Cell-based Representation and Analysis of Spatial Resources in Construction Simulation

Hong Pang

A Thesis

in

The Department

of

Building, Civil and Environmental Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Applied Science (Building Engineering) at
Concordia University
Montreal, Quebec, Canada

June 2007

Your file Votre référence
ISBN: 978-0-494-34762-1
Our file Notre référence
ISBN: 978-0-494-34762-1

ABSTRACT

Cell-based Representation and Analysis of Spatial Resources in Construction Simulation

Hong Pang

Extensive research efforts have been made in construction simulation. Nevertheless, work space is neglected or not explicitly represented, and spatial conflicts are not considered in simulation tools. Accordingly, finding possible solutions to spatial conflicts is not integrated within the simulation. Spatial conflicts should be considered and properly resolved because they affect construction performance in terms of safety and productivity, especially when spatial constraints are crucial to the project. A conflict-free simulation provides a better means for the practitioners to check the simulation results and compare various solutions. In addition, explicit representation of space is required to support planners with the evaluation of spatial organization of more efficient and safer workspaces on construction sites. The objective of this research is to propose a cell-based simulation method considering spatial constraints for the decision-makers to perform efficient spatial evaluation of site layouts, to identify and resolve spatial conflicts during construction operations, and then to facilitate the optimization of the site layout and construction resources from a variety of requirement perspectives that planners would normally consider in a real project. This method consists of the pre-processing, main-processing and post-processing phases, integrating site layout planning, simulation and animation of the simulation results. In this method, the spatial constraints, site layout patterns and resource allocations can be explicitly represented by cells. Consequently, spatial conflicts can be detected and resolved.

Two case studies were used to demonstrate the applications of the proposed method. The effects of spatial constraints were analyzed from multi-perspectives. Through the comparisons with those of MicroCYCLONE, the results of the cell-based method reflect the effects of spatial constraints. Moreover, the difference between the two simulation methods illustrates that in some cases the impact of spatial constraints should not be neglected.

ACKNOWLEDGEMENTS

Many have contributed to this research throughout the entire effort with their ideas, spiritual and finical supports. Without their help, encouragement, and active participation, this thesis would have never come to be. I am grateful to all.

I would like to express my sincere gratitude to my advisor, Professor Amin Hammad. Our initial discussions about construction simulation motivated me to devote myself to this interesting topic. Since then, he has been an incredible motivator and close friend, encouraging me to implement this work. I cannot forget the many weekends he dedicated to this research. His advice, guidance and confidence were an invaluable resource that always inspired me.

I am grateful to Professor Gabriel A. Wainer at Carleton University, the creator of the Cell-DEVS formalism and CD++. He opened his resource to this research and offered valuable advice on the cell-based simulation method. The suggestions made by Professor Tarek Zayed, regarding the modeling with MicroCYCLONE, improved this work significantly.

During these demanding years of study, the support of our Infra-group was of extreme importance. I would like to thank Cheng Zhang, Hui Wang and Linchuan Yang for their cooperation and contribution in the development of the cell-based simulation tool and my programming skills. Also, the help of all the other friends in this research is appreciated.

Most of all, I owe special thanks to my wife, Hui Shang. Her understanding and tolerance were the most important spiritual support that encouraged me to keep on moving forward. My parents have always been a continuous encouragement through all of my endeavors. My twin daughters were born at the beginning of this research. They joined me with the greatest joy and happiness. They are the witnesses of this work and the source of inspiration. To them, I dedicate this thesis.

TABLE OF CONTENTS

LIST OF FIGURES....x

LIST OF TABLES....xii

LIST OF ABBREVATIONS xii

CHAPTER 1 INTRODUCTION 1

1.1 GENERAL BACKGROUND.... 1

1.2 RESEARCH OBJECTIVES 2

1.3 THESIS ORGANIZATION.... 3

CHAPTER 2 LITERATURE REVIEW 5

2.1 SIMULATION IN THE CONSTRUCTION INDUSTRY.... 5

2.2 TYPES OF CONSTRUCTION SIMULATION TOOLS.... 6

2.2.1 Continuous and Discrete Simulation Systems 6

2.2.2 General-purpose and Special-purpose Simulation Tools.... 6

2.3 SIMULATION STRATEGY 7

2.4 CONSTRUCTION SIMULATION TOOLS 9

2.4.1 CYCLONE.... 10

2.4.2 STROBOSCOPE 12

2.4.3 SIMPHONY.... 13

2.4.4 Features of Construction Simulation Tools 15

2.5 VISUALIZING SIMULATED CONSTRUCTION OPERATIONS 15

2.5.1 2D Animation.... 16

2.5.2 3D Animation.... 17

2.6 SITE LAYOUT PLANNING 19

2.7 PROBLEM STATEMENT.... 22

2.8 CELL-BASED DISCRETE EVENT SIMULATION 25

2.8.1 Cellular Automata.... 25

2.8.2 Discrete-Event System Specification (DEVS).... 26

2.8.3 Cell-DEVS 28

2.9 SUMMARY AND CONCLUSIONS 30

CHAPTER 3 CELL-BASED SIMULATION METHOD OF CONSTRUCTION OPERATIONS 31

3.1 INTRODUCTION... 31

3.2 FEATURES OF CELL-DEVS MODELING 32

3.3 PROCESS OF CELL-BASED SIMULATION 33

3.4 PRE-PROCESSING 35

3.4.1 Analyzing Construction Activities and Identifying Involved Resources 35

3.4.2 Identifying and Defining DEVS and Cell-DEVS Models 35

3.4.3 Defining Relationships between Models 36

3.4.4 Deciding the Suitable Size of Cells and the Dimensions of Each Cell-DEVS Model 37

3.4.5 Defining the Layers and Codes for Each Model 39

3.4.6 Defining the Zones of Each Cell-DEVS Model 40

3.4.7 Developing Rules for Each Cell-DEVS Model 40

3.4.8 Developing DEVS Models 44

3.5 MAIN-PROCESSING 45

3.5.1 Deciding Site Layout Patterns and Resource combination 45

3.5.2 Initializing Resources 45

3.5.3 Running the Simulation 46

3.6 POST-PROCESSING 47

3.6.1 Verification and Validation 47

3.6.2 Visualizing the Cell-DEVS Models 48

3.6.3 Analyzing Simulation Results 48

3.7 SUMMARY AND CONCLUSIONS 49

CHAPTER 4 IMPLEMENTATION AND CASE STUDIES 50

4.1 INTRODUCTION... 50

4.2 CASE STUDY I – JACQUES CARTIER BRIDGE REHABILITATION PROJECT 50

4.2.1 Background 50

4.2.2 Data Cllection 53

4.2.3 Previous Study – a MocroCYCLONE Model 54

4.2.4 Previous Study – a Cell-based Model 56

4.2.5 Pre-processing 59

4.2.5.1 Analyzing Construction Activities and Identifying Involved Resources 59

4.2.5.2 Building Cell-based Simulation Models 60

4.2.5.3 Defining Relationships between Models 64

4.2.5.4 Deciding the Suitable Size of Cells and the Dimensions of Each Cell-DEVS Model 64

4.2.5.5 Defining the Layers and Codes for Each Cell-CEVS Model 66

4.2.5.6 Defining the Zones of Each Cell-DEVS Model 69

4.2.5.7 Developing Rules for Each Cell-DEVS Model 69

4.2.5.8 Developing DEVS Models 75

4.2.6 Main-processing 76

4.2.6.1 Site Layout Patterns 76

4.2.6.2 Initializing Resources 78

4.2.7 Post-processing 80

4.2.7.1 Verification 80

4.2.7.2 Visualizing the Cell-DEVS Models 86

4.2.7.3 Analyzing Simulation Results 88

4.3 CASE STUDY II - ANALYSIS OF EFFECT OF WORK ZONES ON TRAFFIC FLOW 96

4.3.1 Introduction 96

4.3.2 Developing Cell-based Simulation Model 97

4.3.3 Deciding the Size of Cells, the Layers and Codes for Each Cell-DEVS Model 99

4.3.4 Defining the Zones of Each Cell-DEVS Model 99

4.3.5 Developing Rules for the Cell-DEVS Model and Developing DEVS models 100

4.3.6 Visualizing the Cell-DEVS Model 101

4.3.7 Simulation Results and Discussions 103

4.4 SUMMARY AND CONCLUSIONS 107

CHAPTER 5 SUMMARY, CONTRIBUTIONS AND FUTURE WORK 109

5.1 SUMMARY AND CONCLUSIONS 109

5.2 CONTRIBUTIONS. 111
5.3 LIMITATIONS AND FUTURE WORK 113

REFERENCES 115

APPENDIX A: Source Codes of (.ma) File **128**
APPENDIX B: Source Codes of (.inc) File **136**
APPENDIX C: Example of Source Codes of DEVS Model **162**
APPENDIX D: Using Patterns for Developing Rules **164**
APPENDIX E: Introduction of DCD++ **165**
APPENDIX F: List of Publications **167**

LIST OF FIGURES

Figure 2.1 CYCLONE Modeling Elements 11

Figure 2.2 SIMPHONY Environment 14

Figure 2.3 Open, Loosely-coupled Visualization Methodology 17

Figure 2.4 VITASCOPE Animation Snapshot of a Construction Site 18

Figure 2.5 Examples of Explicit Representation of Space 21

Figure 2.6 Cell Representations 27

Figure 3.1 Process of Cell-based Simulation 34

Figure 3.2 Abstract Example of a Cell-based Simulation 35

Figure 3.3 Cell Representations of Objects 37

Figure 3.4 Using Different Size of Cells to Represent Objects 38

Figure 3.5 Using Layers and Codes to Indicate the Attributes of Cells 39

Figure 3.6 Neighbor Cells Representation on the Three Layers 41

Figure 3.7 Examples of Rules and Cell State Changes on the Three Layers 43

Figure 3.8 Simulation Panel 46

Figure 4.1 Jacques Cartier Bridge Rehabilitation Project 51

Figure 4.2 Data Used in Case Study I 53

Figure 4.3 MicroCYCLONE Model of the Jacques Cartier Bridge Rehabilitation Project 54

Figure 4.4 Relationships between Cell-based Models of the Jacques Cartier Bridge Rehabilitation Project 57

Figure 4.5 Cell-based Simulation Model of the Jacques Cartier Bridge Rehabilitation Project 61

Figure 4.6 The Three Layers Used to Model *Occupancy, Control*, and *ID* Attributes 66

Figure 4.7 Zone Division and the Rules for Changing Directions in Zone-1 69

Figure 4.8 Examples of Spatial Conflicts 71

Figure 4.9 Examples of Master-Slave Cells 73

Figure 4.10 Example of Rules for Master-slave cells 74

Figure 4.11 Worksite Layout of the Bridge Rehabilitation Project 76

Figure 4.12 Possible Patterns of Teams' Order and Layout 77

Figure 4.13 Initializing Resources in the Cell-DEVS Model 79

Figure 4.14 Example of (.log) Files 80

Figure 4.15 Example of (.out) Files 82

Figure 4.16 Example of (.drw) Files 83

Figure 4.17 Example of (.txt) Files 84

Figure 4.18 Snapshots of the *Occupancy* Layer of the *Bridge* Model 87

Figure 4.19 Delays of Different Site Layout Patterns 90

Figure 4.20 Sensitivity Analysis Using MircoCYCLONE and Cell-DEVS 93

Figure 4.21 Comparison of Productivity between MicroCYCLONE and Cell-based Simulation Method 94

Figure 4.22 Comparison of Unit Cost between MicroCYCLONE and Cell-based Simulation Method 94

Figure 4.23 Effect of Slower Turning Speed on Productivity 95

Figure 4.24 Asphalt Pavement Project 96

Figure 4.25 Simulation Model of the Traffic Flow under Asphalt Pavement Construction 98

Figure 4.26 Snapshots of the Animation of the Traffic Flow under different Lane-closure Scenarios 102

Figure 4.27 Comparisons of Traffic Flow with Different Lane-closure Scenarios 106

LIST OF TABLES

Table 4.1 Task Durations and IDs Used in MicroCYCLONE Model 55

Table 4.2 Summary of Models of Case Study I 65

Table 4.3 Codes Used in the Occupancy Layer of Case Study I 68

Table 4.4 Codes Used in the Control Layer of Case Study I 68

Table 4.5 Codes used in the *ID* Layer of Case Study I 68

Table 4.6 Codes Used in the three Layers of Case Study II 99

Table 4.7 Simulation Results of Traffic Flow under Different Lane-closure Scenarios 103

LIST OF ABBREVIATIONS

Abbreviation	Description
2D	Two-dimensional
3D	Three-dimensional
ACD	Activity Cycle Diagram
AOO	Activity Object-Oriented
AS	Activity Scanning
CA	Cellular Automata
CAD	Computer-Aided Design
Cell-DEVS	Cell-Discrete Event system Specification
DEVS	Discrete-Event systems Specification
GA	Genetic Algorithms
GIS	Geographic Information System
GUI	Graphical User Interface
GVSS	GIS-based Visual Simulation System
LoD	Level of Details
OO	Object-Oriented
PI	Process Interaction
SPS	Special-Propose Simulation
TSON	The number of Teams, Saws, OS trucks and NP trucks

CHAPTER 1

INTRODUCTION

1.1 GENERAL BACKGROUND

Construction operations are inherently complex with many uncertainties where each activity has its own variables, and a large number of factors affect them during the construction processes. Simulation, as an effective means, has been used to analyze and design construction operations for more than three decades. After the development of CYCLONE (Halpin 1973), considerable progress has been made in this area. Recently, the need to consider spatial constraints in construction simulation has received a wide recognition. Many researches recognized that space is a resource that is as important as money, time, materials, labor, and equipment (Tommelein et al. 1992). Spatio-temporal conflicts can delay construction activities, reduce productivity, or cause accidents that threaten the safety of workers (Guo 2002). However, work space is not explicitly represented or spatial constraints are ignored in simulation. Therefore, possible solutions to spatial conflicts are unavailable. Explicit representation of space will contribute to a more realistic simulation, allowing the users to better understand the site environment, discover conflicts and compare different solutions as if they were managing a real project.

