td>
</tr>
<tr>
<td>Decision</td>
<td>Battery Life</td>
<td>Moving</td>
<td>Stopped</td>
<td>If the Uboat’s batteries run out, then the Uboat must stop moving but remained Submerged</td>
<td>If batteries_depleted==true then [Stopped];</td>
<td>Champagne 2003</td>
</tr>
<tr>
<td>Decision</td>
<td>Surface Policy</td>
<td>Submerged</td>
<td>On_Surface</td>
<td>If Uboats is submerged because aircraft was detected, then the Uboat will not surface until the next day/night. Otherwise, the Uboat will surface when there is no daylight.</td>
<td>If aircraft_detected==false &amp; daylight==false then [On_Surface]; if next_day==true &amp; daylight==false then [On_Surface];</td>
<td>Champagne 2003</td>
</tr>
<tr>
<td>Decision</td>
<td>Goal Reached</td>
<td>Cross_Bay</td>
<td>In_Atlantic</td>
<td>Once the goal location is reached, the Uboat waits for a period of time in either the Atlantic or in their Home Port.</td>
<td>If from_port==true &amp; current_location==goal then [In_Atlantic]; If from_port==false &amp; current_location==goal then [In_Port];</td>
<td>Champagne 2003</td>
</tr>
<tr>
<td>Decision</td>
<td>Depart Port</td>
<td>In_Atlantic</td>
<td>Cross_Bay</td>
<td>Uboats leave port with approximately 30 days of supplies and there is a 25% chance that they will be able to extend their stay by 60 days.</td>
<td>If Set_Departure==Now then [Cross_Bay];</td>
<td>Champagne 2003</td>
</tr>
</tbody>
</table>
### Figure 56: Uboat Interaction Shape Properties

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>1 Member Of</th>
<th>2 Interacts With</th>
<th>3 Behavior</th>
<th>4 Pseudo Code</th>
<th>5 Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction</td>
<td>Scan for Aircraft</td>
<td>On_Surface</td>
<td>Aircraft</td>
<td>Scan for aircraft using the inverse Cube Rule with 3 devices (Model: 16m, N_max=15m, V_max=2m). Weight cap: $\text{weight}(15^2+15^2+2)$.</td>
<td>For each Aircraft within W $\text{(prob_detect=1/1-e^{-}(W^2)/(2n^2))}$, if $\text{Uniform}(0,1)$, if prob_detect then (aircraft_detected=true).</td>
<td>Champagne 2003, McCue 1980</td>
</tr>
<tr>
<td>Interaction</td>
<td>Bombed by Aircraft</td>
<td>On_Surface</td>
<td>Aircraft</td>
<td>If a Uboat is hit with a bomb from an aircraft, then it is destroyed</td>
<td>If bombed==true then (destroyed=true).</td>
<td>Champagne 2003</td>
</tr>
</tbody>
</table>
heterogeneous, their behavior logic is homogeneous. Therefore, we can represent all Uboats with a single conceptual block with generic parameters that can be adjusted for each individual Uboat.

For simplicity I will describe the behavior of a Uboat as starting at its home port and about to cross the Bay of Biscay in route to the North Atlantic. Therefore, we will begin in the In_Port block, which has one action shape, Set Departure, and one decision shape, Depart Port. While in this block the Uboat executes the Set Departure action, which sets a departure time to be 25 to 40 days into the future, and the Uboat evaluates whether it is time to depart based on the Depart Port decision shape. As described in the Depart Port decision shape properties, the Uboat transitions to the Cross_Bay block when the current simulation time is equal to the departure time.

When the Uboat transitions to the Cross_Bay block, it will immediately transition to the On_Surface block within the Cross_Bay block. This behavior is indicated by the circle with the arc pointing to the On_Surface block. Note that there are two shapes that are members of the Cross_Bay block which have precedence over all of the shapes within the On_Surface block, the Submerged and Destroyed blocks, which are at a higher level of aggregation. Further, even though the Uboat can be crossing the bay in many different states, once they reach their goal they will automatically transition to the In_Atlantic block. This idea of hierarchy is very important in the CM4S Diagram Prototype because it helps capture and convey complex behaviors. Within the Cross_Bay block the Set Goal action shape sets a goal location in the North Atlantic and the Goal Reached decision shape evaluates when that goal is reached and to then transition to the In_Atlantic block. If the goal has not been reached, then the internal blocks of the Cross_Bay continue to execute.

Within the On_Surface block there are several different shapes. The Charge Battery action shape is a reoccurring action shape that charges the batteries for up to 3 hours. The Travel action shape is a reoccurring shape that moves the Uboat towards
the goal at 10 knots. The Travel action shape updates the location, which is evaluated by the higher level Goal Reached decision shape. The Scan for Aircraft interaction shape is a reoccurring action that accesses all of the Aircraft in the simulation location and calculates whether the Uboat can detect any aircraft in their area. For more details, see the Uboat Interaction Shape Properties in Figure 56. The Bombed by Aircraft interaction shape checks to see if any Aircraft have recorded hitting the Uboat. Finally, the Surface Policy decision shape determines whether it is time to transition to the Destroyed block of the Submerged Block. If the Uboat has been hit, then it transitions to the Destroyed block. If the Uboat has spotted an Aircraft, or if the sun is up, then the Uboat will transition to the Submerged block. (Uboats preferred submerged operations as they were harder to locate by aircraft).

Within the Destroyed block there is one action block: End Uboat. The End Uboat action shape removes the Uboat from the simulation after it has been destroyed.

Inside the Submerged block, there is another level of blocks as well as the different Surface Policy decision shape. This Surface Policy decision shape evaluates when it is time to transition to the On_Surface block based on the surfacing policy set forth in Scenario 1. Once the Uboat has transitioned to the Submerged block it immediately transition to the internal Moving block. Thus, in this case, the Uboat is crossing the bay, submerged. Within the Moving block there are two action shapes and one decision shape. The Travel action shape is a reoccurring shape that moves the Uboat towards the goal at 2.5 knots. The Deplete Batteries action shape is also a reoccurring shape that depletes the battery charge. The Battery Life decision shape determines if the batteries have been completely depleted and then transitions the Uboat to the Stopped block. Within the Stopped block the Travel action shape moves the Uboat towards the goal at 0 knots.

When the Uboat reaches its goal coordinates the Uboat transitions to the In_Atlantic block. Within the In_Atlantic block the Set Departure action shape sets the time
at which the Uboat begins heading back toward its home port. The Depart Port decision shape evaluates whether the current time equals the departure time at which point the Uboat will transition to the Cross_Bay block. The goal coordinates at this time are the Uboat’s home port and not the North Atlantic.

Although this description of the Uboat’s CM4S Diagram Prototype behavior is at a relatively high level, describing it has provided a better understanding of how to read and interpret a CM4S Diagram Prototype. More detail about the Uboat and the entire Bay of Biscay simulation, and the “whys” for each behavior as it supported Champagne’s goals, is available in the CM4S Diagram Prototype file. The ability of the CM4S Diagram Prototype to provide both high and low levels of details further fills the need for evaluators of various level of expertise to understand the conceptual model of the simulation. Thus, the CM4S Diagram Prototype can be used as a tool for sanctioning and as a tool for verifying simulation performance.

9.4 Identified Improvement Areas for the CM4S Diagram Prototype

The Bay of Biscay proof of concept shows the CM4S Diagram Prototype useful as a tool for sanctioning and verifying that a simulation performs as intended. However, the need for several additions to the CM4S Diagram Prototype were apparent. These include:

1. An additional property for the action and interaction shapes that defines time and frequency of occurrence. While attempting to reproduce the Bay of Biscay simulation, there was a glaring need to better define the timing aspect of the actions within CM4S Diagram Prototype for simulation. For example, the notion of travel is easy to conceptually comprehend, but abstracting this continuous action into a discrete action was a critical part of building the simulation.

2. An additional shape or property that completely defines the collection of a key
statistics used to evaluate the operational effectiveness of the simulation. As discussed earlier, not defining how statistics from the simulation are defined can be just as troublesome as not defining behaviors of the simulation.

3. A naming or numbering convention for the shapes was needed to better show the hierarchy of the shapes, relationships between them, and to allow for easier identification of unique shapes. A naming or numbering convention such as the one in IDEF0 was needed.

4. The representation of the interactions and coordination between agents needs improvement and better definition. At the prototype state, a dotted line represents the passing of information between the Aircraft block and the Uboat block (see Figure 49). The coordination of the Aircraft search areas was also represented by a dotted block. Whether this is the best way approach is debatable, however parallel actions on the Agent level are not well represented by the CM4S Diagram Prototype.

9.5 Sanctioning the Reproduced Bay of Biscay ABM Simulation

Using the CM4S Diagram Prototype as a guide and verification tool, the reproduced Bay of Biscay ABM Simulation was constructed using AnyLogic simulation software. Figure 57 shows a screen shot of the simulation in action (the dark ellipses are submerged Uboats). From this screen shot, it can be seen that Uboats are crossing the Bay of Biscay and Aircraft are searching 50 NM by 50 NM areas denoted by the squares in the bay. Furthermore, because it is night (indicated by the dark circle at the bottom of the figure) most of the Uboats are surfaced while some are submerged because they have spotted a plane.

The CM4S Diagram Prototype was then used to verify and sanction the sim-
Figure 57: Bay of Biscay ABM Simulation Screen Shot
ulation. The CM4S Diagram Prototype helped determine that the simulation was running as intended (verification). The simulation was then sanctioned against the results of Champagne’s Dissertation based on both conceptually and operationally sanctioning. To conceptually sanction the simulation, the CM4S Diagram Prototype documented and described how the conceptual model from Champagne’s Dissertation was abstracted. For behavior in the CM4S Diagram Prototype of the Bay of Biscay, and therefore executed by the simulation, a source that provides the “why” justification for each of those behaviors was provided. The few behaviors where Champagne’s description is unclear are documented in the Source property. Therefore, based on the documentation within the CM4S Diagram Prototype, we have a sanctionable conceptual model of the Bay of Biscay simulation that closely mimics the model from Champagne’s 2003 Dissertation.

To operationally sanction the simulation three key statistics were collected and compared to the results published in Champagne’s Dissertation. These statistics were the total number of Uboat kills, the total number of Uboat sightings, and the total number of sortie hours flown at the end of the simulation. A total of 20 replications of the simulation were performed and the results are shown in Figure 58. From this data, several two-sided Two-Sample t-Tests assuming unequal variances were performed at 95% level of significance. The results from these tests are displayed in Figure 59.

The conclusions that can be drawn from the statistical tests are that the reproduced simulation is not significantly different in terms of the number of kills and the number of sightings. However, it is significantly different in terms of the number of sortie hours flown. There are several possible reasons for this discrepancy. The first is that it is unclear exactly how Champagne modeled the Aircraft schedules. In his dissertation he mentions that there are weather delays, however he does not mention the frequency of these delays. Also, there are several different ways one could interpret his description of the Aircraft scheduling procedure and length of flight time. For
Figure 58: Bay of Biscay ABM Simulation Results

<table>
<thead>
<tr>
<th></th>
<th># of Kills</th>
<th># of Sightings</th>
<th># of Sortie Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Champ.</td>
<td>Our</td>
<td>Champ.</td>
</tr>
<tr>
<td>Iteration 1</td>
<td>2</td>
<td>5</td>
<td>138</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>2</td>
<td>3</td>
<td>139</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>3</td>
<td>2</td>
<td>140</td>
</tr>
<tr>
<td>Iteration 4</td>
<td>3</td>
<td>5</td>
<td>150</td>
</tr>
<tr>
<td>Iteration 5</td>
<td>4</td>
<td>4</td>
<td>141</td>
</tr>
<tr>
<td>Iteration 6</td>
<td>2</td>
<td>4</td>
<td>142</td>
</tr>
<tr>
<td>Iteration 7</td>
<td>5</td>
<td>4</td>
<td>143</td>
</tr>
<tr>
<td>Iteration 8</td>
<td>3</td>
<td>2</td>
<td>144</td>
</tr>
<tr>
<td>Iteration 9</td>
<td>6</td>
<td>1</td>
<td>194</td>
</tr>
<tr>
<td>Iteration 10</td>
<td>6</td>
<td>4</td>
<td>158</td>
</tr>
<tr>
<td>Iteration 11</td>
<td>4</td>
<td>1</td>
<td>192</td>
</tr>
<tr>
<td>Iteration 12</td>
<td>3</td>
<td>2</td>
<td>158</td>
</tr>
<tr>
<td>Iteration 13</td>
<td>2</td>
<td>2</td>
<td>187</td>
</tr>
<tr>
<td>Iteration 14</td>
<td>2</td>
<td>4</td>
<td>136</td>
</tr>
<tr>
<td>Iteration 15</td>
<td>4</td>
<td>2</td>
<td>131</td>
</tr>
<tr>
<td>Iteration 16</td>
<td>4</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>Iteration 17</td>
<td>3</td>
<td>4</td>
<td>170</td>
</tr>
<tr>
<td>Iteration 18</td>
<td>5</td>
<td>2</td>
<td>140</td>
</tr>
<tr>
<td>Iteration 19</td>
<td>5</td>
<td>1</td>
<td>158</td>
</tr>
<tr>
<td>Iteration 20</td>
<td>4</td>
<td>4</td>
<td>136</td>
</tr>
<tr>
<td>Mean</td>
<td>3.16</td>
<td>2.95</td>
<td>135.26</td>
</tr>
<tr>
<td>Std Dev.</td>
<td>1.22</td>
<td>1.28</td>
<td>21.42</td>
</tr>
</tbody>
</table>

Figure 59: Two-sided Two-Sample t-Test Results

<table>
<thead>
<tr>
<th></th>
<th>H₀</th>
<th>H₁</th>
<th>α</th>
<th>p-value</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Kills</td>
<td>μ₁ = μ₂</td>
<td>μ₁ ≠ μ₂</td>
<td>0.05</td>
<td>0.074</td>
<td>Fail to reject H₀</td>
</tr>
<tr>
<td># of Sightings</td>
<td>μ₁ = μ₂</td>
<td>μ₁ ≠ μ₂</td>
<td>0.05</td>
<td>0.266</td>
<td>Fail to reject H₀</td>
</tr>
<tr>
<td># of Sortie Hours</td>
<td>μ₁ = μ₂</td>
<td>μ₁ ≠ μ₂</td>
<td>0.05</td>
<td>0.008</td>
<td>Reject H₀</td>
</tr>
</tbody>
</table>

197
example, he says that Aircraft searched the Bay until 70% of their fuel was depleted. Based on a few sentences we interpreted this to mean that the Aircraft were searching for 7 hours until they began to return to the base, but one could interpret this aspect of the Aircraft flight time differently. For example, we could attempt to calculate the actual fuel level based on the flight of the Aircraft.