Second, site layout planning is among the most challenging tasks of the construction planning processes, involving several steps of human interpretations and manipulation of data and knowledge (Tawfik and Fernando 2001). In the past, site layout planning heavily depended on the subjective experience of project managers. Nowadays, various methodologies, such as genetic algorithm, have been adopted to optimize the site layout. If the optimized or any other site layout could be explicitly represented, the process of site layout planning could be associated with simulation. Then, these site layouts could be further used to predict the performance of construction operations considering various alternative site layouts.

The third concern is visualization. Some visualization tools have been developed from iconic representation (Shi and Zhang 1999) to 3D animation (Kamat 2003). Since spatial conflicts are inevitable in construction operations, such visualization might be interrupted when conflicts occur. Obviously, if the detecting and resolving of conflicts could be done in simulation, an uninterrupted and conflict-free visualization would be realized to assist managers to recognize the interactions and interferences during construction processes.

It is desirable, therefore, to develop a method that can explicitly represent space in simulation, can analyze construction operations considering site layouts, and can detect/resolve the conflicts during construction operations.

1.2 RESEARCH OBJECTIVES

The efforts in this research are directed to propose a more realistic, space-involved simulation method for the decision-makers to perform efficient spatial evaluation of site layouts, to identify and resolve spatial conflicts during construction operations, and then to facilitate the optimization of the site layout and construction resource from a variety of requirement perspectives that planners would normally consider in a real project. The objectives of this research are summarized as follows:

(1) To develop a cell-based simulation method that introduces explicit representation of space and considers different construction site layout patterns.

(2) To investigate possible solutions to identify and resolve spatial conflicts and integrate these solutions into the proposed simulation method to achieve a conflict-free and uninterrupted animation performance.

(3) To provide case studies to establish the concept of the proposed method and analyze the effects of spatial constraints through which, the necessity and advantages of the cell-based simulation is demonstrated.

1.3 THESIS ORGANIZATION

This study will be presented as follows:

Chapter 2 Literature Review: This chapter presents the current situation of the construction simulation tools, the site layout planning approaches as well as the animation techniques, describing their limitations and inspirations. Then, a feasible solution — the cell-based simulation method and its related technologies are introduced.

Chapter 3 Cell-based Simulation Method of Construction Operations: This chapter porposes the cell-based simulation method for construction processes, which connects simulation with site layout planning and animation. The methodology of developing cell-based construction simulation models is discussed.

Chapter 4 Implementation and Case Studies: In this chapter, two cell-based simulation models are developed for the case studies of the Jacques Cartier Bridge rehabilitation project and the analysis of effect of work zones on traffic flow. The simulation results are compared and the effects of spatial constraints are analyzed to demonstrate the applicability and effectiveness of the proposed method.

Chapter 5 Summary, Contributions and Future Work: This chapter summarizes the present research work, presents gengeral conclusions, highlights its contributions, and suggests recommendations for the future research.

CHAPTER 2

LITERATURE REVIEW

2.1 SIMULATION IN THE CONSTRUCTION INDUSTRY

A simulation is the imitation of the operation of a real-world process or system over time (Banks and Carson 1984). One of the key powers of simulation is the ability to study the behavior of a system under specific conditions prior to the implementation of the system. Simulation permits to compare various alternatives without perturbing the real system and to address the uncertain and dynamic features of the system. The simulation results provide insight into a system for decision-making or operating performance evaluation. In construction, often the resource allocation decisions are made based more on subjective analysis – the previous experience of the construction manager – and less on objective analysis (Jen 2005). This practice might result in high risk, either physical or financial, due to the spatio-temporal variances. In addition, the operating performance is usually unknown to the concerned practitioners before construction so that timely and corrective measures cannot be facilitated. Simulation, however, offers a powerful tool to estimate the productivity for project planning and scheduling, identify bottlenecks or overflows in the production system, investigate equipment utilization and perform sensitivity analysis to adequately select resource combinations. Therefore, the productivity could be increased and the cost could be minimized. During the past decades, simulation has been applied to the construction industry to design and analyze construction processes.

2.2 TYPES OF CONSTRUCTION SIMULATION TOOLS

2.2.1 Continuous and Discrete Simulation Systems

Simulation systems can be categorized as discrete or continuous. In the case of continuous simulation, the state variables change continuously over time. A simple example is a moving truck whose position varies from time to time. In contrast with the continuous systems, in discrete systems, the state variables change only at a discrete set of points in time (Banks and Carson 1984). Most construction operation scenarios are candidates for discrete simulation. For example, in a bridge re-decking project (e.g., removing old sections and installing new panels with cranes and transporting them with trucks), the simulation tasks are defined by their start and end events, such as when a truck or a crane is available and when a truck leaves. During the time between the beginning and end points of the task, the extension of the crane's boom and the maneuver of the crane's superstructure could be ignored since they have no interactions with other tasks. Accordingly, the system model is concerned only with the movements that trigger or end the task at a set of points in time. These points represent discrete events along time. Because of the discrete nature of construction tasks, discrete simulation systems are dominant in this area.

2.2.2 General-purpose and Special-purpose Simulation Tools

A model is designed to represent a real-world system. The scope of models differentiates construction simulation tools into two groups — general-purpose and special-purpose simulation tools. Both groups of these tools facilitate the analysis of construction operations.

General-purpose tools can be used for a broad domain and almost any type of operations (Martinez and Ioannou 1999). This type of simulation tools usually provide some predefined constructs, such as modeling elements, to assist system modeling. The benefits of this type include: (1) They have a wide application range; (2) Users can develop and explore their own models for a variety of operations; and (3) The models developed previously could be reused with less effort so that modeling experience could be accumulated. However, users have to learn the simulation syntax because the effectiveness of a model partly depends on the familiarity with the particular tool.

Special-purpose simulation tools narrow their scopes to a particular domain such as an aggregate production plant (Hajjar and AbouRizk 1998), the earthwork planning (Martinez 1998) and tunneling operations (AbouRizk et al. 1999). These tools are properly customized to fit the specific needs and requirements of construction operations so that the simulation models are very close to the construction environments. Besides, special-purpose tools always provide a high-level model which includes familiar elements in the domain so that users do not need to be familiar with simulation languages. The drawback of this type is: the reusability of pre-designed models may be lost due to its narrow application range.

2.3 SIMULATION STRATEGY

The simulation strategy used by a simulation tool has a strong impact on the model development process as well as on the way a model is presented to the computer (Evans 1988; Zhang et al. 2005). A simulation strategy is the conceptual framework that guides model development and determines how the modeler views the system being modeled (Hooper 1986; Balci1 988). The main simulation strategies presently used are Process Interaction (PI) and Activity Scanning (AS).

PI assumes that the components or entities of a system progress through a sequence of processes that consist of various events and activities (Zhang et al. 2005). PI is particularly suited to modeling operations where the moving entities are differentiated by many attributes and where the machines or resources that serve these entities have few attributes, a limited number of states, and do not interact too much (Hooper 1986). Therefore, this strategy is primarily applied in manufacturing in which entities arrive, undergo some processing where they capture and release scarce resources, and then exit. Some well known simulation tools such as General Purpose System Simulator (GPSS), SLAM and SIMAN, adopted this modeling paradigm.

AS views a system as the composition of activities that are subject to scheduled times and activation conditions (Zhang et al. 2005). In the terms of Hooper (1986), AS is particularly adept at modeling systems with independent components subject to complex activity start-up conditions that require resources with distinct properties and that must collaborate according to highly dynamic rules. When the conditions are satisfied, the program performs the appropriate actions which typically include acquiring the resources that enabled the activity to start, determining how long the activity will last, holding the acquired resources for the duration of the activity, and releasing the resources when the activity ends (Martinez and Ioannou 1999). This feature makes it well suited to model construction operations. The extended forms of AS strategy include three-phase AS (Tocher 1963; Pidd 1998) and three-stage AS (Shi 1999). Martinez and Ioannou (1999) demonstrated the effect of PI and AS on modeling a typical push-loading earth-moving process and pointed out that the extended AS is the wave of the future for construction simulation.

From the software engineering point of view, applying object-oriented (OO) techniques (Booch 1991) to construction simulation have received a lot of publicity in recent years (Oloufa 1993; Shi 2000; Marzouk and Moselhi 2003). The power of OO lies in that the modular classes are easily modified and reused, and the library of classes can be built up for simulation modeling. Thus, the maintenance, flexibility and reusability of simulation tools will be greatly enhanced. Zhang et al. (2005) proposed an Activity Object-Oriented (AOO) simulation strategy which considers activities as objects essential to the system under study to resolve some strategy-related problems using graphical description.

2.4 CONSTRUCTION SIMULATION TOOLS

Simulation tools specially designed for construction operations have been used for more than three decades. Extensive research efforts have been made in this area and various simulation techniques have been developed to model and analyze construction processes and to help decision-makers for different purposes. CYCLONE (Halpin 1977), STROBOSCOPE (Martinez 1996) and Simphony (Hajjar and AbouRizk 1999) are three representative systems not only because of their widespread uses in construction industry and their distinct characteristics, but also because they have inspired the development of many other simulation systems.

2.4.1 CYCLONE

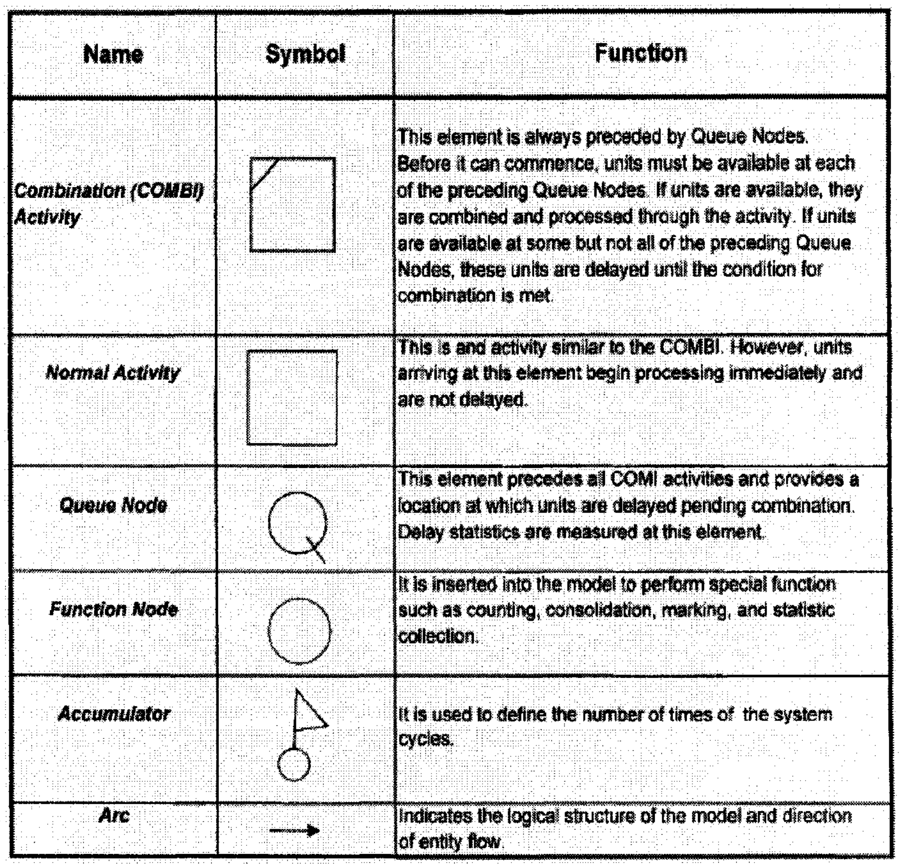
CYCLic Operation NEtwork (CYCLONE) is the oldest and most widely used general-purpose construction simulation tool. Halpin (1973) pioneered the effort by introducing the CYCLONE methodology. Considerable progress has been made in improving and utilizing this computer-based construction simulation tool since. Numerous implementations of CYCLONE have been documented in a wide range of publications including job site processes (Halpin 1977), earth-moving (Halpin and Riggs 1992), horizontal earth-boring (Gonzalez-Quevedo et al. 1993), tunneling (Touran and Asai 1987), heavy construction projects (Vanegas et al. 1993), concrete batch plant production (Zayed and Halpin 2001) and other real construction projects.

CYCLONE adopts the three-phase AS simulation strategy and enhances pure Activity Cycle Diagram[1] (ACD) with two different types of extensions: (1) "Conceptual" extensions allow for the explicit classification of activities as conditional (COMBI Activity) or bound (NORMAL Activity) and for the elimination of superfluous queues (Figure 2.1); and (2) ''Functional'' extensions enable the CYCLONE ACD to be a complete and unambiguous representation of a simulation model. These extensions include additional functions and nodes (GENERATE, CONSOLIDATE, COUNTER),

[1] ACD, developed by Tocher (1963), is a method of modeling the interactions of system entities. State (i.e., active or inactive) and links (logic) are defined to trace the simulation entities. The state change of a simulation entity is triggered by the state of other simulation entities.

probabilistic branching for activities followed by more than one other activity (Martinez and Ioannou 1999).

CYCLONE is a well-established, simple and easy to learn, and effective tool for modeling many simple construction operations. These advantages make it broadly used. Nevertheless, the fact that some simplifying assumptions (e.g., all resource flows and requirements are unitary and all resource units are indistinguishable and interchangeable) must be made when modeling makes it unable to recognize resource attributes, or to allow the inclusion of user-defines functions. (Gonzalez-Quevedo et al. 1993).



Name	Symbol	Function
Combination (COMBI) Activity	◸	This element is always preceded by Queue Nodes. Before it can commence, units must be available at each of the preceding Queue Nodes. If units are available, they are combined and processed through the activity. If units are available at some but not all of the preceding Queue Nodes, these units are delayed until the condition for combination is met.
Normal Activity	□	This is and activity similar to the COMBI. However, units arriving at this element begin processing immediately and are not delayed.
Queue Node	○	This element precedes all COMI activities and provides a location at which units are delayed pending combination. Delay statistics are measured at this element.
Function Node	○	It is inserted into the model to perform special function such as counting, consolidation, marking, and statistic collection.
Accumulator	⚐	It is used to define the number of times of the system cycles.
Arc	→	Indicates the logical structure of the model and direction of entity flow.

Figure 2.1 CYCLONE Modeling Elements
(Zayed and Halpin 2001)

Based on CYCLONE, a series of systems have been developed to improve and extend its capabilities. Lluch and Halpin (1981) developed a microcomputer version of CYCLONE named MicroCYCLONE. INSIGHT (Paulson 1987) is another system based on the CYCLONE elements with a more interactive interface. RESQUE (Chang and Carr 1987) advances the capability of CYCLONE in handing resources by recognizing distinctions among resources that flow through the same path. The significance of COOPS (Liu and Ioannou 1992) is its object-oriented feature (all resources are treated as individually identifiable objects) and its graphical user interface for modeling. CIPROS (Odeh et al. 1992) serves at both process level and project level, which contains an expandable knowledge base of construction techniques and methods. DISCO (Huang and Halpin 1995) allows the user to design CYCLONE models graphically, to set the stop clock and to change the resource assignment rule of the simulation.

2.4.2 STROBOSCOPE

STROBOSCOPE (Martinez and Ioannou 1994) is an acronym of STate and ResOurce Based Simulation of COnstruction ProcEsses. It is a general-purpose programming language with support for the three-phase AS.

Several significant features of STROBOSCOPE distinguish it from other similar simulation systems: (1) It is programmable, which makes it flexible and extensible for modeling complex construction processes; (2) It has the ability to access the dynamic state of the model and the properties of resources involved in an operation at run time, which makes it possible to differentiate multiple resources with different properties. The state of the model, such as the current simulation time, and the properties of resources such as the size, weight or cost could be reported in detail. This feature offers the necessary information for animation used in VITASCOPE (Kamat 2003) and allows tailoring the behaviour of modeling elements; and (3) It eliminates the limitations of many simplifying assumptions required by CYCLONE, which makes the model more realistic and more precise.

In contrast with these benefits, the drawback of STROBOSCOPE is evident. STROBOSCOPE programming language is specially designed for construction simulation. It provides a higher level of flexibility in contrasting models at the cost of an increase in system complexity. The user has to be familiar with the language syntax, the modeling mechanism as well as construction engineering. This disadvantage limits the use of STROBOSCOPE.

Examples of STROBOSCOPE include tunnelling operations with variance reduction techniques (Ioannou and Martinez 1996a), quarry operations (Martinez and Ioannou 1995) and impacts of changes in construction work (Cor 1998). In addition, STROBSCOPE has been used to build special purpose simulators (Martinez 1998).

2.4.3 SIMPHONY

The objective of SIMPHONY is to provide a standard, consistent and intelligent environment for both the development as well as the utilization of construction Special-Purpose Simulation (SPS) tools (Hajjar and AbouRizk 1998). Figure 2.2 depicts its overall environment. Developers utilize the Simphony Designer program to create SPS templates which are a collection of modeling elements targeted for a single domain and stored in the Modeling Element Library. Users utilize the Simphony Editor to create simulation models based on the existing SPS templates in the Modeling Element Library. Constructed simulation models and their simulated results are stored in and retrieved from the Construction Simulation Project Database, a relational database management system that other systems can access to extract the desired information. The User Model Library is another database that is utilized by users to store and retrieve reusable simulation models (Hajjar and AbouRizk 1999).

The use of libraries and database in SIMPHONY allows both the developers and users to inherit the the predefined functionality of a generic modeling element and then customize

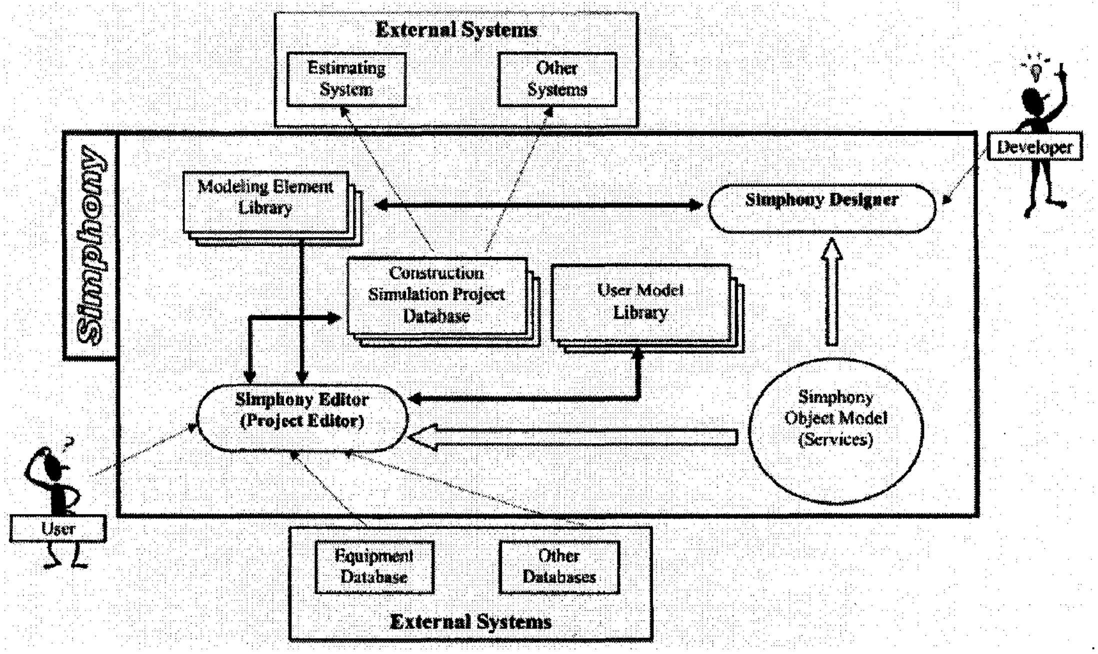


Figure 2.2 SIMPHONY Environment (Hajjar and AbouRizk 2002)

it as required. Therefore, the development of new models is greatly simplified and shortened. The reusibility of these elements are also enhanced.

2.4.4 Features of Construction Simulation Tools

In this research, it is found that modern construction simulation tools paid more attention to some new features such as flexibility, reusability and graphical interfaces. Flexibility emphasizes the ability of the system to adapt to a wide range of application requirements, to model complex operations and to report simulation results at different levels of detail. Reusability aims to store modules in a modular library that follows general creating rules and can be reused to simplify the development of new construction models. Some simulation tools provided graphical modeling icons and graphical interfaces to facilitate data input or output. These features greatly improved the convenience and ease of use.

2.5 VISUALIZING SIMULATED CONSTRUCTION OPERATIONS

Construction simulation tools introduced above typically cannot illustrate the modeled processes graphically, but provide results in the form of numerical or statistical data. This poses significant difficulty in communicating the results and the logic of simulation models, in debugging models during the development stage and in validating the simulation results. Visualization provides valuable insights into subtleties of the simulated construction operations that are otherwise non-quantifiable and non-presentable (Kamat 2003).Visualizing simulated construction operations, acting as the post-processor of construction simulation, is an effective means for simulation modeling and use. The developer of the simulation model can make sure that there are no errors in the coding (verification). On the other hand, the experts, field personnel, and decision makers can discover differences between the way they understand the operation and the way the model developer understands it (validation). Thus, the model, coupled with verification and validation, can be communicated effectively and becomes credible to be utilized in the construction industry (Kamat and Martinez 2001). Visualizing simulated construction operations, or construction animation in 2D or 3D, has become a necessary supplement of construction simulation.

2.5.1 2D Animation

Zhang et al. (2002) used 2D icons to represent the resources, which could move along the path between activities. Zhong et al. (2004) developed the Geographic Information System (GIS) -based Visual Simulation System (GVSS) to offer planning, visualizing, and querying capabilities of complex construction processes. Especially, a commercial animation tool PROOF, developed by Wollverine Software Corporation, has been used in the past to visually communicate modeled construction operations (Ioannou and Martinez 1996; Martinez 1998). Developers use the tool's built-in drawing capabilities to create layouts and specify resource movement paths. They can run a simulation model to generate the required the animation commands and store them in a trace file; then, PROOF post-processes the recorded animation commands to depict the dynamic state of the modeled processes (Henriksen 1998).

2.5.2 3D Animation

3D construction animation has attracted many researchers in this area. Kamat and Martinez (2001, 2003, 2004, and 2006) proposed an extensible and scalable 3D animation tool – VITASCOPE – to visualize the results of discrete-event construction simulation models. VITASCOPE provides an open loosely-coupled, animation language-based 3D visualization scheme (Figure 2.3) to describe the performance of construction operations in a virtual world. More important, it integrates geometric techniques to describe the 3D spatial configuration of virtual construction resources and an interference detection framework to identify and report spatial conflicts and/or collisions that occur among static, dynamic, and abstract construction resources in animations of simulated construction operations.

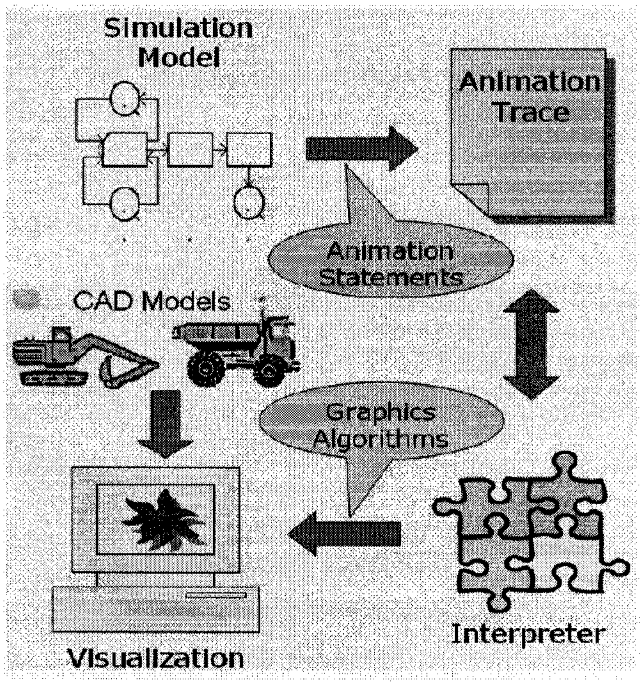


Figure 2.3 Open, Loosely-coupled Visualization Methodology (Kamat 2003)

The visualization scheme (Figure 2.3) of VITASCOPE is similar to that of PROOF. First, animation statements describe what has happened in simulation and store them in animation trace files. Second, each of the animation statements is interpreted by the implementation's interpreter and appropriate computer graphics algorithms are invoked to visually depict the operation, and finally, the invoked algorithms and routines manipulate the appropriate CAD models that represent construction resources after instantiating them in a 3D virtual world. Figure 2.4 shows an animation snapshot using VITASCOPE, which gives a vivid effect of the construction environment. A recent application of VITASCOPE can be found in Khoury et al (2006).