Another potential reason for this discrepancy is that Champagne does not discuss how he collected the number of sorties hour flown. Different interpretations of when the Aircraft are taking part in searching for Uboats could result in different number of sortie hours flown. This point brings up an unforeseen need that a future version of the CM4S Diagram Prototype needs to incorporate and clearly define how measures of performance are captured.

A third potential reason for this discrepancy is the way in which Champagne arrived at getting his number of sortie hours to match the historical results. The number of Aircraft in his simulation was set to 19 because it resulted in his simulation obtaining flying hour results that were close to the historical figures. This modeling fitting aspect of his simulation could present further difficulties and complexities to the problem of reproducing his simulation because I am not 100% certain how he modeled every single aspect of the airplane and furthermore I cannot be 100% certain that his given descriptions were accurately executed in his simulation. Reproducing a simulation when the original simulation used model fitting may be a more challenging task than reproducing a simulation that is built using model testing.

Although one of the three measures of operational performance was significantly different, it was one of the least critical measures in terms of the objective of the original simulation which was to evaluate strategies of the Aircraft and Uboats. As a result, a reasonable conclusion is that the CM4S Diagram based simulation is both conceptually and operationally sanctioned.

This proof of concept study provides evidence that the CM4S Diagram is very
promising and could aid in the advancement of ABM as an analysis tool.
Utilizing the CM4S Diagram for an ABM Simulation of Order Pickers in a Manual Low Picking, Picker-to-Parts Distribution Center with Congestion

This chapter describes the revised CM4S Diagram and corresponding ABM Simulation of a Distribution Center (DC). In particular, the simulation represents the behaviors of the order pickers in a Picker-to-Part, Low Picking DC and focuses on representing the goals, movements, and interactions of the pickers. The key motivation for simulating this system is personal experience and the lack of literature discussing simulations capable of representing the congestion component of order pickers. The conceptual model of the simulation is described and justified using the CM4S Diagram and the simulation is constructed using the simulation software AnyLogic. To operationally sanction the simulation a series of experiments are performed to test the simulation’s results against the expected dynamics of the system as described in [112]. After operationally sanctioning the simulation, the key results are discussed, the effectiveness of the in CM4S Diagram in representing the conceptual model is evaluated, and suggestions to improve the diagramming technique are made.

To be submitted for publication consideration to the International Journal of Production Research.
10.1 Background and Motivation

The primary reason for studying order pickers in a DC comes from my previous work experience in the domain. While employed as an industrial engineering consultant, I worked on several DC projects where I was predominantly responsible for developing labor standards for powered truck (i.e. forklift, pallet lifts) operators in DCs [48]. Typically, these DCs are classified as Manual Low Level, Picker-to-Parts because items are manually picked from the ground level and the order pickers drive low lift pallet trucks to pick items [31], respectively. During these projects it quickly became evident that the amount of time it takes for an average skilled operator to perform a task (drive to the next item on their order list and pick the item) depended in part on the traffic congestion of their work space. The more order pickers and/or forklift operators in the area, or along their traveled path, the longer it would take them to complete a task.

There are several ways to account for the extra time required to complete a task due to congestion. The most common ways involve observing how much time, on average, an order picker is delayed due to traffic. However, these techniques often become educated guesses because congestion is dynamic and the delay due to congestion is a mixture of many different elements. In short, it is hard to know based on pure observation how much congestion truly impacts the time it takes to complete a task across various dynamic conditions. Thus, there is an opportunity to utilize an analytical tool to gain insights into the impact congestion has on the time it takes to complete a task as well as how the congestion factor changes over time and conditions.

A simulation allows for a variety of experiments in silico to examine the impact that congestion has on ordering picking task completion time. Such experiments would otherwise be extremely costly or impossible to conduct with the real system. For example, it is impossible in the real system for the order pickers to “ignore” each other and proceed as if there was no congestion. However, in a simulation this can
easily be done since the representation of physical space in the simulation is not fixed.

Simulations are often utilized for these reasons and to represent dynamic systems.

One reason for studying the congestion of order pickers in a Manual Low Level, Picker-to-Parts DC comes from academic literature on the subject. While the literature on order picking and DCs is fairly extensive, there are aspects of these systems that lack detailed study. First, the majority of the literature focuses on types of DC systems that compose only 20% of all the observed systems. In other words, the literature is studying some of the least popular systems being utilized today. Conversely, the Manual Low Level, Picker-to-Parts DC system comprise up to 80% of the observed DC systems [31]. Study of these types of systems thus expand upon the literature in this area.

The second area in the academic literature lacking detailed study is the impact congestion has on DC performance, design, and evaluation. The DC or Warehouse design problem has many facets that are highly interconnected, making it a complex problem [100]. As a result, much of the literature is spent identifying and tackling various sub-problems such as determining layout, routing strategy, process flow, allocation of resources, storage and sorting systems, or release strategy for orders [31, 100, 112]. A search of the literature within the general order picking in Manual Low Level, Picker-to-Parts DC domain yielded articles focused on similar sub-problems [38, 53, 31, 89, 88, 99, 112], but often the authors either assume congestion is not a consideration or neglect it in their mathematical or simulation model. In several studies where congestion is considered [45, 84, 85, 86], the mathematical models or simulation models utilized are typically restricted by assumptions such as a limited routing strategy, aisle restrictions, picking strategy, DC type, and/or movement parameter values. One of the main reasons for the limiting assumptions of these analyses is that the types of models being used are not conducive to naturally representing congestion. For example, representing congestion using mathematical
models (e.g. math programming models) before general congestion factors are understood is challenging since mathematical models typically take a very high-level view of the problem. Similarly, process-flow simulation models have trouble representing the micro-level details of picker interactions as the model incorporates these and relaxes assumptions. Thus, another modeling technique is needed to represent and more fully explore the congestion problem.

A more natural way of representing the congestion and movement aspects of the order pickers in this system is the ABM paradigm. The ABM paradigm is defined as representing abstractions of distributed autonomous entities (agents) that are capable of interacting with others and their environment. For the DC system it is easy to envision order pickers as agents since they are distributed, autonomous, and interact with each other (congestion) and their environment (items, aisles, etc.). With the ABM paradigm, a simulation can be constructed that focuses on appropriately representing the observed micro-level behavior of the order pickers in an attempt to obtain the overall macro-level behavior (congestion) of the system. Furthermore, there are examples of utilizing the ABM paradigm in similar systems involving congestion in the road traffic domain [16, 87, 111]. Thus, an ABM Simulation of this system should provide more general insights into congestion than attainable using other modeling paradigms.

10.2 The Simulation Experimental Setup

The objective of this study is to build a generic yet representative ABM Simulation of order pickers in a Manual Low Picking, Picker-to-Parts DC to gain general insight into congestion and to demonstrate the effectiveness of the ABM paradigm for simulating these kind of systems. The conceptual model of this simulation is constructed and sanctioned using the revised CM4S Diagram. The simulation is then constructed based upon the CM4S Diagram and implemented within AnyLogic. Upon verifying
that the simulation is working as intended, experiments are performed to first operationally sanction the simulation and then to gain insight into congestion and the role it plays in order picking.

The experiments performed are based on several principles of order picking that involve congestion as discussed in the widely accepted and referenced textbook *Facilities Planning* by Tompkins et al. [112]. The first of these principles is that spreading out picking locations across the DC will reduce congestion. The second is to use a presorted picking document to control the order that the order picker picks items. It is implied that presorting a list accordingly will reduce travel time. The third is to utilize Pareto’s rule to choose storage locations for the items in the DC such that the most popular items are located together to reduce the total travel time of the order pickers. The fourth is to choose storage locations for the items in the DC such that the most popular items are in the most easily accessed locations to reduce the travel time of the order pickers. From these general rules-of-thumb an experiment was created to operationally sanction the simulation by verifying adherence to these intuitive rules. Next, the results from the same experiment can be used to learn the impact congestion has on these systems.

To test these rules-of-thumbs and to gain further insight into congestion, the experiment is designed with four factors. The first factor (SList) describes how the list of items for each order is sorted and also describes the process used to determine the sequence of locations visited by order picker. The order picker always follows the sequence on the list. The levels of SList are as follows:

1. The list of items is not sorted in any systematic way. The order picker may visit the same aisle multiple times. (Random)

2. The list of items is only sorted by aisle such that items within the same aisle are grouped together and sorted such that the item in that aisle that is closest to the order picker is picked first. The order picker can visit each aisle once at
most, however the sequence that the aisles are visited is not sorted. (Aisle)

3. The list of items is sorted by aisle as described in 2, but the sequence of the
aisles visited is sorted from left to right (based on physical location in the DC)
and aisles with no items to pick are skipped. (Complete)

The second factor (SLoc) describes how the location of each item in the DC is deter-
mined. The levels of SLoc are as follows:

1. The locations of the items are not sorted and items are randomly distributed
across the entire DC. (Random)

2. The locations of the items are sorted such that the most popular items are
placed in the center most aisles. The top 80% of the most popular items are
randomly distributed in aisle 3 and 4 and the remaining items are randomly
distributed between aisle 1, 2, 5, and 6. (Center)

3. The locations of the items are sorted such that the most popular items are
placed in locations that are closest to the shipping area in the front of the DC.
The top 80% of the most popular items are randomly distributed across all of
the aisles and placed in the 3 to 4 locations closest to the shipping area. The
remaining items are randomly distributed in the remaining locations. (Front)

The third factor (TAvoid) describes whether the order pickers attempts to avoid
collisions with any order pickers in their path. When the level of TAvoid is set to 0
the order pickers do not attempt to avoid collisions. Thus, congestion delay will not
occur. When the level of TAvoid is set to 1 the order pickers will attempt to avoid
collision and congestion delay will occur. The final factor (NPickers) describes the
number of pickers in the simulation. There are six levels of NPickers that correspond
to 2, 4, 6, 8, 10, or 12 pickers in the simulation. There are 108 factor combinations
for this full-factorial experiment.
For each factor combination the simulation is replicated 30 times for a total of 3240 runs for the entire experiment. Each run ends when 100 orders of 10 items are picked. Selecting this stopping condition helps determine the impact the number of pickers, and the resulting congestion, have on the time it takes to pick same set number of orders and items.

The raw data collected from each run are as follows:

1. total labor time to pick all of the orders;
2. total time spent traveling;
3. total time spent picking;
4. total time spent setting up between orders;
5. total time stopped due to congestion delays; and
6. total number of times the order picker stopped to walk to the pick location.

To capture the impact of congestion, runs with the same NPickers, SLoc, and SList settings and common random numbers are directly compared when TAvoid is on and off. Thus, all aspects of congestion are directly measured.

10.3 The Conceptual Model

This section discusses the conceptual model of the simulation and presents sanctioning criteria in the form of the revised CM4S Diagram. First, the major revisions to the CM4S Diagram based on the Bay of Biscay Scenario proof of concept are discussed. Next, a high level written description of the conceptual model is presented with a full representation of the CM4S Diagram located in Appendix B. Finally, basic sanctioning criteria of the conceptual model are discussed.
10.3.1 Revisions to the CM4S Diagram

Key revisions were made to the CM4S diagramming technique based on the proof-of-concept effort on the Bay of Biscay Simulation. The changes and additions made to the CM4S Diagram are as follows:

1. The interaction shape (circles) are removed and interactions are simply represented using the action shape; interactions are just special cases of actions. Eliminating the interaction shape simplifies the diagramming technique. The complexity of interactions between blocks is captured and can be appropriately defined using standard Statechart semantics.

2. Decisions shapes (diamonds) are no longer located within the block shape from which they are active. Decisions shapes are now located outside, but still touching, the block from which they are active and arcs are extended from each decision shape to the block shape to which it transitions when triggered. This keeps with the “standard” practices in diagramming techniques and visually represents the transitioning between blocks with the appropriate decision shape.

3. Information shapes are now called variable shapes and are represented as circles. Their primary purpose is to highlight key variables being updated and referenced throughout the simulation.

4. A recorder shape (pentagon) was added to represent how and when data from the simulation is collected.

5. A naming convention was added to all of the shapes making it easier to distinguish between hierarchical levels and to identify what shapes are associated with each other. Block shapes are identified with numbers and all other shapes are identified with letters. The naming convention treats numbers and letters of each shape in a similar manner as in an outline. The top level block is numbered
1 and blocks within this block are numbered 1.1, 1.2, 1.3, etc. This numbering continues for each block within a block. Each action, variable, decision, and recorder shape that belongs to a block has the block’s number along with a distinct letter in alphabetical order. For example, action shapes within the block numbered 1.1 would be numbered 1.1.A, 1.1.B, 1.1.C, etc. Also, each shape category has their own series of letters. For example, a block with a single action, variable, decision, and recorder shape could all be numbered 1.1.A. To abbreviate the length of the each shape’s name only the last number or letter of each shape shown on a page of a diagram is needed.

6. Many properties were added to each shape to provide more flexibility and completeness in how the simulation is executed and to provide more documentation to each shape. The new properties for each shape include:

(a) New Block Shape Properties: Recorders (list of the recorder shapes that belong to the block shape)

(b) New Action Shape Properties: Pseudo Code - Function (the pseudo code of the function), Pseudo Code - Update (how frequently the function is updated or executed), Pseudo Code - Start (when the function begins execution), Pseudo Code - Stop (when the function stops executing), Variables (a list of variables being utilized by the action shape), and Sequence Within Block (a number associated with an action that describes the order that it is executed within the block, one purpose of this is to define how ties are broken).

(c) New Decision Shape Properties: Pseudo Code - Condition (the pseudo code of the condition evaluated), Pseudo Code - Update (how frequently the condition is evaluated), Pseudo Code - Start (when the condition is evaluated), Pseudo Code - Stop (when the condition stops being evalu-
ated), Variables (a list of variables being utilized by the decision shape), and Sequence Within Block (a number associated with a decision that describes the order that it is executed within the block; one purpose of this is to define how ties are broken).

(d) Recorder Shape Properties: Member Of (the block the recorder shape belongs to), Behavior (written description describing the data being collected), Pseudo Code - Function (the pseudo code of the function to be executing), Pseudo Code - Update (how frequently the function is to be updated or executed), Pseudo Code - Start (when the function begins execution), Pseudo Code - Stop (when the function stops execution), Variables (a list of variables utilized by the recorder shape), Sequence Within Block (a number associated with a recorder that describes the order that it is executed within the block; one purpose of this is to define how ties are broken), and Purpose (description of the purpose of the data being collected, not to be confused with Source, which is justification of that behavior).

These revisions help CM4S Diagram more effectively capture, describe, and aid in sanctioning the conceptual model of a simulation.

10.3.2 Description of the Conceptual Model

Next a high level description of the conceptual model of the simulation is presented to highlight key features of the simulation. A more complete description of the conceptual model using the CM4S Diagram is provided in Appendix B. The CM4S Diagram presents all of the variables, behaviors, decisions, justifications, and other elements of the simulation.

The goal of this simulation to capture the congestion factors of a generic, yet realistic, DC. The conceptual model focuses on appropriately capturing the key behaviors, characteristics, and activities performed by the order pickers. The order
Pickers are viewed as autonomous entities (agents) that are given an order with list of items to pick and move through the DC to pick these items. After picking all of the items on the list the order pickers drop the items off at their assigned shipping location and then receive another order of items to pick. It is assumed that the order pickers know the location of each item on their list and can navigate the DC to get from one location to the next. Also, each order picker has the capability to detect and avoid running into other order pickers based on the rules of their current location while traveling between locations.

Picker movement and collision avoidance is not complex. First, the order pickers determine the next item’s location with respect to their current location. Next, they set an intermediate goal location to get them closer to their next item’s location and they begin to move towards that goal location. While traveling they scan forward to check for other order pickers. If an order picker is in their path, then, based on a series of conditions, the order picker will either slow down, stop and wait, stop and walk to pick a close item, or attempt to switch lanes to pass the order picker in their path. Traffic deadlocks are broken by a pre-determined wait time and then the order pickers in deadlock are ordered to move to their intermediate goal location without regard to the congestion around them. As a result, they will travel over each other.

The order pickers move at various speeds depending upon how long they have been moving, the distance to their goal location, and traffic around them. If the order picker is just beginning to move they will move at a slower speed to mimic acceleration. Once they have traveled the acceleration distance they travel at full speed. If the order pickers are approaching a stop, they move at a slower rate for the deceleration distance. Note that order pickers cannot travel in reverse and cannot turn around in the aisle. In practice, traveling in reverse is difficult and it is difficult to turn around when the aisle width is limited in relation to the length of the picker trucks.
The modeled DC has six wide aisles with a total of 168 picking locations and six general shipping locations. Each aisle is wide enough that order pickers can pass each other; however, there are no directional rules in the aisles. So order pickers can travel either on the left or right side of the aisle. To save time and minimize worker fatigue order pickers attempt to get as close as possible to the item location before picking the item. Outside of the aisle are two “highways” that do have directional rules; order pickers only travel on the right side of the road in their direction of travel. At the end of the aisles are intersection points to allow for the order pickers to see traffic in the highways. A picture of the simulated DC environment is shown in Figure 60.

In the conceptual model each item’s location is fixed. The items within each
order are randomly generated when an order picker needs a new order. However, the probability of including an item in an order is based upon the level of SLoc, which determines the popularity of the item based on its current location. Once the order list is generated the order is sorted based on the level of SList.

The complete CM4S Diagram description of the conceptual model is found in Appendix B.

10.3.3 Sanctionability of the Conceptual Model

The sanctionability of the conceptual model of this simulation is primarily based on expert opinion. All of the behaviors captured in the conceptual model are based on personal observations of similar systems or from general observations discussed in various published sources. The CM4S Diagram of the conceptual model in Appendix B details and justifies each behavior captured in the simulation.

10.4 Construction and Operational Sanctionability of the Simulation

The simulation was constructed using AnyLogic, a Java-based simulation software. To verify that the simulation was executing as intended the simulation was executed over many different conditions to watch for bugs and unintended behaviors. This involved over 3000 different runs of the simulation after which the simulation model was tentatively verified for its intended purposes. A screen shot of the simulation running is shown in Figure 61.

The operational sanctionability of the simulation is determined by checking if generally accepted rules-of-thumb for order picking operations are followed. The remainder of this section discusses each operational sanctionability test and demonstrates that the simulation is operationally sanctionable. For each rule-of-thumb the data from all 3240 runs in the experimental setup are utilized unless otherwise noted.
Figure 61: Screen Shot of the DC Order Picker Simulation
Figure 62: DC Order Picker Simulation - Total Mean Congestion Time vs. Item Distribution

The first order picker rule-of-thumb is that spreading out picking locations across a warehouse should reduce congestion. In the simulation experiment items are distributed in three different ways: popular items are placed in the center aisles (Center), popular items are placed in the front of the DC near the shipping locations (Front), and the items are randomly distributed (Random). If this rule-of-thumb holds, then the Random setting will have the lowest mean congestion time. In Figure 62 it is clear that, in general, distributing the items across the DC reduces the total mean congestion time. This is especially true when more pickers are in the system. Therefore, this rule-of-thumb holds in the simulation.

The second order picker rule-of-thumb is that presorting an order appropriately reduces the travel time of the order pickers. In the simulation experiment the order list is sorted in three different ways: only items within the same aisle are sorted appropriately (Aisle), the entire list is sorted such that aisle one is visited first and aisle six is visited last (Complete), and the items are not sorted at all (Random). If this rule-of-thumb holds, then the Random setting will have the highest total mean

214
travel time. In Figure 63 it is clear that not sorting the items in the order significantly increases the total mean travel time to pick all of the orders. Therefore, this rule-of-thumb holds in the simulation.

The third order picker rule-of-thumb is placing the most popular items together in a few aisles will reduce the travel time of the order pickers. Once again, items are distributed in three different ways: popular items are placed in the center aisles (Center), popular items are placed in the front of the DC near the shipping locations (Front), and the items are randomly distributed (Random). If this rule-of-thumb holds, the Center setting will have the lower total mean travel time when compared to the Random setting. In Figure 64 it is clear that placing the most popular items in a few aisles has a smaller total mean travel time than randomly distributing the items when traffic avoidance is turned off. However, this rule-of-thumb does not hold when traffic avoidance is turned on, as shown in Figure 65. As more order pickers are inserted into the system, causing more congestion, the advantage of the Center setting is lost because more pickers are being forced to navigate a highly congested
Figure 64: DC Order Picker Simulation - Total Mean Travel Time vs. Item Distribution, Traffic Avoidance Off

Figure 65: DC Order Picker Simulation - Total Mean Travel Time vs. Item Distribution, Traffic Avoidance On

Panel variable: Num of Pickers
area. While this result is counter to the rule-of-thumb, it intuitively makes sense and only highlights why congestion should be further studied.

The final order picker rule-of-thumb is that placing the most popular items in the front of the DC near the shipping location will reduce the travel time. As in rule-of-thumb three, items are distributed in three different ways: popular items are placed in the center aisles (Center), popular items are placed in the front of the DC near the shipping locations (Front), and the items are randomly distributed (Random). If this rule-of-thumb holds, the Front setting will have the lower total mean travel time when compared to the Random setting. However, upon examining Figure 64 and 65 it is clear that this is not the case. The key reason for this is that the order pickers in the simulation are not allowed to turn around in an aisle, or move in reverse. The majority of the academic literature and textbooks assume that order pickers can turn around in the aisle. Thus, distributing the items in the front in the simulation forces each order picker to traverse the entire DC. Our result is counter to this rule-of-thumb, but based on personal observations of such a DC it is unrealistic for order pickers to turn around in an aisle. Therefore, this result again highlights some deficiencies of these general rules-of-thumb and the need for new study methods.

Based on these four rules-of-thumb evaluations, we generally conclude that the DC Order Picker ABM Simulation is operationally sanctionable under the given conditions and purposes. These evaluations also highlight some of the deficiencies in understanding the operations of order picking and the influence of congestion. Some of the key results of the operationally sanctioned simulation are reviewed in the next section.

10.5 Key Results and Discussion

A primary goal of this simulation is to determine the impact of congestion on an order picking system. From the series of simulation experiments the components of
congestion are not only identified but also quantified. The first main component of congestion is Blocking. Blocking is defined as the time the order picker is at a complete stop due to traffic. There are subcategories of Blocking that are captured in the simulation. These include Blocking in the Aisle, Blocking at an Intersection, and Blocking in the Highway. Previous to this experiment the majority of academic papers on Blocking focused on only estimating Blocking in the Aisle [45, 85, 86]. In future experiments these Blocking subcategories will be explicitly collected and analyzed.

Another major component of congestion is Extra Walking. Extra Walking is the extra time spent walking to pick an item because the location of the item to be picked is blocked. This congestion component is directly related to the Pick-to-Walk distance, which is the distance the order picker is willing to walk to pick an item. Understanding this component of congestion, and its influence on the total congestion time, is an important avenue of research. For example, setting the Pick-to-Walk distance to zero means the order picker can only pick an item that they are directly next to, which should drastically increase congestion time. However, setting the Pick-to-Walk distance to a large distance may increase the total labor time to pick all of the orders because walking to an item is slower than driving. Certainly, there exists some trade-off relationship between congestion time and the Pick-to-Walk time. This aspect of a DC does not seem to have been studied in any detail.

The final major component of congestion is Travel Related. Travel Related is the extra travel time associated with attempting to avoid traffic and collisions. Two main subcategories of Travel Related congestion are the extra distance to drive around traffic and the slower rate of travel due to traffic. It is clear that order pickers travel longer distances to avoid traffic, however the impact of slower rate of travel due to traffic is often not considered. In the simulation when an order picker encounters traffic in their path they must slow down to avoid collisions. Once the traffic has
cleared, they must re-accelerate from that slow speed. The constant stopping and starting means they cannot travel at full speed and the more traffic they encounter the longer it will take for them to complete their order. Often in the academic literature travel speed is considered constant, which is unrealistic. Acceleration and deceleration of order pickers has a significant impact on their performance.