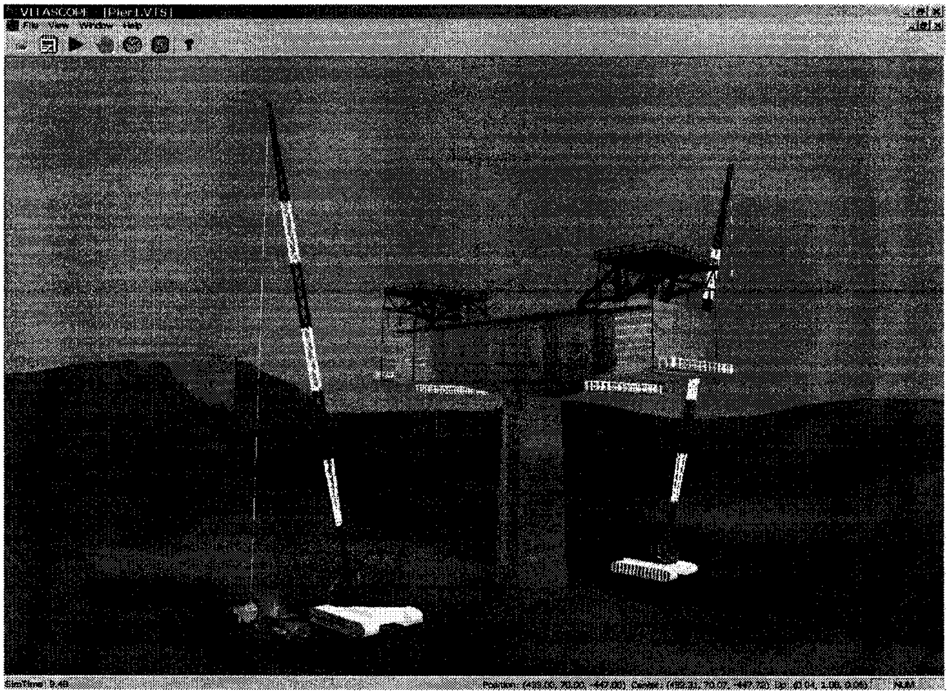


Figure 2.4 VITASCOPE Animation Snapshot of a Construction Site (Kamat 2003)

2.6 SITE LAYOUT PLANNING

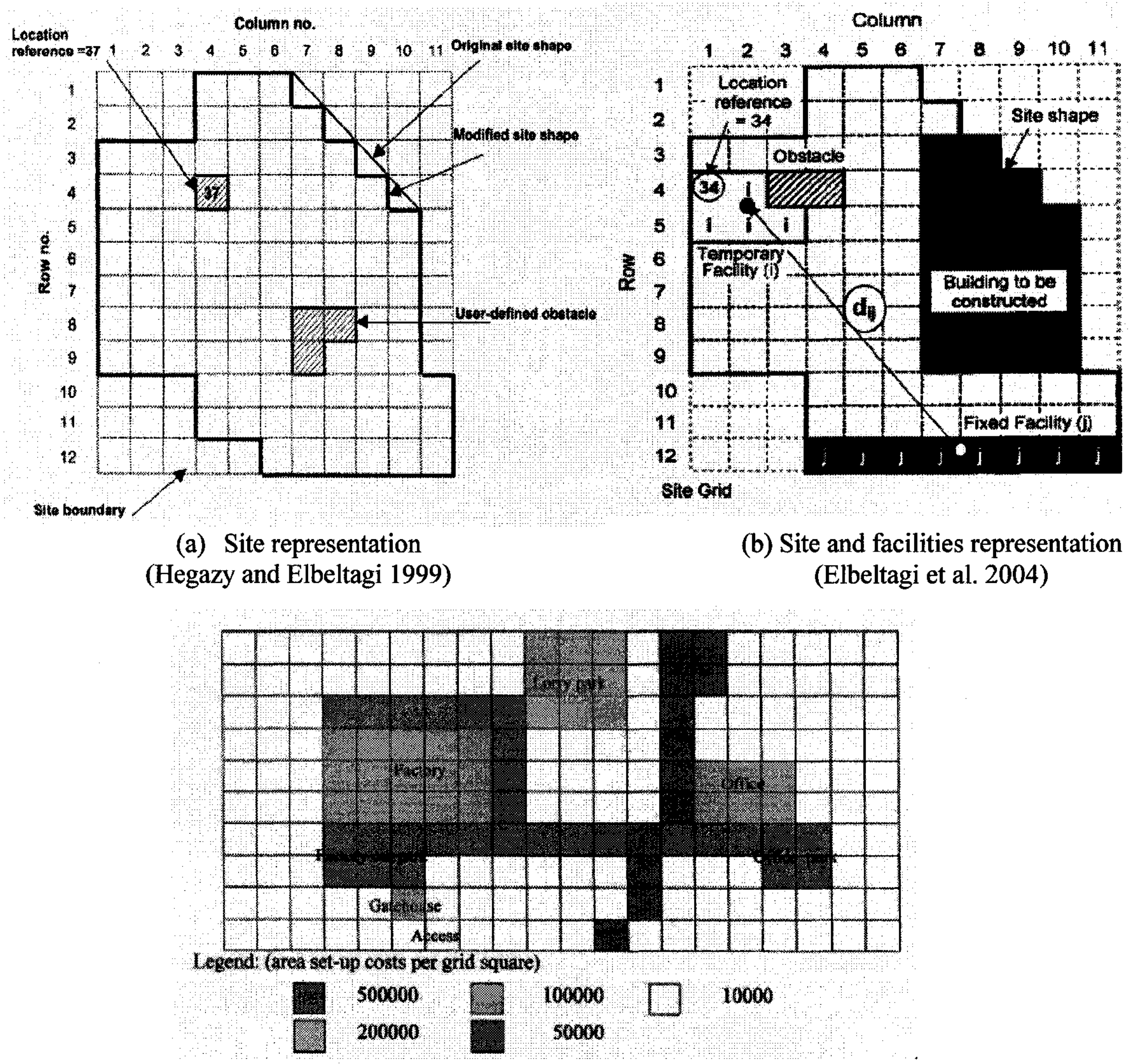
Site space is a resource that is as important as money, time, materials, labor, and equipment. Despite its importance, site planning is often neglected, and the attitude of engineers has been that it will be done as the project progresses. A proper site layout, however, can lead to (1) Reducing the cost of material handling; (2) Minimizing travel times of labor, materials, and equipment on site; (3) Improving construction productivity; and (4) Promoting construction safety and quality (Tommelein et al. 1992; Anumba and Bishop 1997; Hegazy and Elbeltagi 1999; and EI-Rayes and Khalafallah 2005).

Site layout research recognizes that space is needed, for some length of time, to store materials, prefabricate assemblies of materials, and perform work (Riley and Sanvido 1995, 1997). Numerous workers, equipment, material, temporary facilities, as well as permanent structures share the limited space during construction, which may affect productivity and the critical path (Guo 2002). Site layout planning is therefore typically developed at the beginning of a project as a tool to allocate construction resources, organize the available space efficiently and minimize space conflicts.

Considerable studies have been conducted in order to improve site layout planning in construction projects. These studies adopted a wide range of methodologies and development tools. The following are some examples: (1) Yeh (1995) used annealed neural networks to arrange a set of predetermined facilities on a set of predetermined locations on construction sites. (2) Several expert systems and knowledge-based systems were also developed to integrate domain knowledge of experts and assist in facility layout planning tasks (Kumara et al. 1988; Hamiani 1989). (3) Zouein and Tommelein (1999) proposed heuristic algorithms including the use of the early commitment criterion to design site layouts. (4) Genetic algorithms were also used in several studies to optimize the layouts of construction sites (Tam 2001; Li and Love 1998; Islier 1998; Elbeltagi and Hegazy 2004; Harmanani et al. 2000; and El-Rayes and Khalafallah 2005). (5) Cheng and Yang (2001) proposed an automated site layout system for construction materials, integrating with a GIS-based cost estimation system.

Simulation has been utilized to decide the layout planning of seaports (Wadhwa, 2000) and warehouse layout planning (Caron 2000; Randhawa and Sharoff 1995). Marasini and Dawood (2002) used simulation to develop a discrete event simulation model to optimize stockyard layouts for pre-cast concrete products and evaluate different layout scenarios. Tawfik and Fernando (2001) integrated simulation with genetic algorithms for the evaluation and optimization of construction site layouts.

Due to the vast number of trades and interrelated planning constraints in site layout planning, no matter what methodology is used, in order to analyze site layouts, a model should be built and space has to be taken into consideration. Space can be represented implicitly or explicitly. Implicit representation describes space at the conceptual level, which is abstract and difficult to understand. For better understanding of workspace, Riley and Sanvido (1995) proposed repeatable patterns that describe how typical activities use space over time. Thabet and Beliveau (1994) suggested explicitly accounting for workspace as a constraint in the scheduling process to improve safety, decrease conflicts among workers, and reduce project delays. However, because of the complex characteristics (e.g., size, shape, dynamic changes and other uncertainties) of space, it might be very difficult to describe space explicitly. When applying the explicit representation of space, a basic question is how to represent a space and how to position facilities. In order to analyze site layouts, some researchers developed explicit representation of space. For eample, Hegazy and Elbeltagi (1999) proposed a model for



(a) Site representation (Hegazy and Elbeltagi 1999)

(b) Site and facilities representation (Elbeltagi et al. 2004)

(c) Site layout and setup cost distribution for factory project (Mawdesley et al. 2002)

Figure 2.5 Examples of Explicit Representation of Space

explicit site layout planning (Figure 2.5 (a)). This model depicts any user-defined site space with a two-dimensional grid. Each facility is represented as a number of small grid units and can take irregular shapes. All facilities are represented as multiples of these units. To define the position of any facility on the site, a location reference is formulated and assigned to it (Figure 2.5 (a)) or to its top left position (Figure 2.5 (b)). Besides, non-orthogonal boundaries can be modeled using stepwise approximation. Hegazy and Elbeltagi (1999) utilized this model to visually place facilities within a construction site and then Elbeltagi et al. (2004) improved this model to analyze site layout incorporating heath, safety and productivity. Another application is presented by Mawdesley et al. (2002). In their model, the positions and areas of all the facilities are shown in the form of grid units (Figure 2.5 (c)). The advantages of explicit representation of space are: (1) Space can be graphically described so that it is easy to understand; (2) Space can be irregular or user-defined shapes; and (3) It increases the communication between model developers and users.

The literature in site layout planning provides valuable inspirations: (1) Simulation has been applied to site layout planning; (2) Using cells to explicitly represent space is feasible and effective, and (3) Explicit representation of space bridges the gap between site layout planning and construction simulation.

2.7 PROBLEM STATEMENT

As introduced in the above sections, a number of construction simulation tools have been developed to study real construction operations. These tools are mainly based on simulation networks. Although the use of simulation networks opened the door for construction simulation research, it did not appropriately consider an important issue in construction operations – space. Space, as an independent factor in construction operations, plays an important role in the outcome of construction, so construction planners and designers get a more realistic feedback from the simulation analysis when the space is accounted for (Oloufa 1992). Simulation networks may provide the logical relationships between the different resources; however, they are not suited to define the spatial relationships between resources on a specific construction sites. In other words, they failed to include the geometry and shape of construction sites, which are undoubtedly important factors influencing any construction operation (Oloufa 1992; Naji 1997). In construction, spatial conflicts always have a negative impact not only on the productivity, but also on health and safety. However, in simulation networks, space is either neglected or represented implicitly using abstract symbols. This could result in spatial constraints being ignored in the simulation, and the simulation result may not reflect the real situation of the construction site. Many researchers suggested that construction simulation should support the geometry and dynamics of construction resources. Halpin and Riggs (1992) mentioned that location or space-type flow units usually constrain the access to certain work processed and thus constrain the movement of other type units. Accordingly, spatial problems need to be studied in a way that is easy to understand, and the model should be built in a way that the space can be represented explicitly. Moreover, explicit representation of space in simulation will make the model closer to the real construction environment and thus contribute to a more realistic simulation result.

Incorporating explicit representation of space is not only necessary for construction simulation, but also necessary for animation and site layout planning. Construction animation, as a helpful verifying and validating means, illustrates every construction detail including detecting and resolving spatial conflicts. VITASCOPE, for example, provides a dynamic interference detection tool, C-Collide, which allows engineers to identify any undesired conflicts that can occur among construction resources. When the C-Collide tool detects collisions, it gives the user two response modes – interactive and silent. In interactive mode, C-Collide pauses a running animation and outputs details about each detected interferences. It will then waits for the user to specify whether (1) The interference detected should be ignored, (2) The animation should be aborted, or (3) The scene object pair should be deactivated from further computations. In silent mode, C-Collide does not interrupt a running animation even if interferences are detected. Instead, details about all collisions that occur during an animation run are time stamped and written to a formatted disk file for later analysis. There is a dilemma on this point. First, the interferences detected will be skipped without any solution in silent mode. Second, if the user does want to resolve the spatial conflicts, the only way is to abort animation, go back to simulation and modify the simulation results such as the travel path of a truck. If this modification changes the conditions of the subsequent work tasks (e.g. start time or duration), the simulation must be rerun. Supposing a construction is performed within a space-constrained site and spatial conflicts occur frequently, the user might be trapped into the loop of animation – modification – simulation – animation.

Finally, the studies of site layout planning and construction simulation have been done separately. Site layout planning focused on how to arrange resources on site, without considering specific construction operations; construction simulation emphasized how construction operations are executed, regardless of site layouts. Actually, site layout planning and construction operations are related continuous processes that should be considered together as an in-situ project manager would do to reach the project goal. The present separation might be because of the lack of explicit representation of space in construction simulation. Adequate space representation can bridge the gap between site layout planning and construction simulation. Recently, some methods (Mawdesley et al. 2002; Elbeltagi et al. 2004), as discussed in Section 2.6, introduced an explicit representation of space by dividing the site space into small rectangular grids of cells. However, their models are not sufficient enough for complex analysis when the position or state of facilities may change from time to time. More attributes, such as the type, moving state, and ID of a specific entity should be added to identify and reflect its dynamic changes.