In addition to identifying these components of congestion, the simulation can quantify the components. Figure 66 shows the three key components of congestion and quantifies their percent contribution to the mean congestion time. This pie chart indicates that Blocking is the largest contributor of congestion time at 82.7% with Extra Walking at 15.9% and Travel Related at 1.4%. Although Travel Related is only a small component of congestion in these experiments, I believe that in a larger DC this would comprise a larger percentage of congestion because the current modeled DC is not large enough to allow for order pickers to regularly reach full speed. In
Figure 67: Impact of Congestion on DC Performance

![Cost/Order vs. Orders/Hour vs. Congestion Considerations with 95% Confidence Intervals around the Means](image)

The results from the simulation experiment can also be used to learn how congestion impacts a DC’s operational performance. Figure 67 compares the performance of the simulated DC when traffic avoidance is on and off. On the x-axis is the number of order-per-hour achieved in the simulation, on the y-axis is the labor cost per order, the number next to each data point is the number of order pickers in the simulation at the marked point, and the star data points represent the upper and lower 95% confidence intervals for each mean data point on both the x- and y-axis. From this graph it is clear that there are diminishing returns as more order pickers are inserted into the simulation due to congestion for both the number of orders per hour and the

future experiments the size of the DC will be increased to more accurately represent this size component. Fundamentally, I am unaware of any simulation or mathematical model previous to this one that is capable of representing and quantifying all of these components of congestion.
labor cost per order. Thus, this simulation highlights that not considering congestion in DC analysis can result in unachievable performance and cost expectations. In analyses of actual DCs, graphs such as this could be used by managers and supervisors to make operational decisions, such as assessing the real value of adding another picker.

This kind of DC simulation can also determine the impact that various operating strategies have on the DC, and determine which ones are the best. A radar chart in Figure 68 shows the impact that various strategies have on the mean total labor time to pick the orders. In this chart each axis represents the number of order pickers in the simulation, traveling along each axis represents the mean total labor time to pick all of the orders, and each series represents a different combination of order sorting and item location distribution levels. This chart provides useful insights. For instance, sorting the order list significantly reduces the total labor time. The best set of strategies to reduce total labor time across all picker levels is to combine either the Aisle or Complete sorting strategy with the Center distribution policy. Note that statistical tests also indicate that these two cases are the best and their difference is statistically insignificant across all picker levels, at an alpha of 0.05. Further, the total mean labor time increases as the number of pickers increase for any particular level. These results once again confirm the impact congestion has on system performance.

Overall, this simulation study demonstrates the ability of the ABM paradigm to aid in the understanding and quantifying of congestion in a DC Order Picker system. This study has also shown the ability to determine the impact of congestion on the operational performance of these systems as well as the ability to quantify the diminishing rate of return that congestion causes in these systems. Finally, simulation allows one to analyze the more holistic impact that various DC operational strategies have on the DC’s performance.
Figure 68: Radar Chart of Mean Total Labor Time Under Various Conditions

Mean Total Labor Time to Pick All Orders (sec)

Number of Order Pickers

<table>
<thead>
<tr>
<th>Sort</th>
<th>Dist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>Random</td>
</tr>
<tr>
<td>Random</td>
<td>Center</td>
</tr>
<tr>
<td>Random</td>
<td>Front</td>
</tr>
<tr>
<td>Aisle</td>
<td>Random</td>
</tr>
<tr>
<td>Aisle</td>
<td>Center</td>
</tr>
<tr>
<td>Aisle</td>
<td>Front</td>
</tr>
<tr>
<td>Complete</td>
<td>Random</td>
</tr>
<tr>
<td>Complete</td>
<td>Center</td>
</tr>
<tr>
<td>Complete</td>
<td>Front</td>
</tr>
</tbody>
</table>
10.6 The Effectiveness of the Revised CM4S Diagram

After constructing, running, and analyzing the results of this simulation, the overall effectiveness of the revised CM4S Diagram in representing the conceptual model and appropriate validation emphasis is evaluated. From a technical standpoint it is clear that the technical revisions to the CM4S Diagram were effective in documenting and constructing the conceptual model. The added properties and revised shapes made it much easier to document and construct dynamic and complex behaviors for a simulation. Throughout the use of the revised CM4S Diagram no behavior or activity was encountered that could not be represented or collected with the available shapes and properties. Thus, no major technical changes to the CM4S Diagram are needed at this point.

Another important area to evaluate the diagramming technique’s effectiveness is its ability to concisely and completely document the conceptual model. This DC Order Picker simulation is much larger and has many more complicated behaviors and activities to capture, computerize, and document than in the proof of concept effort. However, just as with the Bay of Biscay Scenario Simulation, the CM4S Diagram performed well. The combination of visual formalisms with the data base allows for someone attempting to better understand the conceptual model to gain both deep and shallow knowledge concerning the execution of the conceptual model and the simulation. Attempting to fully document the conceptual model of this simulation to the extent of the CM4S Diagram in only 25 pages would be extremely difficult; the CM4S Diagram is concise yet complete.

The final area to evaluate the CM4S Diagram’s effectiveness is its ability to aid in sanctioning the conceptual model. While the Source property certainly provides key justifications for each action, decision, and variable that aided in the sanctioning of this conceptual model, the revised version of the CM4S Diagram does miss the
ability to convey justification or reasoning for abstracting a system a certain way. For example, in this simulation there is no way in the revised CM4S Diagram to justify or convey why I chose to represent order pickers in the way that I did. To rectify this problem the Source property was added to the Block shape. Adding this property allows for a more complete documentation of the justifications for representing a system a certain way. This also allows for the motivation of the conceptual model and/or the simulation itself to be documented in the diagram.

Utilizing the revised CM4S Diagram to construct this simulation demonstrated that its technical, sanctioning, and documenting capabilities align with its intended purpose and design criteria. Thus, adding the aforementioned Source property to the Block shape represents the final major change to the first publicly released version of the CM4S Diagram. Note that the CM4S Diagram is an evolving diagramming technique that will change and improve over time and application. There will likely be new versions of the CM4S Diagram to be released in the future.

10.7 Conclusions

This chapter demonstrates the effectiveness of the CM4S Diagram using the construction of a sanctionable ABM Simulation of order pickers in a DC. As a result of the CM4S Diagram’s evaluated capabilities only minor changes are needed before the first version of the diagramming technique is publicly released. This chapter also demonstrated the capabilities of the ABM paradigm to represent DC and warehousing systems, which has not been done before. This simulation effectively captures and quantifies the impact of congestion on a DC’s operational performance and can be utilized to both improve the research in and practice of DC and warehousing management strategies at a level that has not been previously achieved. Fundamentally, this simulation demonstrates how the ABM paradigm with proper tools, sanctioning practices, and system abstraction can help explore and analyze difficult to understand
systems from both a research and practice perspective.
Contributions and Future Research Opportunities

This dissertation advances ABM as a generic analysis tool such that ABM can reach its full potential as a revolution in modeling and simulation. To achieve this goal, the field of ABM was examined from many perspectives to provide contributions to each perspective. The first three perspectives of ABM examined were complex systems, the historical emergence of ABM, and philosophical issues related to ABM. Investigating these ABM topics established clear foundations for the field across multiple disciplines. Next, the current practice of ABM was investigated. Through a comprehensive 279 article survey current deficiencies and opportunities in ABM were identified. Based on these deficiencies, a new diagramming technique called the CM4S Diagram was developed. The CM4S Diagram represents the first diagramming technique designed specifically for the effective representation, construction, and sanctioning of ABM computer simulations based on identified needs in the ABM modeling field and simulation modeling philosophy. Finally, the effectiveness of the CM4S Diagram is evaluated through the development of social science, military, and supply chain ABM simulations.

11.1 Contributions

The contributions of the research towards advancing ABM as a generic analysis tool are as summarized:

(a) Clarified the meaning and differences between real systems and model systems.

(b) Described the independent components that are used to measure complexity (Size and Unexplored) and related them to a problem solving process using model systems.

(c) Extended Weaver’s problems framework into model systems by further defining the differences, similarities, and relationships between Primitive Model Systems, Simple Model Systems, Disorganized Complex Model Systems, and Organized Complex Model Systems.

(d) Reconciled the various definitions of complex systems by incorporating Weaver’s framework and breaking complex systems into two sub categories depending upon the observed properties of the abstracted complex system problem.


(a) Explored how the development of computers, cybernetics, complexity, cellular automata, and chaos as well as the quest to understand natural systems led to the emergence of ABM today.

(b) Connected fundamental ABM behaviors and properties to key theories to provide ABM developers with a clearer understanding of the field and its scientific roots.

3. Simulation and Agent-Based Modeling Validation Philosophy. Packaged with the History Chapter and published as Chapter 3 in the Handbook of Research

(a) Conducted an extensive 279 article survey to establish the practices in ABM from 1998 to 2008 that to my knowledge is the only such survey of its kind for ABM. In particular, data was collected and reported for each article including the year, author(s), journal, field of study, employed software, validation techniques and standards, complete description, and purpose.

(b) Derived six fundamental needs for ABM based on current practice deficiencies and opportunities. Including:
i. The need for development and documentation tools for ABM that are independent of software such that proper simulation programming techniques are being utilized;

ii. The need for ABM study as an independent discipline that is a subset of the simulation discipline such that standard techniques, practices, philosophies, and methodologies can be established extending ABM as a functional analysis tool;

iii. The need for different expectations for ABM based upon the level of understanding concerning the system that is being simulated;

iv. The need for complete references to a model in all articles in the form of the actual simulation or some other descriptive tool that can be used to independently develop and evaluate the effectiveness of the model;

v. The need for reviewers and publication outlets to require that all models be completely sanctioned and documented; and

vi. The need for statistical and non-statistical sanctioning techniques to be specifically developed for ABM and effectively conveyed to those building agent-based models.


(a) Clarified the differences between diagramming techniques, models, and simulations.

(b) Created three distinct classifications of diagramming techniques based on their objectives and capabilities: Organizational Diagramming Techniques, Process Flow Behavioral Diagramming Techniques, and Machine Behavioral Diagramming Techniques.

(a) Extended my simulation framework to relate the level of understanding about the real system to the appropriate sanctioning emphasis and discussed the implications of this addition in terms of how the understanding of the system dictates whether conceptual or operational sanctioning should be the focus of sanctioning activities.

(b) Defined why fitting a model to the real world results is not an appropriate sanctioning technique for simulation and ABM.

(c) Identified the inability of existing diagramming techniques for satisfying the identified requirements.

(d) Designed, tested, and implemented the Conceptual Model for Simulation Diagram, which is the first diagramming technique designed specifically for the effective representation, construction, and sanctioning of ABM computer simulations based on identified needs in the ABM modeling field and simulation modeling philosophy.


7. Supply Chain Distribution Center Operations.

(a) Implemented and analyzed an ABM simulation of order picker activities to capture the effects of congestion on the operational performance of the Distribution Center.

(b) To be submitted for publication consideration to the International Journal of Production Research.
11.2 Future Research Opportunities

The breath of this research presents many future research opportunities. Some of them are summarized below:

1. Explore and refine the framework of model systems and complexity to better understand how model systems are utilized in solving problems.

2. Further integrate the history and philosophy of ABM with the model systems complexity framework.

3. Investigate opportunities to develop quantitative measures or tests specifically for the unique needs of ABM paradigm.

4. Explore the use and usefulness of the CM4S Diagram for representing conceptual models from other simulation paradigms as well as other modeling paradigms such as mathematical programs.

5. Explore the usability of the CM4S Diagram in terms of application, learning, and improved coding practice.

6. Investigate creating an a tool that automatically creates code based on the CM4S Diagram syntax and semantics.

7. Investigate creating a tool that guides and forces modelers to build the CM4S Diagram in a correct and standard way.

8. Explore the utilization of history tracking methods to manage versions of the CM4S Diagram over the history of a simulation.

9. Examine the use of the CM4S Diagram as a simulation educational tool.

10. Use the ABM Distribution Center simulation to further examine the impact of congestion and similar measures on various scenarios of warehousing operations.
Bibliography


Appendix A: Surveyed Articles


Appendix B: CM4S Diagram of the DC Order Picker Simulation
**Figure 2: CM4S Diagram Page 1 - Action Shapes**

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Function</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variables</th>
<th>Sequence within Block</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>A Create Coor</td>
<td>2 Build, Environment</td>
<td>Create x &amp; y coordinates of each 168 item locations, 6 shipping location, and starting location. 6 wide-aisles with no traffic direct restrictions, 2 highways. Entrance and Exit points to the aisles coordinates where highways and aisles intersect, X coordinate range 0-2000, Y coordinate range 0-2050. Highways require the pickers to drive on the right side of the road.</td>
<td>Create all environment coordinates (K,L,O,P,Q)</td>
<td>On Entry</td>
<td>K,L,O,P,Q</td>
<td>1</td>
<td>See picture of the environment. Based on assumptions of size of pallets (40”x48”) and based on DC discussed in Koster, Le-Duc, &amp; Roodbergen 2007. Order pickers only pick from the ground level</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>A Collect Var</td>
<td>1 Experimental Setup</td>
<td>Collect all of the variables (A-J) of the experimental setup</td>
<td>Store user entered values of the variables of the experiment (A-J)</td>
<td>On Entry</td>
<td>A-J</td>
<td>1</td>
<td>See picture of the environment. Based on assumptions of size of pallets (40”x48”) and based on DC discussed in Koster, Le-Duc, &amp; Roodbergen 2007.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>B Assign Lanes</td>
<td>2 Build, Environment</td>
<td>Assign the lanes and aisles of each item</td>
<td>Based on x &amp; y coordinates, assign each item a lane and aisle (M,N)</td>
<td>On Entry</td>
<td>M, N</td>
<td>2</td>
<td>Based on the environment.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>A Create New P</td>
<td>2 Inserting Pickers</td>
<td>Create a new picker at the entry point location</td>
<td>Create new picker with all appropriate parameter values. Place at entry point</td>
<td>On entry</td>
<td>A,J, O</td>
<td>1</td>
<td>N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>A End the Sim</td>
<td>4 Stop Sim</td>
<td>Stop the simulation</td>
<td>End</td>
<td>on entry</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>From</td>
<td>To</td>
<td>Actions</td>
<td>Decisions</td>
<td>Recorders</td>
<td>Variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>------</td>
<td>----</td>
<td>---------</td>
<td>-----------</td>
<td>-----------</td>
<td>-----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Experimental Setup 1. DC Order Picker Simulation</td>
<td>1. DC Order Picker Simulation</td>
<td>2. Build_Environment</td>
<td>A. Collect Variable Values</td>
<td>A. Proceed?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Build_Environment</td>
<td>1. Experimental Setup</td>
<td>3. Execution</td>
<td>A. Create Coordinates, B. Assign Lanes &amp; Aisle to Items</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Environment</td>
<td>2. Inserting Pickers</td>
<td>A. Insert Pickers?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Inserting Pickers</td>
<td>1. Environment</td>
<td>1. Environment</td>
<td>A. Create New Picker</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Stop Sim</td>
<td>3. Execution</td>
<td>A. End the Sim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Decision Shapes