2.8 CELL-BASED DISCRETE EVENT SIMULATION

2.8.1 Cellular Automata

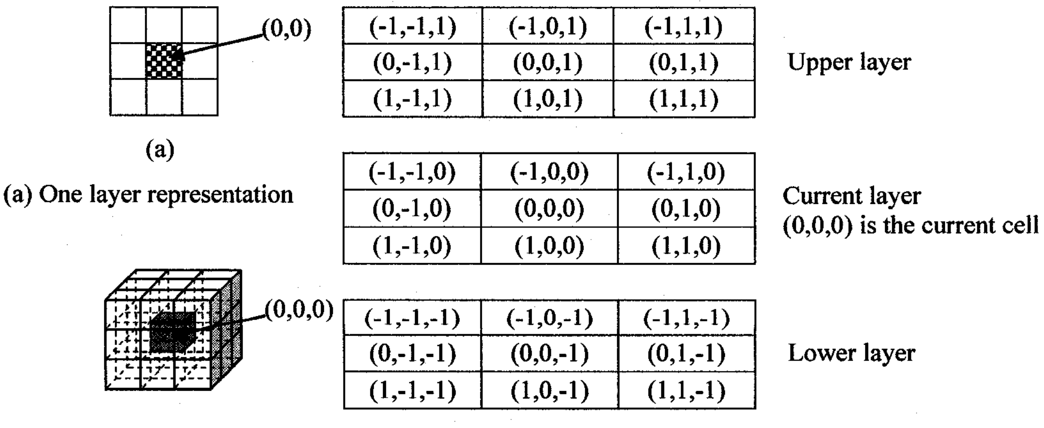
As mentioned in the previous sections, to explicitly represent a construction space, a possible solution is to divide the area into cells. Cellular Automata (CA), developed John Von Neumann and Stephan Ulam in 1940's, is a well-known modeling formalism which holds this potential. A CA is organized as an n-dimensional infinite lattice of elements, each holding a state value and a computing apparatus. The state of each cell is changed by a local computing function, which uses the present value for the cell and a finite set of neighboring cells to compute the new state. CA is suited to define spatial systems and allow the description of cell-based models by using simple rules (Wolfram 1986). CA has attracted the attention of researchers in geography, ecology, and other environmental sciences because of their ability to model and visualize spatially distributed processes (Ahmad and Simonovic 2004).

Nevertheless, CA has problems that constrain its power, i.e., the usability and feasibility to analyze complex systems: (1) CA is restricted by the simplicity of their formal description: neighborhood, uniformity of the cells, one discrete state per cell, closure to external events, and infinite lattices. According to experimental conditions, cell behavior and neighborhoods often need to be different. One state per cell is also usually not sufficient when dealing with complex deterministic systems. For example, Ahmad and Simonovic (2004) found that in deterministic cellular automata, transition rules do not change during the course of a simulation; (2) It may be necessary to have external events changing individual cells; and (3) CA uses a discrete time base for cell updates, constraining the precision and efficiency of the simulated models. Therefore, pure CA cannot be used for many applications, and they often need to be modified for simulation purposes (Wainer 2006; Muzy et al. 2005).

2.8.2 Discrete-Event System Specification (DEVS)

Zeigler (1976) defined a theory for Discrete-EVent system Specification (DEVS). It is a formal approach to build models using a hierarchical and modular method. This approach allows the developer to build a Model Base, permitting easy reuse of models that have been validated. A real system modeled using DEVS can be described as a set of behavioral (atomic) and structural (coupled) sub-models. Each model uses input/output ports to communicate with other models. The input external events are received in input ports, and the model specification defines the behavior under such inputs. The internal events produce state changes, whose results are spread through the output ports. The models can be hierarchically integrated, allowing model reuse. This improves the security of the simulations, reduces testing time and enhances productivity.

Using DEVS formalism, Zeigler (1976) defined a cell space model, which consists of an infinite set of geometrically defined cells, each cell containing the same computational apparatus as all other cells and connected to other cells in a uniform way. As shown in Figure 2.6 (a), for a specified cell located at the origin (0,0), the nearest neighbors would be those located at: (0,1) (1,0) (0,-1) (-1,0), which are at a distance of one cell away orthogonally and (1,1) (-1,1) (-1,-1) (1,-1) which are at a distance of one cell away



(-1,-1,1)	(-1,0,1)	(-1,1,1)	
(0,-1,1)	(0,0,1)	(0,1,1)	Upper layer
(1,-1,1)	(1,0,1)	(1,1,1)	

(-1,-1,0)	(-1,0,0)	(-1,1,0)	
(0,-1,0)	(0,0,0)	(0,1,0)	Current layer (0,0,0) is the current cell
(1,-1,0)	(1,0,0)	(1,1,0)	

(-1,-1,-1)	(-1,0,-1)	(-1,1,-1)	
(0,-1,-1)	(0,0,-1)	(0,1,-1)	Lower layer
(1,-1,-1)	(1,0,-1)	(1,1,-1)	

(a)

(a) One layer representation

(b)

(b) Three layers representation

(c)

(c) Neighborhood cells in three layers

Figure 2.6 Cell Representations

diagonally. The cell model could be two dimensional or three dimensional. Figures 2.6 (b) and (c) show the generalization of the cell neighborhood representation in three dimensions.

Recently, DEVS has been used in construction simulation to analyze the work flow between various trade contractors in production home building (Palaniappan et al. 2006). A shortcoming of DEVS is that it requires the user to have expertise in advanced programming techniques, which makes it difficult for the user to explore their models at first hand.

2.8.3 Cell-DEVS

Cell-DEVS stems from the following two concepts: (1) How to use the ability of CA to describe very complex systems using very simple rules (which is its main advantage), and (2) How to bridge the gap between a continuous variable formalism and a discrete event description such as DEVS, while allowing users to focus on the application itself (Giambiasi and Wainer 2005). Wainer (1998) developed an approach which was defined as an extension to CA combined with DEVS formalism, called Cell–Discrete Event System Specification (Cell-DEVS). The proposal of the Cell-DEVS paradigm considers each cell of a CA as a hierarchical and modular discrete events model (Wainer and Giambiasi 2001, 2002). In this way, complex models can be defined using a continuous time base. On one hand, this approach is based on the formal specifications of DEVS. Cell-DEVS defines a cell as a DEVS atomic model and a cell space as a coupled model. Each cell of a Cell-DEVS model holds a set of states for the neighborhood and a behavior or internal delay for the updates. A cell's output ports are linked to the cell's neighbor's input ports. The use of DEVS as the basic formal specification mechanism enables to define interactions with models defined in other formalism. On the other hand, users can use simple rules (as with CA) for modeling, allowing the definition of complex models easily. Considerable applications of Cell-DEVS have been developed in biology and medicine (Shang and Wainer 2005), environmental system (Wainer 2006), traffic (Tartaro et al. 2001; Davidson and Wainer 2006), and construction simulation (Zhang et al. 2007; Pang et al. 2006, 2007).

CD++ (Wainer 2002) is a modeling and simulation tool which implements DEVS and Cell-DEVS theories based on an abstract simulator mechanism. CD++ has been widely used in various applications from simple queuing systems to complex urban traffic systems or physical systems. Different versions of CD++ have been developed to facilitate various applications:

- Stand alone CD++: implements DEVS and Cell-DEVS simulation in a stand alone mode (Wainer 2002).
- Parallel CD++: is aiming to enhance the performance of Cell-DEVS simulation by distributing calculation of different cells over multiple processors (Liu and Wainer 2007).
- Distributed CD++: is developed to facilitate the coordination of the different simulating engines in different sites through the standard distributed computing protocols (Madhoun and Wainer 2007).

- Real time embedded CD++: is constructed especially for real-time embedded systems. A timing feature of the real-time systems has been included in CD++ to check the timing deadlines of given points of the systems, based on which the scheduability of the real-time system can be judged (Shang and Wainer 2006).

2.9 SUMMARY AND CONCLUSIONS

In this chapter, the literature about construction simulation was reviewed. By introducing the background of simulation, the necessity of using simulation in construction industry is discussed. The advantages and deficiencies of the currently-used construction simulation tools, site layout planning approaches as well as animation techniques were evaluated. Finally, the cell-based discrete event simulation technique (Cell-DEVS), which combines CA and DEVS formalisms and makes it well suited to explicitly represent space and easily reflect construction operations, is discussed. Based on the previous efforts of many researchers reviewed in this chapter, it is reasonable and achievable to develop an integrated simulation method for evaluating the performance of construction operations.

CHAPTER 3

CELL-BASED SIMULATION METHOD OF CONSTRUCTION OPERATIONS

3.1 INTRODUCTION

In Chapter 2, three representative construction simulation tools were reviewed. Theses tools have the following two limitations: (1) Work space is neglected or not explicitly represented; and (2) Spatial conflicts are not considered or collision detection is checked only after the simulation (i.e., using animation). Accordingly, finding possible solutions to spatial conflicts is not integrated within construction simulation. Spatial problems may cause conflicts between construction resources which usually have a negative influence on construction safety, quality and productivity. Space should be taken into account and spatial conflicts during the construction should be properly resolved, especially when spatial constraints are crucial to the project. Incorporating explicit representation of space will contribute to a more realistic simulation and a better understanding of construction environment. By developing adequate simulation methods which explicitly represent space, the process of site layout planning could be associated with simulation and the analyzed site layouts could be further used to predict the performance of construction operations considering various alternative site layouts. The alternative site layout patterns could be compared from the perspectives of productivity and spatial conflicts. In addition, a conflict-free and uninterrupted animation may be achieved. Therefore, space is the bridge that links construction simulation with site layout planning and animation to allocate construction resources, mitigate conflicts and improve productivity. This fact leads to finding an approach which can explicitly represent space, detect and resolve spatial conflicts, as well as integrate site layout planning, simulation and animation.

The research presented in this thesis proposes a new cell-based construction simulation method using the Cell-DEVS simulation methodology, which allows considering site spatial constraints and the sensitivity analysis with different site layout patterns. Using the cell-based simulation method, spatial resource allocation, site layouts, as well as the movement of equipment can be explicitly represented, analyzed and visualized. Spatial conflicts can be detected, resolved, and the simulation results can be compared based on different site layout patterns and resource combinations. As a result, the simulation accuracy is improved.

3.2 FEATURES OF THE CELL-DEVS SIMULATION METHODOLOGY

The Cell-DEVS methodology provides practitioners with a number of powerful features for modeling and simulation which can be effectively applies to construction simulation:

(1) Cell-DEVS inherits the advantages of CA and DEVS (discussed in Section 2.8.3). It explicitly represents the construction site by distributing the space into cells and allows the practitioners to control the states of cells using a set of simple rules.

(2) Because Cell-DEVS defines the cell space as discrete-event models based on the formal specifications of DEVS, it supports hierarchical simulation model

development and conforms to the discrete nature of construction operations. This enables the modeler to build and test complex models in an incremental fashion.

(3) Cell-DEVS is a general purpose simulation tool. In construction simulation, its application is not limited to any process or resource based simulation. Users have the flexibility to meet the specific requirements of a problem by customizing the simulation models.

(4) Cell-DEVS supports parallel and distributed simulation of models, allowing development and deployment of large-scale models (Liu and Wainer 2007; Madhoun and Wainer 2007).

(5) The Cell-DEVS implementation environment incorporates 2D animation tool which can visualize the simulation results automatically. In addition, a 3D animation tool, DEVSView (Khan et al. 2005), has been developed based on Maya software. These tools help the model developer to check and debug simulation models.

3.3 PROCESS OF CELL-BASED SIMULATION

The present research is to propose a cell-based simulation method that can be used for construction applications, integrating site layout planning, simulation and animation. In order to apply this method to construction operations, a special procedure is needed to instruct the users to develop cell-based simulation models; to consider different site layout patterns in simulation; to define a set of rules that can detect and resolve spatial conflicts among any objects and to analyze the simulation results considering the effects of spatial conflicts.

The process of the cell-based simulation is shown in Figure 3.1. It consists of three phases: Pre-processing, Main-processing and Post-processing. Pre-processing focuses on model building. This phase instructs how to build a cell-based simulation model through eight steps; Main-processing focuses on deciding site layout patterns and resource combinations, resource initialization, and running the simulation system. Post-processing includes analyzing, verifying and validating, as well as animating the simulation results. It emphasizes the analysis of simulation results.