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Transitions To</th>
<th>Behavior</th>
<th>Pseudo Code - Condition</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Sequence within Block</th>
<th>Variable</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>A Proceed?</td>
<td>2 Environment</td>
<td>When the experimental conditions are set, transition to the environment of the simulation.</td>
<td>If user is ready to proceed, then proceed.</td>
<td>1 second</td>
<td>On Entry</td>
<td>On Exit</td>
<td>1</td>
<td></td>
<td>Assumption</td>
</tr>
<tr>
<td>Decision</td>
<td>A Insert Pickers?</td>
<td>2. Inserting Pickers</td>
<td>If it is time to insert more pickers based on the release delay and not all of the pickers have been inserted, then insert the next picker.</td>
<td>If ( \text{time_since_last_insert} \geq \text{DelayRelease} ) and ( \text{Current_num_pickers} &lt; \text{NumPickers} ) then transition.</td>
<td>Every second</td>
<td>on entry</td>
<td>on exist</td>
<td>1</td>
<td>DelayRelease, NumPickers</td>
<td>N/A</td>
</tr>
<tr>
<td>Decision</td>
<td>A All Orders Picked?</td>
<td>4. Stop Tim</td>
<td>If the current number of orders collected is equal to the num of orders to be collected, then stop the simulation.</td>
<td>If ( \text{cur_Num_Of_Orders} = \text{Num_Orders} ) then transition.</td>
<td>Every second</td>
<td>on entry</td>
<td>on exist</td>
<td>1</td>
<td>J. NumOrders, S. curNumOrders</td>
<td>N/A</td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Pseudo Code - Function</td>
<td>Pseudo Code - Update</td>
<td>Pseudo Code - Start</td>
<td>Pseudo Code - Stop</td>
<td>Variables</td>
<td>Sequence within Block</td>
<td>Purpose</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>----------</td>
<td>-------------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>----------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Record A</td>
<td>Num of Collisions</td>
<td>3. Execution</td>
<td>Check if any agents get within 2' of each other. If there is a collision remember the collision for 5 seconds and do not double count the collisions.</td>
<td>For each picker: If (distance_to(other_picker) &lt;= 2' and Time - previous_col_time[other_picker] &gt;= 5 sec) then add the collision and previous_col_time[other_picker] = Time;</td>
<td>0.1 seconds</td>
<td>on entry</td>
<td>on exit</td>
<td>time</td>
<td>6</td>
<td>Collect the number of collisions that occur in the simulation.</td>
</tr>
<tr>
<td>Record B</td>
<td>TotOrders</td>
<td>3. Execution</td>
<td>Sum up all of the orders collected from all of the pickers</td>
<td>For each order picker: TotOrders = TotOrders + order_picker.NumOrders;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Collect total number of orders.</td>
</tr>
<tr>
<td>Record C</td>
<td>TotItems</td>
<td>3. Execution</td>
<td>Sum up all of the items collected from all of the pickers</td>
<td>For each order picker: TotItems = TotItems + order_picker.NumItems;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Collect total number of items.</td>
</tr>
<tr>
<td>Record D</td>
<td>TotPickTime</td>
<td>3. Execution</td>
<td>Sum up the total pick time</td>
<td>For each order picker: TotPickTime = TotPickTime + order_picker.PickTime;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Record E</td>
<td>TotSetupTime</td>
<td>3. Execution</td>
<td>Sum up the total setup time</td>
<td>For each order picker: TotSetupTime = TotSetupTime + order_picker.SetupTime;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Record F</td>
<td>TotDelayTime</td>
<td>3. Execution</td>
<td>Sum up the total delay time</td>
<td>For each order picker: TotDelayTime = TotDelayTime + order_picker.DelayTime;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Record G</td>
<td>TotWorkTime</td>
<td>3. Execution</td>
<td>Calculate the total work time of all the pickers</td>
<td>For each order picker: TotWorkTime = TotWorkTime + order_picker.TotTime;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Record H</td>
<td>TotTravelTime</td>
<td>3. Execution</td>
<td>Calculate the total travel time</td>
<td>For each order picker: TotTravelTime = TotTravelTime + order_picker.TotTravelTime;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>-----------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable A</td>
<td>NumPickers</td>
<td>1 DC Order Picker Simulation</td>
<td>Number of Pickers in the Simulation. Varies from 2-12.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable B</td>
<td>DelayRelease</td>
<td>1 DC Order Picker Simulation</td>
<td>The time delay between each order picker being released into the simulation at the entry point. Set to 10 seconds.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable C</td>
<td>TravelSpeed</td>
<td>1 DC Order Picker Simulation</td>
<td>The maximum travel speed of the pickers while on the truck. Set to 5mph.</td>
<td>Based on experimental data and personal experience.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable D</td>
<td>PickTime</td>
<td>1 DC Order Picker Simulation</td>
<td>The time it takes to pick one item and place it on the pallet. Only includes the pallet being directly next to the item. No variation of the time based on cube or weight of the item. Set to 14.75 sec based on BasicMOST: A10 B3 G3 A6 B3 P6 A10.</td>
<td>Based on personal experience.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable E</td>
<td>SetupTime</td>
<td>1 DC Order Picker Simulation</td>
<td>The time it takes to setup a pallet after all of the items have been picked. Time does not vary based on items. Set to 60 sec.</td>
<td>Based on personal experience.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable F</td>
<td>AccelBreak</td>
<td>1 DC Order Picker Simulation</td>
<td>The distance it takes to reach full speed. Set to 50 feet.</td>
<td>Based on personal experience and collected data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable K</td>
<td>ItemXCoord</td>
<td>1 DC Order Picker Simulation</td>
<td>An items x coordinate. An Array. 168 Locations.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable H</td>
<td>TrafficOn</td>
<td>1 DC Order Picker Simulation</td>
<td>Describes whether traffic detection is on or off. 1=on, 0=off.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable I</td>
<td>WalkToPickDist</td>
<td>1 DC Order Picker Simulation</td>
<td>The distance at which the picker will walk to the item if they are stopped due to traffic delays. Set to 10 feet.</td>
<td>Based on observation and estimate.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable J</td>
<td>NumOrders</td>
<td>1 DC Order Picker Simulation</td>
<td>The total number of orders to be picked in the simulation. Set to 100.</td>
<td>Based on estimates of 30-60 minutes for each order and an 8 hour day.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable G</td>
<td>DecelBreak</td>
<td>1 DC Order Picker Simulation</td>
<td>The distance it takes to stop when at full speed. Set to 25 feet.</td>
<td>Based on personal experience and collected data.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable L</td>
<td>ItemYCoord</td>
<td>1 DC Order Picker Simulation</td>
<td>An items y coordinate. An Array. 168 Locations.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable M</td>
<td>ItemAisle</td>
<td>1 DC Order Picker Simulation</td>
<td>An items aisle (1-6) location. An Array.</td>
<td>6 aisles based on estimate.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable N</td>
<td>ItemLane</td>
<td>1 DC Order Picker Simulation</td>
<td>An items lane (0-1) location. An array.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable O</td>
<td>EntryLocation</td>
<td>1 DC Order Picker Simulation</td>
<td>Entry Location of the pickers. (75, 1075)</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable P</td>
<td>ShipLoc</td>
<td>1 DC Order Picker Simulation</td>
<td>Shipping Locations (12 possible coordinates). Y=1175, X based on aisle locations. An array.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Q</td>
<td>EndAisleLoc</td>
<td>1 DC Order Picker Simulation</td>
<td>End of aisle locations. 4 for every aisle. Y=225 or Y==975. X based on aisle locations/lanes. An array.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable R</td>
<td>Time</td>
<td>1 DC Order Picker Simulation</td>
<td>The current time of the simulation. 1 unit=1 second.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable S</td>
<td>CurNumOrders</td>
<td>1 DC Order Picker Simulation</td>
<td>The current number of orders completed</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable T</td>
<td>DeadBreakTime</td>
<td>1 DC Order Picker Simulation</td>
<td>The maximum time the picker will wait until the deadlock is broken. Set to 30 seconds.</td>
<td>Based on estimate time to break a deadlock.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>Member Of Behavior</td>
<td>Pseudo Code - Function</td>
<td>Pseudo Code - Update</td>
<td>Pseudo Code - Start</td>
<td>Pseudo Code - Stop</td>
<td>Variables</td>
<td>Sequence within Block</td>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------------</td>
<td>--------------------</td>
<td>------------------------</td>
<td>----------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>----------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Update Time &amp; Location</td>
<td>2. Time Update Mechanism</td>
<td>Advance the clock and update each picker’s location</td>
<td>As needed</td>
<td>on entry</td>
<td>on exist</td>
<td>1. R. Time</td>
<td>1</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Set Next Order 1. Get Next Order</td>
<td>Generate the next list of 5 items in the order. This may be uniform or some other distribution depending upon the objective of the experiment.</td>
<td>For i=1 to 5, i+1. ItemList[i]=Uniform(1,168)</td>
<td>on entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Set Next Item 2. Get Next Item</td>
<td>Set the next target location to be the next item in the list. If the order is done, go back to shipping unless shipping is directly behind you then set up phantom item an aisle down. Check if any other orders in the list are in the same aisle, if so sort</td>
<td>If (order_complete==true) then set target to shipping location or phantom location one aisle over. Else If (other_items_in_aisle==true) then set the next item accordingly. Set target to the next item location.</td>
<td>on entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Set Pick Time Delay 6. Picking Item</td>
<td>If the current item is a phantom item then CurPickTime=0. Otherwise the CurPickTime is the walking distance to the item plus the standard pick time.</td>
<td>If (current_item==phantom) then CurPickTime=0; Else CurPickTime=PickTime+walkTo<em>2</em>walkingDist*(1/2mph);</td>
<td>on entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Update num orders 7. Setting Up completed</td>
<td>Add one to the current number of orders completed</td>
<td>curNumOrders++;</td>
<td>on entry</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 8: CM4S Diagram Page 2 - Action Shapes**

- **Master Name**: Action
- **Displayed Text**: A
- **Member Of**: Update Time & Location
- **Pseudo Code - Function**: Advance the clock and update each picker’s location
- **Pseudo Code - Update**: As needed
- **Pseudo Code - Start**: on entry
- **Pseudo Code - Stop**: on exist
- **Variables**: 1. R. Time
- **Sequence within Block**: 1
- **Source**: N/A

- **Master Name**: Action
- **Displayed Text**: A
- **Member Of**: Set Next Order 1. Get Next Order
- **Pseudo Code - Function**: Generate the next list of 5 items in the order. This may be uniform or some other distribution depending upon the objective of the experiment.
- **Pseudo Code - Update**: For i=1 to 5, i+1. ItemList[i]=Uniform(1,168)
- **Pseudo Code - Start**: on entry
- **Pseudo Code - Stop**: | | | | |
- **Variables**: A. ItemList
- **Sequence within Block**: 1
- **Source**: N/A

- **Master Name**: Action
- **Displayed Text**: A
- **Member Of**: Set Next Item 2. Get Next Item
- **Pseudo Code - Function**: Set the next target location to be the next item in the list. If the order is done, go back to shipping unless shipping is directly behind you then set up phantom item an aisle down. Check if any other orders in the list are in the same aisle, if so sort
- **Pseudo Code - Update**: If (order_complete==true) then set target to shipping location or phantom location one aisle over. Else If (other_items_in_aisle==true) then set the next item accordingly. Set target to the next item location.
- **Pseudo Code - Start**: on entry
- **Pseudo Code - Stop**: | | | | |
- **Variables**: B. CurItemX, C. CurItemY, D. CurItemAisle, E. CurItemLane
- **Sequence within Block**: 1
- **Source**: Order pickers cannot turn around in an aisle to get to a location directly behind them - based on personal observation.

- **Master Name**: Action
- **Displayed Text**: A
- **Member Of**: Set Pick Time Delay 6. Picking Item
- **Pseudo Code - Function**: If the current item is a phantom item then CurPickTime=0. Otherwise the CurPickTime is the walking distance to the item plus the standard pick time.
- **Pseudo Code - Update**: If (current_item==phantom) then CurPickTime=0; Else CurPickTime=PickTime+walkTo*2*walkingDist*(1/2mph);
- **Pseudo Code - Start**: on entry
- **Pseudo Code - Stop**: | | | | |
- **Variables**: 1. D. PickTime, L. CurPickTime, M. walkTo, N. walkingDest
- **Sequence within Block**: 1
- **Source**: 2mph is based on slow walking pace because they are only walking when there is traffic. They must walk there and back.