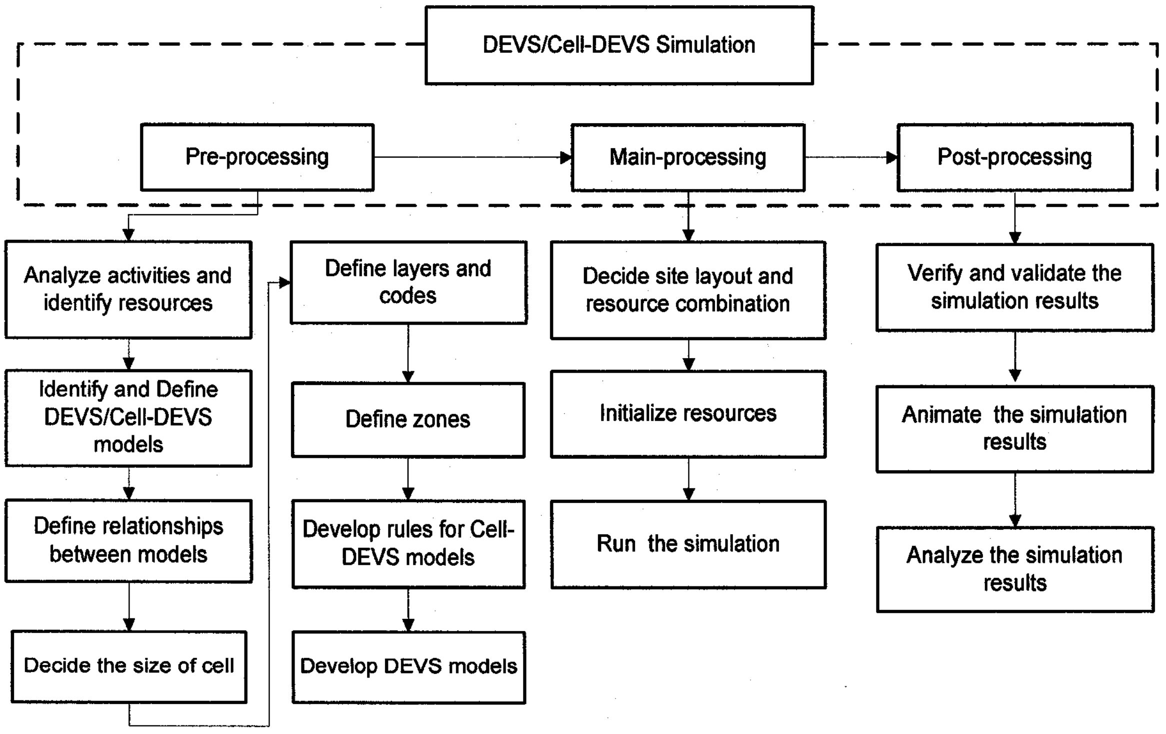

DEVS/Cell-DEVS Simulation
Pre-processing
Main-processing
Post-processing
Analyze activities and identify resources
Identify and Define DEVS/Cell-DEVS models
Define relationships between models
Decide the size of cell
Define layers and codes
Define zones
Develop rules for Cell-DEVS models
Develop DEVS models
Decide site layout and resource combination
Initialize resources
Run the simulation
Verify and validate the simulation results
Animate the simulation results
Analyze the simulation results

Figure 3.1 Process of Cell-based Simulation

3.4 PRE-PROCESSING

3.4.1 Analyzing Construction Activities and Identifying Involved Resources

As the first step, the development of a construction simulation model requires breaking the construction operations down to recognizable work tasks and identifying the construction resources involved in the operations.

3.4.2 Identifying and Defining DEVS and Cell-DEVS Models

The second step is to define models to represent construction operations. When defining a model, the type of this model should be identified. Two types of models are used in cell-based simulation – Cell-DEVS and DEVS models. Since Cell-DEVS models provides virtual cell space to represent the real worksite, they are employed where the spatial representation is emphasized; while DEVS models are employed to simulate processes that cannot be represented by cells or when spatial representation is not necessary. For

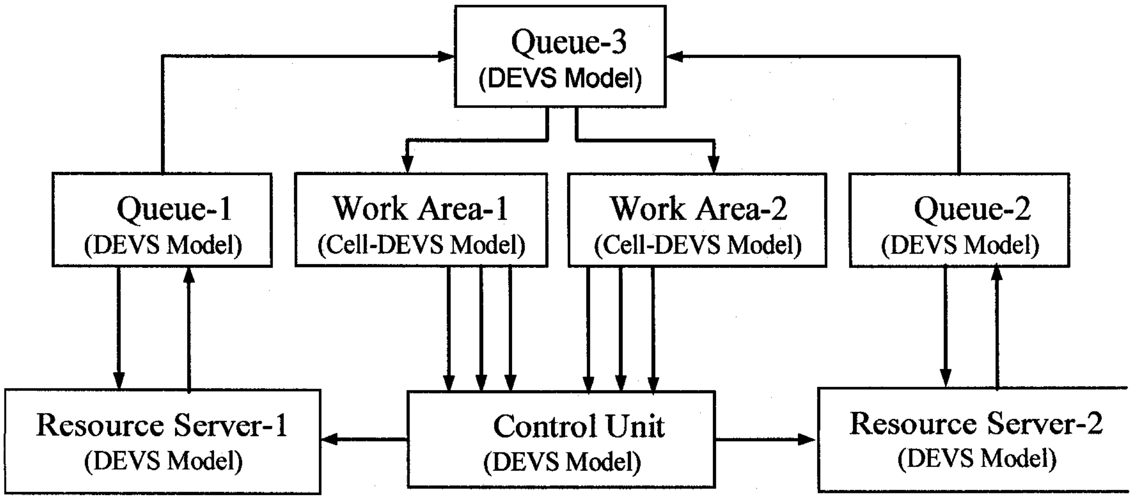


Figure 3.2 Abstract Example of a Cell-based Simulation

example, suppose that a construction is performed on *Work Area-1* and *Work Area-2* as shown in Figure 3.2, the spatial representation of the *Work Areas* is important because most of spatial conflicts occur on them and every detail of construction work spaces should be visualized. As a result, these *Work Areas* should be defined as a Cell-DEVS model. *Control Unit* and *Queues* can only be defined as DEVS models. *Resource Servers* can be defined as either of the two types. If spatial problems of these models are important for the final results, they should be defined as Cell-DEVS models; otherwise, they should be defined as DEVS models. Each model might be composed of several sub-models, forming different hierarchical levels.

3.4.3 Defining Relationships between Models

Models can communicate with each other through input/output ports. Therefore, it is necessary to define correct input/output ports and build up links between them to pass messages. In cell-based simulation, links between models show the directions in which messages are sent. For example, in Figure 3.2, suppose that the each *Work Area* have three lanes and the mobile objects (e.g., vehicles) could enter through one of the three lanes. If it is necessary to know where (on which lane) a vehicle accesses, the three lanes must be monitored. Accordingly, three links should be defined to send and receive messages that indicate the vehicle's access. Figure 3.2 shows the three links between the *Work Area*s and *Control Unit*.

3.4.4 Deciding the Suitable Size of Cells and the Dimension of Each Cell-DEVS Model

Different levels of details can be used to decide the size of a cell considering the following factors:

(1) The size of cell depends on the size of the smallest object to be represented. At this stage, because the cell-based simulation method can only represent objects with equally square, triangle and hexagonal cells, it is difficult to represent an object with specific shapes and dimensions, such as a truck of 12m-length and 3m-width. For a more accurate representation, a combination of several cells (called "block") could be applied to represent the space occupied by an object. For example, in Figure 3.3, a truck can be represented by four cells of 3m x 3m size. The telescopic cranes and the old sections to be removed are represented by three cells;

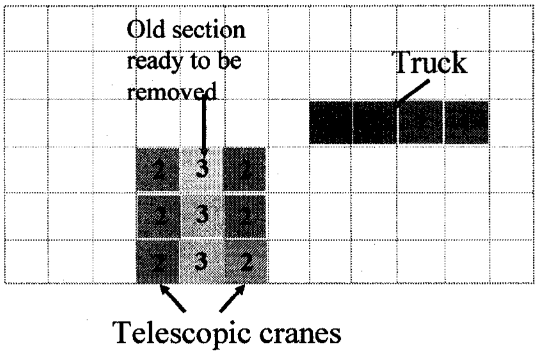
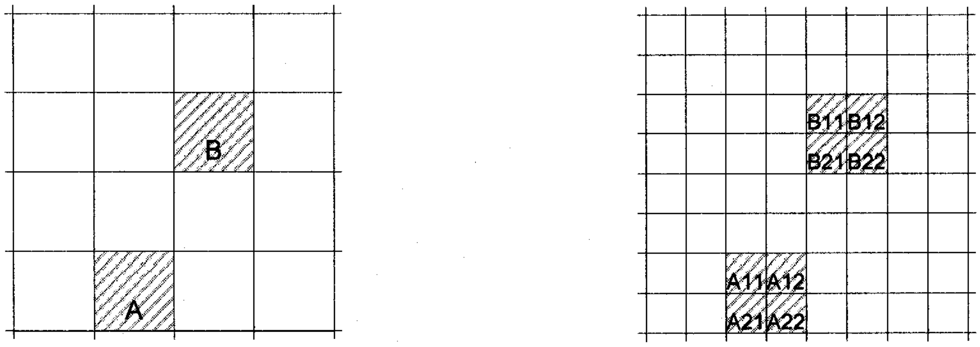


Figure 3.3 Cell Representations of Objects

(2) If the cell is small enough to represent the shape of the objects, the accuracy can be improved. However, the smaller the size of cells, the more complex are the rules in order to cover the extended neighborhood. Figure 3.4 shows an informal

comparison of the complexity with different sizes of cell. The complexity could be seen from two points of view. From the point of the object itself, such as objects A and B in Figure 3.4, an object is represented by four cells in Figure 3.4(b) rather than one cell in Figure 3.4(a), so when the object is moving, additional rules are needed to control more cells' states in Figure 3.4(b) than in Figure 3.4(a); on the other hand, from the point view of other objects, the situation is also more complex. For example, when the object A is moving, it detects its neighbor cells to make sure there are no obstacles. In Figure 3.4(a), the object A only detects one cell to identify the object B, while in Figure 3.4(b) it has to detect four cells;



(a) Big Size of Cell

(b) Small Size of Cell

Figure 3.4 Using Different Size of Cells to Represent Objects

(3) The size of a cell determines how long an object passes through a cell giving that the speed of the mobile object is constant. For instance, if a truck moves at the speed of 10km/h and the size of cell is 3m x 3m, it takes about one second for the truck to go from one cell to the next.

Considering these factors, the user could define suitable size of cells to represent objects and then the dimension of the cellular space is determined.

3.4.5 Defining the Layers and Codes for Each Cell-DEVS Model

A layer carries the attribute of cells. Each layer describes one aspect of these attributes. The state of a cell is the collection of these attributes. The cell space can be taken as an N-dimensional space and its attributes are distributed on the N layers. For example, in a general construction site model, it may be necessary to indicate if a cell is occupied by an object, its type (e.g., truck, crane or other types of equipment), the moving behavior of that object (e.g., static, moving and the moving direction), and its ID. Therefore, three layers could be defined to describe the three attributes as shown in Figure 3.5. Predefined

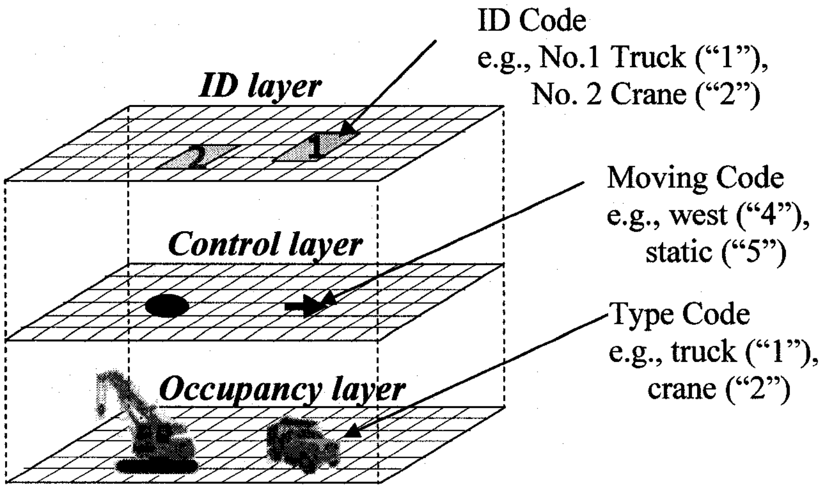


Figure 3.5 Using Layers and Codes to Indicate the Attributes of Cells

codes are assigned to the cells of each layer to indicate these attributes. Consequently, the attributes of a cell can be represented as a vector <x, y, z> (indicating the type of equipment, the moving state and the ID code). Collecting the attributes on each layer, the attributes of a specific cell can be defined and then any object within the cell space can be represented by one or more these cells. The number of layers (attributes) may be more or less depending on the requirements. For a more complex system, more attributes, such as the weight and height of equipment, can be integrated.

3.4.6 Defining the Zones of Each Cell-DEVS Model

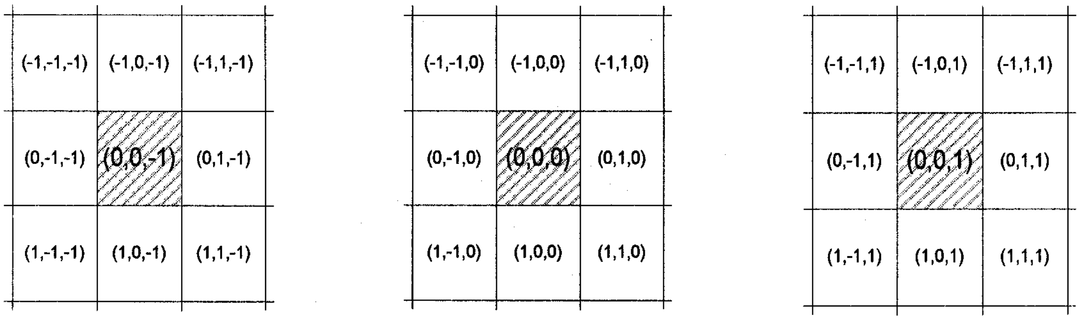
A zone defines a region of the cellular space that will use a different local transition function. By default, rules of a Cell-DEVS model apply to all the cells of the model. However, in some cases, it is necessary to specify a part of cells as a zone where different local rules apply. For example, in a road-paving project, one lane is closed to perform the construction and the other lanes can be still used for traffic. Zones reserved for the pavement and for the traffic could be defined to apply different sets of rules to simulate the paving operation and traffic flow within one cellular space.