- **Master Name**: Action
- **Displayed Text**: A
- **Member Of**: Update num orders 7. Setting Up completed
- **Pseudo Code - Function**: Add one to the current number of orders completed
- **Pseudo Code - Update**: curNumOrders++;
- **Pseudo Code - Start**: on entry
- **Pseudo Code - Stop**: | | | | |
- **Variables**: 1. S. curNumOrders
- **Sequence within Block**: 2
- **Source**: N/A
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>From</th>
<th>To</th>
<th>Actions</th>
<th>Decisions</th>
<th>Recorders</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>1.3.1 Environment</td>
<td></td>
<td>2.</td>
<td>Inserting Pickers</td>
<td>A. Insert Pickers?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Time &amp; Location Update Mechanism</td>
<td></td>
<td></td>
<td>A. Update Time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Get Next Order</td>
<td>7.</td>
<td>2.</td>
<td>Setting Up</td>
<td>A. NumOrders, B. TotTime</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Movement</td>
<td>3. Get Next Goal Location</td>
<td></td>
<td></td>
<td>A. At Goal?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>6 Picking Item</td>
<td>5. Check Location</td>
<td>2. Get Next Item</td>
<td>A. Set Pick Time Delay</td>
<td>A. Next Item?</td>
<td>A. PickTime</td>
<td></td>
</tr>
</tbody>
</table>

Figure 9: CM4S Diagram Page 2 - Block Shapes
<table>
<thead>
<tr>
<th>Behavior</th>
<th>Transitions To</th>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Transition Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision A</td>
<td>Next Order? 2. Get Next Item</td>
<td>If the order picker is done picking the item, then get the next item to pick. If (Time - time_since_entry &gt;= CurPickTime) then transition to Get Next Order. On entry 1. R. Time, L. CurPickTime 2 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Insert Pickers?</td>
<td>If it is time to insert more pickers based on the release delay and not all of the pickers have been inserted, then insert the next picker. If (time_since_last_insert &gt;= DelayRelease and Current_num_pickers &lt; NumPickers) then transition. Every second on entry DelayRelease, NumPickers 1 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Next Order?</td>
<td>Checking if there is another order to run. If no order is available, then transition to Get Next Item. On entry 1. R. DelayRelease, NumPickers 4 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Not Target? 3. Get Next Goal Location</td>
<td>If the picker's location is not the target, then find the next goal location to get to the target. If (CurX != CurItemX and CurY != CurItemY) then transition. On entry F. CurX, G. CurY, B. CurItemX, C. CurItemY 1 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>At Item? 6. Picking Item</td>
<td>If the picker's location is at the current item, then pick the item. If (CurX == CurItemX and CurY == CurItemY) then transition to Picking Item. On entry F. CurX, G. CurY, B. CurItemX, C. CurItemY 2 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>At Goal? 3. Get Next Goal Location</td>
<td>If the picker is currently at the goal location, then check the my current location. If (CurX == GoalX and CurY == GoalY) then transition. On entry 1. R. Time, E. SetupTime 1 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Movement If the next goal location has been set and the picker is ready to move, then transition to get general movement. If (readyToMove == 1) then transition.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Setting Up</td>
<td>If the picker's location is their shipping location, then set up the pallet for the next order. If (CurX == ShipX and CurY == ShipY) then transition to Setting Up. On entry F. CurX, G. CurY, B. CurItemX, C. CurItemY 3 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Get Next Order</td>
<td>If the order picker is done with setting up for the next order (waited the SetupTime), then they can get their next order. If (Time - time_since_entry &gt;= SetupTime) then transition to Get Next Order 0.1 seconds On entry 1. R. Time, 1. E. SetupTime 1 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision B</td>
<td>At Item?</td>
<td>If the picker's location is at the current item, then pick the item. If (CurX == CurItemX and CurY == CurItemY) then transition to Picking Item. On entry F. CurX, G. CurY, B. CurItemX, C. CurItemY 2 N/A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision C</td>
<td>Setting Up</td>
<td>If the picker is done picking the item, then get the next item to pick. If (Time - time_since_entry &gt;= CurPickTime) then transition to Get Next Order 0.1 seconds On entry 1. R. Time, L. CurPickTime 2 N/A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 10**: CMS Diagram Page 2 - Decision Shapes
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Pseudo Code - Function</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variables</th>
<th>Sequence within Block</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record A</td>
<td>PickTime</td>
<td>6. Picking Time</td>
<td>Record the amount of time spent picking</td>
<td>PickTime=PickTime+time_exit-time_entered;</td>
<td>on entry</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td>Count the time spent picking by the picker</td>
</tr>
<tr>
<td>Record A</td>
<td>SetupTime</td>
<td>7. Setting Up</td>
<td>Record the amount of time spent setting up</td>
<td>SetupTime=SetupTime+time_exit-time Entered;</td>
<td>on entry</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td>Count the time spent setting up by the picker</td>
</tr>
<tr>
<td>Record A</td>
<td>NumOrders</td>
<td>1. Get Next Order</td>
<td>Record the number of orders</td>
<td>if (NumOrders!=null) then NumOrders++;</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of orders by the picker</td>
</tr>
<tr>
<td>Record A</td>
<td>NumItems</td>
<td>2. Get Next Item</td>
<td>Record the number of items</td>
<td>if (NumItems!=null) then NumItems++;</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Number of items by the picker</td>
</tr>
<tr>
<td>Record B</td>
<td>TotTime</td>
<td>1. Get Next Order</td>
<td>Record the total time the picker has been working</td>
<td>TotTime=TotTime+time_last_left</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Count the total time spent working by the picker</td>
</tr>
<tr>
<td>Record A</td>
<td>DelayTime</td>
<td>1. Order Picker</td>
<td>Record the amount of time spent not moving</td>
<td>DelayTime=DelayTime+resume_move_time-stop_move_time; if (isMoving==false) then Update=0.1;</td>
<td>on entry</td>
<td>on exit</td>
<td></td>
<td></td>
<td></td>
<td>Count the time spent delayed by the picker</td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>Member Of</td>
<td>Behavior</td>
<td>Source</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------</td>
<td>----------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable A</td>
<td>ItemList</td>
<td>1. Order Picker</td>
<td>An array of 10 items in the list of an order.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable B</td>
<td>CurItemX</td>
<td>1. Order Picker</td>
<td>Current target's x coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable C</td>
<td>CurItemY</td>
<td>1. Order Picker</td>
<td>Current target's y coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable D</td>
<td>CurItemAisle</td>
<td>1. Order Picker</td>
<td>Current target's aisle location</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable E</td>
<td>CurItemLane</td>
<td>1. Order Picker</td>
<td>Current target's lane location</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable F</td>
<td>CurX</td>
<td>1. Order Picker</td>
<td>The Order Picker's current x coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable G</td>
<td>CurY</td>
<td>1. Order Picker</td>
<td>The Order Picker's current y coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable H</td>
<td>GoalX</td>
<td>1. Order Picker</td>
<td>The Order Picker's current goal x coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable I</td>
<td>GoalY</td>
<td>1. Order Picker</td>
<td>The Order Picker's current goal y coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable J</td>
<td>ShipX</td>
<td>1. Order Picker</td>
<td>The Order Picker's ship x coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable K</td>
<td>ShipY</td>
<td>1. Order Picker</td>
<td>The Order Picker's ship y coordinate</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable L</td>
<td>CurPickTime</td>
<td>1. Order Picker</td>
<td>The Order Picker's current pick time (it is dependent upon distance to the item and whether that item is a phantom item)</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable M</td>
<td>walkTo</td>
<td>1. Order Picker</td>
<td>Describes whether the picker is walking to the item. 1=walking, 0=not walking</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable N</td>
<td>walkingDist</td>
<td>1. Order Picker</td>
<td>The walking distance of the order picker to get the item</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable O</td>
<td>CurAisle</td>
<td>1. Order Picker</td>
<td>The order picker's current aisle, -1 if not in an aisle</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable P</td>
<td>switchNeeded</td>
<td>1. Order Picker</td>
<td>The order picker's current aisle, -1 if not in an aisle</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable Q</td>
<td>startDeadLock</td>
<td>1. Order Picker</td>
<td>The start time when the picker is forced to stop due to traffic.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable R</td>
<td>iniX</td>
<td>4. Movement</td>
<td>Describes the initial X Coordinate when movement is started after stopping.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable S</td>
<td>iniY</td>
<td>4. Movement</td>
<td>Describes the initial X Coordinate when movement is started after stopping.</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 13: CM4S Diagram Page 3

1.3.1.1.3 Get Next Goal Location

1 Picker Location Check
   A Reset Variables
   B Intersect?
   C Aisle?
   D Ship?

2 In Highway
   A Highway?
   B readyToMove
   C altInter

3 At Intersection
   1 Get Int Goal Locations
     A Set Temp Inter Goal Loc
     B waitDelay
     C TempX
     D TempY

4 In Aisle
   1 Item Location Check
     A Same Aisle?
     B Diff Aisle?

5 At Shipping
   1 Get Ship Goal
     A Entering?
     B Exiting?

2 Going To Set
   A Set Ship Goal Loc

3 Going To S Int
   A Set S Inter Goal Loc

2 Going To Inter
   A Set Inter Goal Loc

3 Going To Item
   A Set Item Goal Loc
A Set Highway Goal Loc. 1. Picker Traffic Check If the picker is entering any lanes, then scan from the entering point to the DecelBreak distance in the pickup lane for any traffic and if any traffic is heading toward the entering point. If so, wait. If (scan(my coordinates, DecelBreak->returns a picker) then waitDelay=1; Else waitDelay=0; 0.1 on entry on exit B. waitDelay, G. DecelBreak 2

3. Going To Item 2. Going To Location Proceed to the closest intersection in your heading direction. GoalX=CurX; If (heading==north) then GoalY=225. Else GoalY=975; readyToMove=1; atInter=0; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Item Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.

A Set Inter Goal Loc. 3. Going To Item Proceed to the closest intersection (current x coordinate & y coordinate=1125). GoalX=CurX; GoalY=1125; readyToMove=1; atInter=1; on entry F. CurX, H. GoalX, I. GoalY, A.
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>From</th>
<th>To</th>
<th>Actions</th>
<th>Decisions</th>
<th>Recorders</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block</td>
<td>1 Picker Location Check</td>
<td>2. In Highway, 3. At Intersection, 4. In Aisle, 5. At Shipping</td>
<td>1. Picker Location Check</td>
<td>A. Reset Variables</td>
<td>A. Highway?, B. Intersection?, C. Aisle?, D. Ship?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 In Highway</td>
<td>1. Picker Location Check</td>
<td>1. Picker Location Check</td>
<td>A. Set Highway Goal Loc</td>
<td>A. waitDelay, B. wavedAt, C. TempX, D. TempY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 At Intersection</td>
<td>1. Picker Location Check</td>
<td>1. Picker Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td>A. Same Aisle?, B. Diff Aisle?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>5 At Shipping</td>
<td>1. Picker Location Check</td>
<td>1. Picker Location Check</td>
<td>A. Set Item Goal Loc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 In Aisle</td>
<td>1. Picker Location Check</td>
<td>1. Picker Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td>A. Entering?, B. Exiting?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Item Location Check</td>
<td>2. Going To Inter, 3. Going To Item</td>
<td>1. Item Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td>A. Same Aisle?, B. Diff Aisle?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Going To Inter</td>
<td>1. Item Location Check</td>
<td>1. Item Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Going To Item</td>
<td>1. Item Location Check</td>
<td>1. Item Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td>A. Same Aisle?, B. Diff Aisle?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Get Ship Goal</td>
<td>2. Going To Set, 3. Going to S Int</td>
<td>1. Item Location Check</td>
<td>A. Set Inter Goal Loc</td>
<td>A. Entering?, B. Exiting?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Going To S Int</td>
<td>1. Item Location Check</td>
<td>1. Item Location Check</td>
<td>A. Set S Inter Goal Loc</td>
<td>A. Same Aisle?, B. Diff Aisle?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Inter Traffic Check</td>
<td>1. Get Int Goal Locations</td>
<td>3. Move Thru Inter</td>
<td>A. Scan In Front, B. Scan Crossing Lanes, C. Scan Entering Aisle, D. Scan Entering Highway, E. Blocked Check</td>
<td>A. Move?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Get Int Goal Locations</td>
<td>2. Inter Traffic Check</td>
<td>1. Get Int Goal Locations</td>
<td>3. Move Thru Inter</td>
<td>A. Scan In Front, B. Scan Crossing Lanes, C. Scan Entering Aisle, D. Scan Entering Highway, E. Blocked Check</td>
<td>A. Move?</td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Move Thru Inter</td>
<td>2. Inter Traffic Check</td>
<td>4. Get Next Goal</td>
<td>A. Move To Temp</td>
<td>A. At Temp?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Get Next Goal</td>
<td>3. Move Thru Inter</td>
<td>4. Get Next Goal</td>
<td>A. Move To Temp</td>
<td>A. At Temp?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Get Next Goal</td>
<td>3. Move Thru Inter</td>
<td>4. Get Next Goal</td>
<td>A. Move To Temp</td>
<td>A. At Temp?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Figure 16: CM4S Diagram Page 3 - Decision Shapes

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Transitions To</th>
<th>Behavior</th>
<th>Pseudo Code - Condition</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Stop</th>
<th>Variable</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision A</td>
<td>Highway? 2. In Highway</td>
<td></td>
<td></td>
<td>If the current location is in a highway (y coordinating 125, 175, 1025 or 1075 and atInter = 0), then follow the highway logic.</td>
<td>If (CurY == 125 or 175 or 1025 or 1075 and atInter = 0) then transition to In Highway.</td>
<td></td>
<td>on entry</td>
<td>G. CurY, B. atInter</td>
</tr>
<tr>
<td></td>
<td>Intersect? 3. At Intersections</td>
<td></td>
<td></td>
<td>If the current location is a intersection (atInter = 1), then follow the intersection logic.</td>
<td>If (atInter == 1) then transition to At Intersection.</td>
<td></td>
<td>on entry</td>
<td>B. atInter</td>
</tr>
<tr>
<td>Decision C</td>
<td>Aisle? 4. In Aisle</td>
<td>If the current location is an aisle y coordinating 125, 175, 1025 or 1075 and atInter = 0, then follow the aisle logic.</td>
<td>If (CurY == 125 or 175 or 1025 or 1075 and atInter = 0) then transition to In Aisle.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>G. CurY, B. atInter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Same Aisle? 5. Going To Item</td>
<td>If the current item is in the same aisle as the picker, then set goal to the item location.</td>
<td>If (CurItemAisle == CurAisle) then transition to Going To Item.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>O. CurItemAisle, G. CurAisle</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off Aisle? 6. Going To Inter</td>
<td>If the current item is not in the same aisle, then the picker must proceed to the nearest intersection.</td>
<td>If (CurItemAisle != CurAisle) then transition to Going To Inter.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>O. CurItemAisle, G. CurAisle</td>
<td></td>
</tr>
<tr>
<td>Decision D</td>
<td>Ship? 7. At Shipping</td>
<td>If the current location is in the shipping area and you are not leaving (y coord &gt;= 1125 and atInter = 0) then follow the shipping logic.</td>
<td>If (CurY &gt;= 1125 and atInter = 0) then transition to At Shipping.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>G. CurY, B. atInter</td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Entering? 2. Going To Set</td>
<td>If the current item y coord &lt; 1125, then set goal to the shipping location.</td>
<td>If (CurItemY &lt; 1125) then transition to Going To Set.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>C. CurItemY</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Exiting? 3. Going To Set</td>
<td>If the current item y coord &gt; 1125, then the picker is exiting the shipping area and entering an intersection.</td>
<td>If (CurItemY &gt; 1125) then transition to Going To Inter.</td>
<td></td>
<td></td>
<td>on entry</td>
<td>C. CurItemY</td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Move? 4. Move Through Intersection</td>
<td>If the picker is no longer delayed, or they are waved to move, or the picker has been delayed for longer than the DeadBreakTime, then they are moving through the intersection.</td>
<td>If (waitDelay == 0 or wavedAt == 1 or time - startDeadLock &gt; DeadBreakTime) then transition to Move Through Intersection. 0.1</td>
<td></td>
<td></td>
<td>on entry</td>
<td>T. DeadBreakTime, B. waited, A. waitedAt, Q. startDeadLock, T. DeadBreakTime</td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>At Temp? 4. Get Next Goal</td>
<td>If the picker has reached the temporary goals, then it is ready to set the final goals of the intersection and move to them.</td>
<td>If (CurTemp == Temp and CurTemp == Temp) then transition to Get Next Goal. 0.1</td>
<td></td>
<td></td>
<td>on entry</td>
<td>T. CurX, G. CurY, C. Temp, D. Temp</td>
<td></td>
</tr>
<tr>
<td>Decision A</td>
<td>Begin Move? 4. Movement</td>
<td>If the next goal location has been set and the picker is ready to move, then transition to general movement.</td>
<td>If (readyToMove == 1) then transition. After every action executed in 3. Get Next Goal Location. On entry</td>
<td>On exit</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Based on the environment layout. Highways are quick ways to move around the DC.*
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable A</td>
<td>readyToMove</td>
<td>3. Get Next Goal Location</td>
<td>Describes whether a goal has been set. 1=yes, 0=no.</td>
<td>N/A</td>
</tr>
<tr>
<td>Variable B</td>
<td>atInter</td>
<td>3. Get Next Goal Location</td>
<td>Describes whether the picker is at an intersection. 1=yes, 0=no</td>
<td>N/A</td>
</tr>
<tr>
<td>Variable C</td>
<td>waitDelay</td>
<td>3. At Intersection</td>
<td>Describes if there is a traffic in my path</td>
<td></td>
</tr>
<tr>
<td>Variable D</td>
<td>wavedAt</td>
<td>3. At Intersection</td>
<td>Describes if another picker has &quot;waved&quot; for me to move because I am in their way</td>
<td>Based on observation.</td>
</tr>
<tr>
<td>Variable E</td>
<td>TempX</td>
<td>3. At Intersection</td>
<td>Temporary x coordinates to move the picker through the intersection.</td>
<td></td>
</tr>
<tr>
<td>Variable F</td>
<td>TempY</td>
<td>3. At Intersection</td>
<td>Temporary y coordinates to move the picker through the intersection.</td>
<td>Based on observation.</td>
</tr>
</tbody>
</table>
### Action Shapes

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Pseudo Code - Function</th>
<th>Pseudo Code - Update</th>
<th>Pseudo Code - Start</th>
<th>Pseudo Code - Stop</th>
<th>Variables</th>
<th>Sequence within Block</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action A: Scan In Front</td>
<td>1. Check In</td>
<td>1. Scan In Front</td>
<td>Check if picker is in front of the picker's location.</td>
<td><code>CloseToBreak: 3.0*DecelBreak</code> if picker length to avoid for head-on collisions and get the heading.</td>
<td>on entry</td>
<td>G. DecelBreak, A. closestPicker</td>
<td>1</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Accelerate Move</td>
<td>2. Accelerate Move</td>
<td>Move towards goal at TravelSpeed/2.</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(TravelSpeed/2) 0.1</td>
<td>on entry</td>
<td>on exit</td>
<td>H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>1</td>
<td>Moving half the speed for the entire time it takes to accelerate. This attempt to capture the average speed traded while accelerating.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Max Move</td>
<td>3. Max Move</td>
<td>Move towards goal at TravelSpeed.</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(TravelSpeed) 0.1</td>
<td>on entry</td>
<td>on exit</td>
<td>H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>1</td>
<td>Average moving speed based on observation and measurement is approximately 5mph.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Decelerate Move</td>
<td>4. Decel Move</td>
<td>Move towards goal at TravelSpeed/2.</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(TravelSpeed/2) 0.1</td>
<td>on entry</td>
<td>on exit</td>
<td>H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>1</td>
<td>Moving half the speed for the entire time it takes to come to a stop. This attempt to capture the average speed traded while decelerating.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Set No Delay</td>
<td>5. No Delay</td>
<td>Set traffic delay to zero.</td>
<td><code>trafficDelay = 0;</code></td>
<td>on entry</td>
<td>B. trafficDelay</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Decelerate</td>
<td>6. Slow Down</td>
<td>Decelerate and reset initial starting coordinates (this is sighted time it would take to reach full speed again)</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(TravelSpeed/2); IniX = CurX; IniY = CurY; 0.1</td>
<td>on entry</td>
<td>on exit</td>
<td>H. GoalX, I. GoalY, C. TravelSpeed; F. CurX, G. CurY, R. IniX, S. IniY</td>
<td>1</td>
<td>Slow down enough to stop if needed. Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Stop &amp; Walk</td>
<td>7. Stop &amp; Walk</td>
<td>Stop moving and record the distance to the item.</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(0); <code>switchNeeded = 1;</code></td>
<td>on entry</td>
<td>H. GoalX, I. GoalY, P. switchNeeded</td>
<td>1</td>
<td>Based on Observation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Stop to Switch 6. Switch Lane</td>
<td>Step moving and transition to Switch Lane/Movement</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(0); <code>switchNeeded = 1;</code></td>
<td>on entry</td>
<td>H. GoalX, I. GoalY, P. switchNeeded</td>
<td>1</td>
<td>Based on Observation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Stop for Traffic</td>
<td>9. Stop for Traffic</td>
<td>Step moving and begin waiting for traffic to clear.</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(0); startDeadlock = time;</td>
<td>on entry</td>
<td>H. GoalX, I. GoalY, Q. startDeadlock</td>
<td>1</td>
<td>Based on Observation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Action A: Stop in Switch</td>
<td>10. Stop in Switch</td>
<td>Step moving and transition to Switch Lane/Movement</td>
<td><code>MoveIncrementallyTo(GoalX, GoalY)</code> at Rate(0); <code>switchNeeded = 1;</code></td>
<td>on entry</td>
<td>H. GoalX, I. GoalY, P. switchNeeded</td>
<td>1</td>
<td>Based on Observation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Master Name</td>
<td>Displayed Text</td>
<td>From</td>
<td>To</td>
<td>Actions</td>
<td>Decisions</td>
<td>Recorders</td>
<td>Variables</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
<td>-------------------------------</td>
<td>-------------------------------</td>
<td>------------------</td>
<td>-------------------------------</td>
<td>------------------</td>
<td>-------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1.3.1.1.4 Movement</td>
<td>3. Get Next Goal Location</td>
<td>5. Check Location</td>
<td></td>
<td>A. At Goal?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Standard Movement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Check In Front</td>
<td>2. No Traffic</td>
<td>2. No Traffic</td>
<td>A. Scan In Front</td>
<td>A. No Picker?, B. Follow Picker?, C. Picker Head On?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 No Traffic</td>
<td>1. Check In Front</td>
<td>1. Check In Front</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Accel</td>
<td>1. Speed Check</td>
<td></td>
<td></td>
<td>A. Accel Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Max</td>
<td>1. Speed Check</td>
<td></td>
<td></td>
<td>A. Max Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Decel</td>
<td>1. Speed Check</td>
<td></td>
<td></td>
<td>A. Decel Move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Speed Check</td>
<td></td>
<td>2. Accel, 3. Max, 4. Decel</td>
<td></td>
<td>A. Accel?, B. At Max?, C. Decel?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Head On Logic</td>
<td>1. Check Front</td>
<td>2. No Traffic, 2. Switch Lane Movement</td>
<td></td>
<td>A. No Delay?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 Switch Lane Movement</td>
<td>1. Standard Movement, 9, Switch Lane, 6, Switch Lane</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>1 Check Dist</td>
<td></td>
<td>2. No Delay, 3. Slow Down</td>
<td></td>
<td>A. Goal Close?, B. Picker Close?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>2 No Delay</td>
<td>1. Check Dist, 3. Slow Down</td>
<td></td>
<td></td>
<td>A. Set No Delay,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>3 Slow Down</td>
<td>1. Check Dist</td>
<td>2. No Delay, 4. Eval Options</td>
<td>A. Decelerate</td>
<td>A. P. not w/in stop, B. P. w/in stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Eval Options</td>
<td>3. Slow Down</td>
<td>5. Walk To Item, 6. Switch Lane, 7. Traffic Delay</td>
<td>A. Walk To Item?, B. Switch Lane?, C. Must Stop?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>5 Walk To Item</td>
<td>4. Eval Options</td>
<td></td>
<td></td>
<td>A. Stop &amp; Walk</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>6 Switch Lane</td>
<td>4. Eval Options</td>
<td>2. Switch Lane Movement</td>
<td>A. Stop to Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>7 Traffic Delay</td>
<td>4. Eval Options</td>
<td>2. No Delay</td>
<td>A. Stop for Traffic</td>
<td>A. Clear?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>4 Following Logic</td>
<td>1. Check In Front</td>
<td>2. No Traffic, 2. Switch Lane Movement</td>
<td></td>
<td>A. No Delay?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block</td>
<td>9 Switch Lane</td>
<td>7. Aisle Logic</td>
<td>2. Switch Lane Movement</td>
<td>A. Stop to Switch</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Figure 21: CM4S Diagram Page 4 - Decision Shapes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>Do Picker?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Follow Picker?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Picker Head?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Check Traffic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>A Move?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>No Traffic?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Goal Close?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>P. not in stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>P. not in stop</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>P. Walk To Item</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Walk To Item?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>All Stop?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Switch Lane?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>Clear?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>No Delay?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>No Goal?</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Source Information