3.4.7 Developing Rules for Each Cell-DEVS Model

As mentioned in 3.4.5, the behavior of any object within the cell space is represented by the attributes of a specific cell. From this point of view, cell-based simulation can be used not only to represent a space with cells, but also to control the state of a cell using rules. This feature allows detecting and resolving spatial conflicts. Rules are applied to the cells of each layer of the cell-based models to detect and define the states of cells through communication among them. In this way, spatial conflicts can be detected and resolved.

(1) How rules work

In a Cell-DEVS model, rules detect the states of the reference cell's neighbors by the communication between cells and defining the change of the reference cell's state. Each rule is composed of three elements: a *condition*, a *delay* and a *result* with the format {Result} Delay {Condition}. The simulator takes each rule in the order that they were defined and evaluates the {Condition} clause. If the {Condition} is satisfied, its {Result} and its Delay are evaluated, and the {Result} will be assigned to the cell after the Delay. This change of state will spread to the cell's neighbors through the interface between cells. Cells are related by relative coordinates and any cell that satisfies the {Condition} clause in the current layer is the reference cell. For example, when defining a rule on the current layer (*Control* layer) shown in Figure 3.6, the location of the reference cell is (0, 0, 0). Using this reference cell, the location of any neighbor cell can be defined as



(-1,-1,-1)	(-1,0,-1)	(-1,1,-1)
(0,-1,-1)	▨ (0,0,-1)	(0,1,-1)
(1,-1,-1)	(1,0,-1)	(1,1,-1)

Lower layer
(*Occupancy* layer)

(-1,-1,0)	(-1,0,0)	(-1,1,0)
(0,-1,0)	▨ (0,0,0)	(0,1,0)
(1,-1,0)	(1,0,0)	(1,1,0)

Current layer (*Control* layer)
(0,0,0) is the reference cell

(-1,-1,1)	(-1,0,1)	(-1,1,1)
(0,-1,1)	▨ (0,0,1)	(0,1,1)
(1,-1,1)	(1,0,1)	(1,1,1)

Upper layer
(*ID* layer)

Figure 3.6 Neighbor Cells Representation on the Three Layers

illustrated in Figure 3.6. If the evaluation of the condition of the rule is false, the simulator will take the following rule. If all the rules are evaluated without having some valid one, then the simulation will be aborted. If there is more than one valid rule, the simulator takes the first of them.

(2) Using rules to control the movement of mobile objects

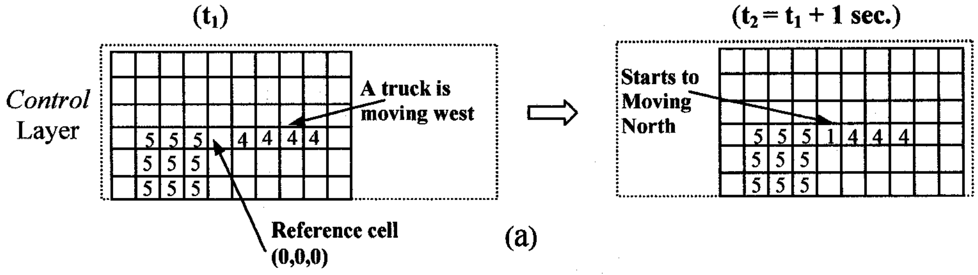
As mentioned in 3.4.5, predefined codes are employed to indicate the state of a cell. For example, if the code "5" indicates a "static" object which is keeping static during the task duration, it can be regarded as a signal of an obstacle. When a moving object meets a static object with code "5", it needs to change its moving direction to get around the obstacle. As an example, taking the cell represnetation of objects shown in Figure 3.3 and the three-layer structure in Figure 3.5, the rules could be defined as the following: first, take the *Control* layer (which manipulates the moving direction of an object) as the current layer; second, define the {Result} Delay {Condition} clause on the *Control* layer that changes the truck's direction from "4" (west) to "1" (north); and third, define rules on the other two layers to follow this action correspondingly (Figure 3.7).

In other cases, spatial conflicts among moving objects may occur. For example, when an object is moving, other moving objects may become obstacles and the conflict among moving objects will occur. Some of them have to stop, giving the moving priority to the others. Therefore, defining a set of priority rules to detect and resolve these conflicts becomes necessary.

Rule on the *Control* layer

```
rule : 1 1 { (0,1,0) = 4 and (0,0,0) = 0 and (-1,0,0) = 0 and
(0,-1,0) = 5 and (1,-1,0) = 5 }
```

Note: if the reference cell detects an obstacle "5" (which means "static"), it changes its direction from "4" (west) to "1" (north) at the next step (one second later).

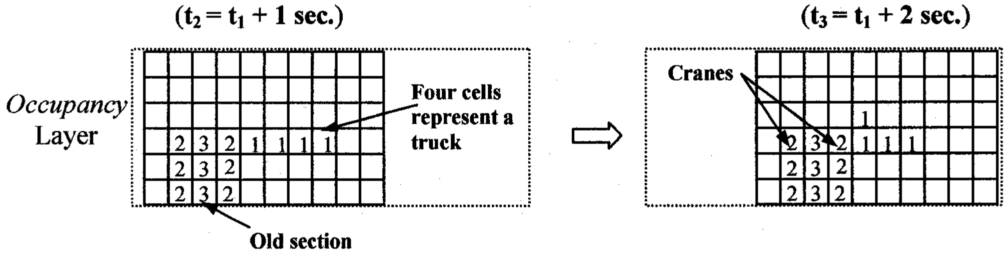


(a)

Rule on the *Occupancy* layer

```
rule : { (1,0,0)} 1 { (1,0,0) != 0 and (1,0,1) = 1 }
```

Note: if the reference cell's moving direction is "1" (north), its occupancy code moves north at the next step (one second later).

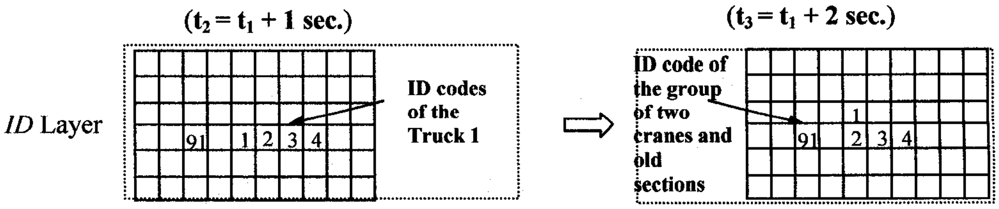


(b)

Rule on the *ID* layer

```
rule : {(1,0,0)} 1 {((1,0,0) > 0 and (1,0,0) < 91) and (1,0,-
1) = 1 }
```

Note: if moving direction of the reference cell is "1" (north), its ID moves north at the next step (one second later).



(c)

Figure 3.7 Examples of Rules and Cell State Changes on the Three Layers

Defining rules to detect and resolve conflicts among moving objects is one of the difficulties in stochastic simulaion. When defining rules, the user has to explore every probability because various conflict patterns or cases may happen. In addition, the reusability is one of the most important features in defining rules to control the movement such as moving north, south, west and stopping. Each action is performed by a set of rules and these rules can be grouped into a "macro" which can be called at request to perform a specific action, as a function in programming languages. Examples of the "macros" are shown in Appendix B.

3.4.8 Developing DEVS Models

DEVS models are important components in cell-based construction simulation. For a DEVS model, the external events are collected through input ports and the *external transition function* defines how to react to such inputs. At the moment the duration for the present state expires, the *internal transition function* is activated and causes the internal state change of the model. The desired results are spread through output ports by activating the *output function.* These functions are defined for each DEVS model and can be fulfilled by programming. The programmability makes these DEVS models very flexible. For example, the user can define the queues as "first in, first out" (FIFO) or redefined them as any other types to meet the construction requirements. Furthermore, these validated models can be standardized and reused for other projects with minor modification. An example of the DEVS model is shown in Appendix C.

3.5 MAIN-PROCESSING

3.5.1 Deciding Site Layout and Resource Combination

Different from other construction simulation tools, the resource allocation in cell-based simulation consists of two aspects: quantity and position. The user can decide not only the number of each resource, but also where to place them. Thus, the productivity of different site layout patterns can be examined and compared. The effects of spatial constraints can be further investigated. The key to investigate different site layouts is that a mechanism that connect the facilities and moving objects must be established, i.e., moving objects, such as trucks, can be sent to their desired positions. In Chapter 4, an example of designing the mechanism is explained.

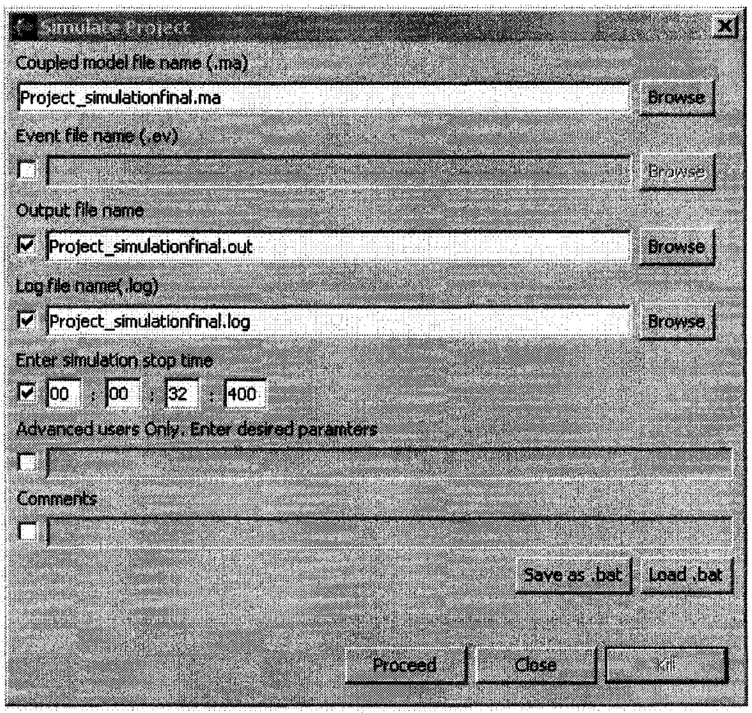
Various resource combinations should be exercised and observed in order to determine the system response and imbalances between resources. These imbalances always result in process bottlenecks and inefficiencies. The purpose of trying different resource combinations is to find these imbalances and then avoid or minimize them by a proper selection and balance of resources. In addition, this will help in optimizing resource combinations.

3.5.2 Initializing Resources

Before running the simulation, all the resources should be placed at their initial positions. For DEVS models, the resources can be directly defined in the proper models by programming. For Cell-DEVS models, resources can be defined using a text file with an extension name (.val). For Cell-DEVS models, the surface of the construction site is represented by a number of cells which have vertical and horizontal coordinates. The top-left corner is the origin on each layer. When initialize the resources, these absolute coordinates can be used. Examples of initializing resources are shown in Section 4.2.6.2.

3.5.3 Running the simulation

To run a cell-based simulation, a (.ma) file is necessary. This file is the model file which defines all the models and their relationships. In addition, it defines the system output ports. Any state changes of these output ports will be recorded in a (.out) file. Another output file that can be chosen is (.log) file which keeps a record of all messages sent between models. These two output files are very useful for verification and will be



Simulate Project

Coupled model file name (.ma)

Project_simulationfinal.ma

Event file name (.ev)

□

Output file name

☑ Project_simulationfinal.out

Log file name(.log)

☑ Project_simulationfinal.log

Enter simulation stop time

☑ 00 : 00 : 32 : 400

Advanced users Only. Enter desired paramters

□

Comments

□

Figure 3.8 Simulation Panel

discussed in Chapter 4. The time precision is millisecond ("MS" thousandth of second). The format used to set the simulation stop time is HH: MM: SS: MS as shown in Figure 3.8.

3.6 POST-PROCESSING

3.6.1 Verification and Validation

Verification and validation of simulation models is one of the most important tasks in simulation. The purpose of verification is to assure that the conceptual model is reflected accurately in the computer code. The goal of the validation process is to produce a model that represents the true system behavior closely enough and to increase the credibility of the model to an acceptable level (Banks 1984).

Cell-based simulation provides four output files for the user to verify and test the simulation results. The (.log) file and the (.out) file can be generated automatically when simulation advances. The (.drw) file can be automatically created after the simulation. Besides, a (.txt) file can be developed by the user. The process of verification is exercised through close examination of these output files. These files depict the simulation state from different perspectives and at different levels of detail. The (.log) file gives a panoramic description; the (.out) file concentrates on the values of the output ports; the (.drw) file focuses on the state of cell space, and the user-customized (.txt) file can output any variables the user is interested in. In addition, these files can also be utilized as useful tools for debugging errors in the practice of simulation. As an example, case studies in Chapter 4 illustrate how to use these output files to trace a specific object.