- **Decision**: Based on observation.
- **No Traffic?**: If the closest picker is no longer in front of me or the deadlock time on entry F, transition to Head Out.
- **Goal Close?**: If the distance to the closest picker is further than the distance it would take you to stop (DecelBreak), then there is no need to stop yet.
- **P. not in stop?**: If the distance to the closest picker is not within 2 picker's length (200"), then there is no need to stop yet.
- **Decel?**: If the distance to the goal coordinates is less than DecelBreak, then the picker is decelerating.
- **Delay?**: If the distance to the goal coordinates is greater than Delay, then the picker must stop and wait for the traffic to clear.
- **Switch Lane?**: If the closest picker is not moving or the picker moves to the right, then go to switch lanes.
- **Clear?**: If the closest picker is no longer in front of me or the dead lock time has elapsed, there is no more delay.
- **Switch Lane Movement**: If the picker has reached goal, then go to switch lanes.
- **Clear?**: If the distance to the closest picker is greater than Clear, then there is no more delay.
- **All Stop?**: If all of the other decisions are not selected, then the picker must stop, and wait for the traffic to clear.
- **Goal Close?**: If the current goal is an item and the item is within the walk-to-pick distance, then park and wait for the item.
- **Walk To Item**: If the current goal is an item and the item is not within the walk-to-pick distance, then park and wait for the item.
- **Switch Lane**: If the closest picker is not moving or the picker moves to the right, then go to switch lanes.
- **Clear?**: If the closest picker is no longer in front of me or the dead lock time has elapsed, there is no more delay.
- **Switch Lane Movement**: If the picker has reached goal, then go to switch lanes.
<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>Member Of</th>
<th>Behavior</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable A</td>
<td>closestPicker</td>
<td>4. Movement</td>
<td>Describes the picker closest to this picker</td>
<td></td>
</tr>
<tr>
<td>Variable B</td>
<td>trafficDelay</td>
<td>4. Movement</td>
<td>Describes if there is a traffic delay. 1=yes, 0=no</td>
<td></td>
</tr>
<tr>
<td>Action A</td>
<td>Get No Delay</td>
<td>2</td>
<td>No Delay</td>
<td>Stop traffic delay to zero.</td>
</tr>
<tr>
<td>----------</td>
<td>--------------</td>
<td>---</td>
<td>----------</td>
<td>-----------------------------</td>
</tr>
</tbody>
</table>

Action A: Stop & Walk

Action A: Stop to Switch

Action A: Stop for Traffic

Action A: Decelerate

Action A: Get Lane Switch Goal & Start Block Time

Action A: Get Lane Switch Goal

Action A: Scan In Front to Possibly Walk

Action A: Scan Up and Down Entering Lane

Action A: Move To New Switch Goal

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Set No Delay</td>
<td>trafficDelay=0; on entry</td>
<td>A. trafficDelay=0;</td>
<td>A. trafficDelay=0;</td>
<td>A. trafficDelay=0;</td>
<td>A. trafficDelay=0;</td>
</tr>
<tr>
<td>Decelerate</td>
<td>MoveIncrementallyTo(GoalX, GoalY) at Rate(TravelSpeed/2); IniX=CurX; IniY=CurY; trafficDelay=0; 0.1 on entry on exit</td>
<td>F. CurX, G. CurY, H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>F. CurX, G. CurY, H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>F. CurX, G. CurY, H. GoalX, I. GoalY, C. TravelSpeed</td>
<td>F. CurX, G. CurY, H. GoalX, I. GoalY, C. TravelSpeed</td>
</tr>
<tr>
<td>Get Lane Switch Goal</td>
<td>Based on the picker's heading orientation, select the appropriate location to move to. MoveSwitchX() in the direction of the picker's heading and one to the adjacent aisle. Set deadlock time. Start DecelBreak; IniX=CurX; IniY=CurY;</td>
<td>set_switch_coordinates(heading, CurX, CurY)-&gt;GoalX, GoalY; startDeadLock=time; trafficDelay=1; on entry</td>
<td>set_switch_coordinates(heading, CurX, CurY)-&gt;GoalX, GoalY; startDeadLock=time; trafficDelay=1; on entry</td>
<td>set_switch_coordinates(heading, CurX, CurY)-&gt;GoalX, GoalY; startDeadLock=time; trafficDelay=1; on entry</td>
<td>set_switch_coordinates(heading, CurX, CurY)-&gt;GoalX, GoalY; startDeadLock=time; trafficDelay=1; on entry</td>
</tr>
<tr>
<td>Scan In Front to Possibly Walk</td>
<td>MoveIncrementallyTo(GoalX, GoalY) at Rate(0); waitDelay=0, GoalX=CurX, GoalY=CurY+/-50, CurItemX=GoalX, CurItemY=GoalY, walkTo=1; walkingDist=Distance(CurX, CurY, GoalX, GoalY); on entry</td>
<td>B. waitDelay=0, G. CurX, H. GoalX, I. GoalY, Q. startDeadLock</td>
<td>B. waitDelay=0, G. CurX, H. GoalX, I. GoalY, Q. startDeadLock</td>
<td>B. waitDelay=0, G. CurX, H. GoalX, I. GoalY, Q. startDeadLock</td>
<td>B. waitDelay=0, G. CurX, H. GoalX, I. GoalY, Q. startDeadLock</td>
</tr>
<tr>
<td>Scan Up and Down Entering Lane</td>
<td>If (scan_up_lane_over-&gt;returns no picker and scan_down_lane_over-&gt;returns no picker) then trafficDelay=0; 0.1 on entry on exit</td>
<td>B. waitDelay=0, G. DecelBreak, H. IniX, I. IniY, R. IniX, S. IniY</td>
<td>B. waitDelay=0, G. DecelBreak, H. IniX, I. IniY, R. IniX, S. IniY</td>
<td>B. waitDelay=0, G. DecelBreak, H. IniX, I. IniY, R. IniX, S. IniY</td>
<td>B. waitDelay=0, G. DecelBreak, H. IniX, I. IniY, R. IniX, S. IniY</td>
</tr>
</tbody>
</table>

**Figure 24: CM4S Diagram Page 5 - Action Shapes**
### Figure 25: CM4S Diagram Page 5 - Block Shapes

<table>
<thead>
<tr>
<th>Master Name</th>
<th>Displayed Text</th>
<th>From</th>
<th>To</th>
<th>Actions</th>
<th>Decisions</th>
<th>Recorders</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1.3.1.1.4.1.4</td>
<td>Following Logic</td>
<td>1. Check In Front</td>
<td>2. No Traffic</td>
<td>A. No Delay</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1</td>
<td>Check Dist</td>
<td>1. Check Dist</td>
<td>2. No Delay, 3. Picker Moving Check</td>
<td>A. P. not Close?, B. Picker Close?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 2</td>
<td>No Delay</td>
<td>1. Check Dist, 4. Slow Down, 6. Traffic Delay</td>
<td>A. Set No Delay</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 7</td>
<td>Aisle Logic</td>
<td>3. Picker Moving Check</td>
<td>4. Slow Down, 8. Walk To Item, 9. Switch Lane</td>
<td>A. P. Not w/in Stop?, B. Walk to Item?, C. Switch Lane?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 8</td>
<td>Walk To Item</td>
<td>7. Aisle Logic</td>
<td>A. Stop &amp; Walk</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 9</td>
<td>Switch Lane</td>
<td>7. Aisle Logic</td>
<td>2. Switch Lane Movement</td>
<td>A. Stop to Switch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 6</td>
<td>Traffic Delay</td>
<td>5. Highway Logic</td>
<td>2. No Delay</td>
<td>A. Stop for Traffic</td>
<td>A. Clear?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 4</td>
<td>Slow Down</td>
<td>3. Picker Moving Check</td>
<td>2. No Delay</td>
<td>A. Decelerate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1.3.1.1.4.2</td>
<td>Switch Lane Movement</td>
<td>1. Standard Movement, 9. Switch Lane, 6. Switch Lane</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1</td>
<td>Get Lane Switch Goal</td>
<td>2. Switch Traffic Check</td>
<td>A. Set Lane Switch Goal</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 2</td>
<td>Switch Traffic Check</td>
<td>1. Get Lane Switch Goal</td>
<td>3. Move Thru Inter</td>
<td>A. Scan In Front to Possibly Walk, B. Scan Up and Down Entering Lane</td>
<td>A. Move?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 3</td>
<td>Switch Lanes</td>
<td>2. Switch Traffic Check</td>
<td>4. Get Next Goal</td>
<td>A. Move To New Switch Goal</td>
<td>A. Switch Done?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Figure 26: CM4S Diagram Page 5 - Decision Shapes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision A</td>
<td>2. No Traffic</td>
<td>If there is no traffic delay, then proceed to move.</td>
<td></td>
<td>If (trafficDelay==0) then transition to No Traffic</td>
<td>01</td>
<td>on entry</td>
<td>on exit</td>
<td>trafficDelay</td>
<td>1</td>
<td>Based on observation</td>
</tr>
<tr>
<td>Decision B</td>
<td>2. No Delay</td>
<td>If the distance to the closest picker is not within DecelBreak plus the picker's length (100), then there is no need to stop yet.</td>
<td></td>
<td>If DistinctCurX, CurY &gt;= closestPicker/DecelBreak+100 then transition to No Delay</td>
<td>on entry</td>
<td>F. CurX, G. CurY, G. DistinctCurX, A. closestPicker</td>
<td>1</td>
<td>This buffer is to ensure the picker has enough time to stop if the picker in front stops.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision C</td>
<td>P. not Close?</td>
<td>If the distance to the closest picker is not within DecelBreak plus the picker's length (100), then determine if the picker is moving.</td>
<td></td>
<td>If DistinctCurX, CurY &gt; closestPicker/DecelBreak+100 then transition to Picker Moving Check</td>
<td>on entry</td>
<td>F. CurX, G. CurY, G. DistinctCurX, A. closestPicker</td>
<td>2</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision D</td>
<td>Moving &amp; Faster?</td>
<td>If the picker in front is moving and they are moving faster or the same speed as me, then there is no delay.</td>
<td></td>
<td>If (isMoving(closestPicker)==true and rate(closestPicker) &gt;= my_rate) then transition to No Delay</td>
<td>on entry</td>
<td>F. CurX, G. CurY, G. rate(closestPicker), A. closestPicker</td>
<td>1</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision E</td>
<td>Walk to Item?</td>
<td>If the current goal is an item and the item is within the walk-to-item distance, then park and walk to the item.</td>
<td></td>
<td>If GoalX==CurItemX and GoalY==CurItemY and Distance(CurX, CurY, GoalX, GoalY) &lt; WalkToItemDist then transition to Walk To Item;</td>
<td>on entry</td>
<td>F. CurX, G. CurY, I. WalkToItemDist, H. GoalX, I. GoalY, B. CurItemX, C. CurItemY</td>
<td>2</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision F</td>
<td>Clear?</td>
<td>If the closest picker is no longer in front of me or the deadlock time has expired, then there is no more delay.</td>
<td></td>
<td>If out_of_way(closestPicker)==true or time-startDeadLock &gt;= DeadBreakTime then transition to No Delay;</td>
<td>on entry</td>
<td>Q. startDeadLock, A. closestPicker, T. DeadBreakTime</td>
<td>1</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision G</td>
<td>Moving &amp; Slower?</td>
<td>If the picker in front is moving and they are moving slower than me, then slow down to match their speed.</td>
<td></td>
<td>If (isMoving(closestPicker)==true and rate(closestPicker) &lt; my_rate) then transition to Slow Down;</td>
<td>on entry</td>
<td>A. closestPicker</td>
<td>2</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision H</td>
<td>Highway Logic</td>
<td>If the picker in front is not moving and this picker is in the highway, then transition to the highway following logic.</td>
<td></td>
<td>If isMoving(closestPicker)==false and my_location=highway then transition to High Way Logic;</td>
<td>on entry</td>
<td>A. closestPicker</td>
<td>3</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision I</td>
<td>Aisle Logic</td>
<td>If the picker in front is not moving and this picker is in the aisle, then transition to the aisle following logic.</td>
<td></td>
<td>If isMoving(closestPicker)==false and my_location=aisle then transition to Aisle Logic;</td>
<td>on entry</td>
<td>A. closestPicker</td>
<td>4</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision J</td>
<td>Not with Stop</td>
<td>If the picker in front is not within a picker's length (100) then slow down but there is no delay.</td>
<td></td>
<td>If (Distance(CurX, CurY, closestPicker) &gt;= 100) then transition to Slow Down</td>
<td>on entry</td>
<td>A. closestPicker, F. CurX, G. CurY</td>
<td>1</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision K</td>
<td>Traffic Delay</td>
<td>If the picker in front is within a picker's length (100) then stop.</td>
<td></td>
<td>If DistinctCurX, CurY &lt; closestPicker/DecelBreak+100 then transition to Traffic Delay;</td>
<td>on entry</td>
<td>A. closestPicker, F. CurX, G. CurY</td>
<td>2</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision L</td>
<td>Not with Delay</td>
<td>If the picker in front is not within a picker's length (100) then slow down but there is no delay.</td>
<td></td>
<td>If (Distance(CurX, CurY, closestPicker) &gt;= 100) then transition to Slow Down</td>
<td>on entry</td>
<td>A. closestPicker, F. CurX, G. CurY</td>
<td>1</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision M</td>
<td>Switch Lane</td>
<td>If there is no longer traffic delay or the deadlock time expires then switch lanes.</td>
<td></td>
<td>If trafficDelay==0 or time-startDeadLock &gt;= DeadBreakTime then transition to Switch Lane;</td>
<td>on entry</td>
<td>A. trafficDelay, Q. startDeadLock, T. DeadBreakTime</td>
<td>3</td>
<td>Based on observation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>