3.6.2 Visualizing the Cell-DEVS Models

In the case of simulated construction operations, animation is a smooth and continuous display of a sequence of images that represent construction processes visually. Animation transfers the numerical simulation results into visual images which can help the user to understand the information in a natural manner. For the proposed cell-based simulation method, it integrates an animation tool that can read the (.log) file or the (.drw) file introduced in Section 3.6.1 so that it can visualize the simulation states of Cell-DEVS models in 2D automatically.

In order to improve model validation and analysis, a new development in cell-based simulation incorporates a 3D visualization engine based on the Maya 3D visualization tool and its scripting language, which allows virtual worlds to be developed using the Maya visualization environment, and permits interaction with DEVS models built in CD++. This 3D animation tool enables sophisticated visualization of DEVS and Cell-DEVS models.

3.6.3 Analyzing Simulation Results

Due to the introduction of space, the simulation results can be analyzed from multiple perspectives according to the requirements. In addition to the conventional sensitivity analysis by changing one or more variables while keeping the others, the delays resulting from spatial conflicts under different site layout patterns can also be identified and quantified. Furthermore, in most of construction simulation tools, the speed of moving objects is assumed to be constant, while in the cell-based simulation, the speed can be changeable by manipulating the cells' states using different delays. For example, when a truck is changing its moving direction, it might slow down. As a result, more realistic simulation conditions might be achieved and the simulation results with changeable moving speed of mobile objects could be taken into consideration.

3.7 SUMMARY AND CONCLUSIONS

In this chapter, a cell-based simulation method was proposed for construction processes. This method includes three phases that cover site layout planning, simulation and animation so that it extends the field of construction simulation.

The process of developing cell-based simulation models discussed in this chapter guides the user to build a system for construction operations step by step. It should be noted that although the general procedure of consists of many steps, it is unnecessary to follow all of them. Dividing space into zones, for example, is not always applied.

CHAPTER 4

IMPLEMENTATION AND CASE STUDIES

4.1 INTRODUCTION

In Chapter 3, the methodology of cell-based simulation was proposed for construction operations. This chapter continues to implement this method using two case studies. The first one is the Jacques Cartier Bridge rehabilitation project, which is a cyclic simulation system. The second case study is trying to simulate the traffic flow under different lane-closure scenarios of an asphalt pavement project, which is a non-cyclic system. The two case studies follow the steps of developing cell-based simulation systems discussed in Chapter 3. These simulation systems can consider different site layouts, detect and resolve spatial conflicts during construction operations, and provide multiple perspectives to analyze the simulation results according to the requirements. The objectives of the two case studies are: (1) To show how to implement the cell-based simulation method and what this method can do; and (2) To set up a basic "Modeling Element Library" to store and retrieve modeling elements that can be easily reused in the future.

4.2 CASE STUDY I – JACQUES CARTIER BRIDGE REHABILITATION PROJECT

4.2.1 Background

The case study of the Jacques Cartier Bridge rehabilitation project is used to implement the cell-based simulation methodology. The Jacques Cartier Bridge, built in 1928, is one of the landmarks in Montreal. It is about 2.7 km in length with a main span of 600 m, spanning the St. Lawrence River between the cities of Montreal and Longueuil (Figure 4.1) (Zaki and Mailhot 2003).

With an increasing rate of 2.4% of traffic volume annually, this bridge carries some 43 million vehicles every year, making it one of the busiest bridges in North America when considering traffic volumes per lane (Zaki and Mailhot 2003). Over the last 70 years, the old reinforced concrete bridge deck had suffered seriously from the increase of the number and load of trucks and the de-icing salts used extensively since the 1960s. Consequently, the deck of this bridge was replaced in 2001-2002. Due to the high traffic

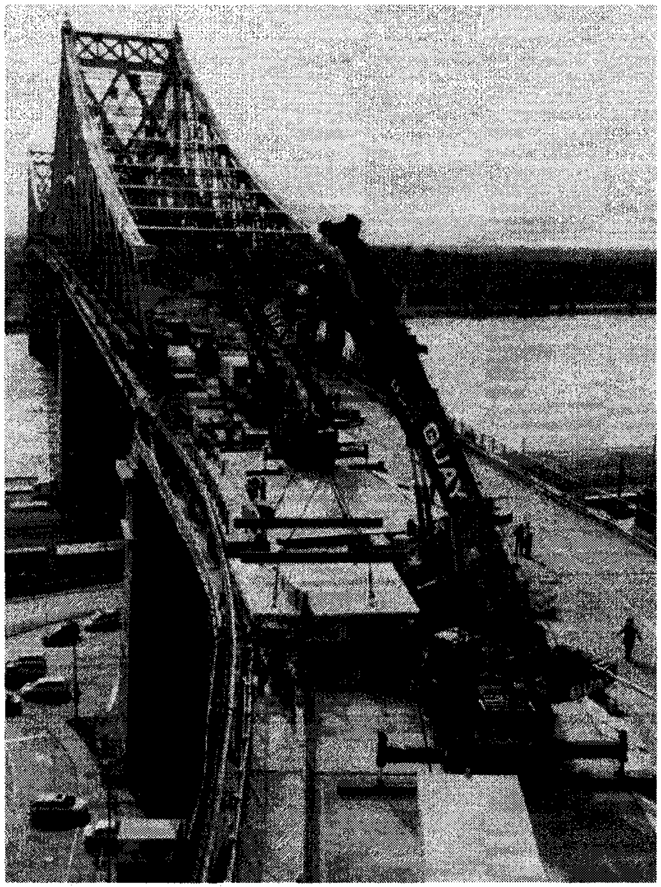


Figure 4.1 Jacques Cartier Bridge Rehabilitation Project

demand during the daytime, deck replacement had to be done during the night of weekdays from 8:30 p.m. to 5:30 a.m. the next day. The new deck is constructed of pre-cast, pre-stressed and post-tensioned panels made of high performance concrete which were prefabricated in a temporary plant near the south end of the bridge. The existing deck was removed by saw-cutting the deck into sections similar in dimensions to the new panels being installed. Each existing deck section was hoisted by two telescopic cranes and transported to the dump area by a truck.

Then, a new panel was delivered by a truck, lifted by the same cranes and lowered onto the new bearing assemblies. Several teams (each includes two cranes) worked in parallel at different positions on the bridge. Each team needed at least 60 m in length of the workspace, which means the maximum number of teams is 10 on the main span of the bridge (Figure 4.1). Because of the spatio-temporal constraints, this project needed a good plan to finish on time with high productivity.

The deck replacement involved six principal types of construction activities as described by Zaki and Mailhot (2003): (1) Steel work that included floor beam repair, strengthening work, and installation of new bearing assemblies to support the panels; (2) Pre-casting of new deck panels; (3) Removal of existing deck sections; (4) Installation of new panels; (5) Joint mortar placement and post-tensioning; and (6) Installation of expansion joint armors, cast-in-place expansion joint dams, and waterproofing and paving work. At peak production, the contractor was able to replace eight panels per crew per night. A total of 1680 precast deck panels were installed, including 410 panels for the main span.

4.2.2 Data Collection

The bridge data were acquired from the bridge management authority (Jacques Cartier and Champlain Bridges Incorporated) (PJCCI 2004; Zaki and Mailhot 2003). The data include AutoCAD drawings and deck rehabilitation schedules. The approximate durations for the tasks and the cost of equipment used in this research (such as the data in Table 4.1 and Table 4.2) came from the consultation with Mr. Raymond Coté and the construction project manager Mr. Germain Cardinal at SNC Lavalin. They provided valuable data about the Jacques Cartier Bridge rehabilitation project as shown in Figure 4.2.

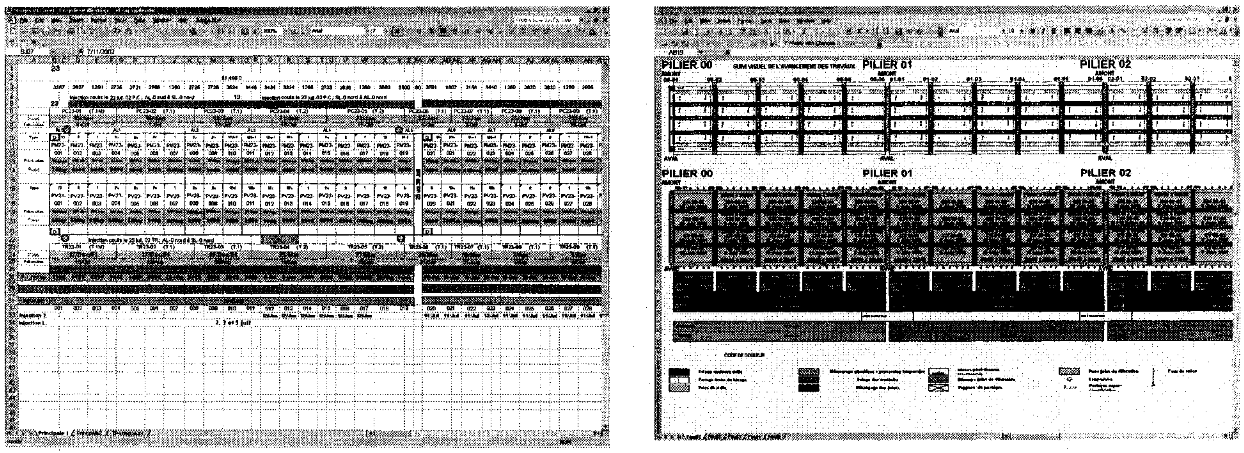


Figure 4.2 Data Used in Case Study I:
(a) Schedule data; (b) Advance of work

4.2.3 Previous Study – a MicroCYCLONE Model

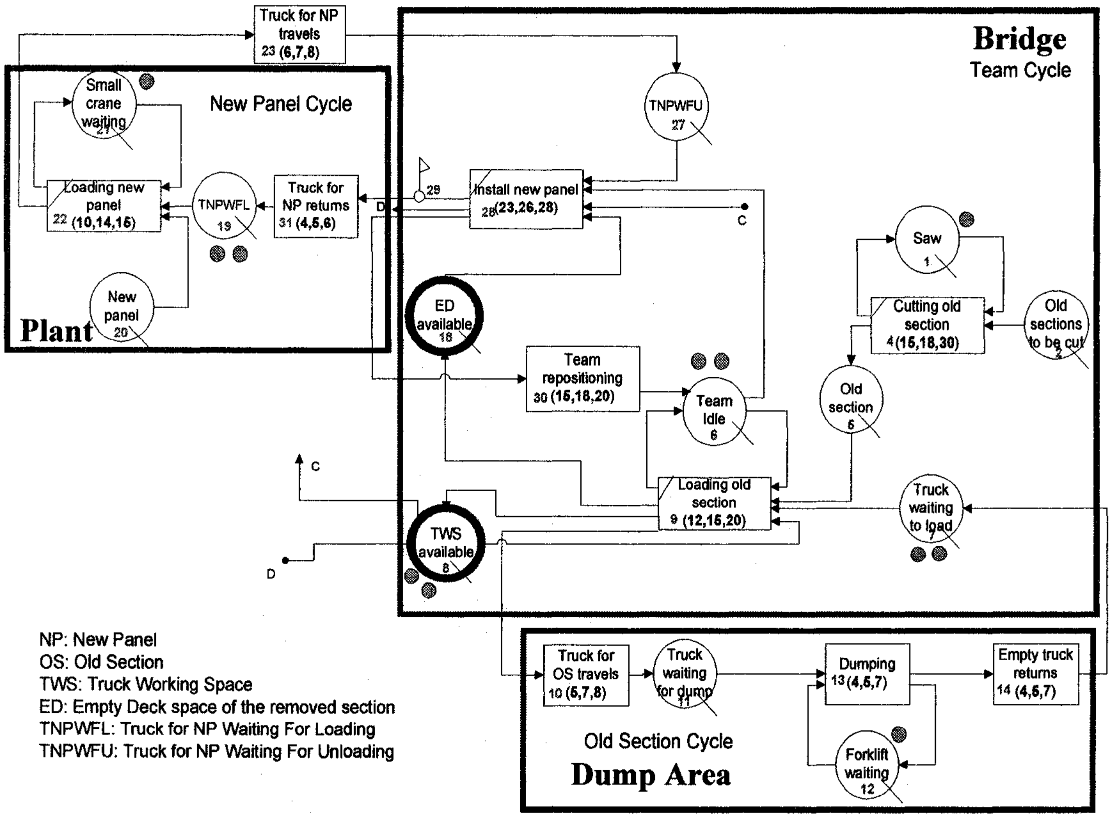

Truck for NP travels 23 (6,7,8)
Bridge
Team Cycle
Small crane waiting 21
New Panel Cycle
TNPWFU 27
Loading new panel 22 (10,14,15)
TNPWFL 19
Truck for NP returns 31 (4,5,6)
D
29
Install new panel 28(23,26,28)
C
Saw 1
New panel 20
Plant
ED available 18
Cutting old section 4(15,18,30)
Old sections to be cut 2
Team repositioning 30 (15,18,20)
Team Idle 6
Old section 5
C
Loading old section 9 (12,15,20)
Truck waiting to load 7
TWS available 8
D
NP: New Panel
OS: Old Section
TWS: Truck Working Space
ED: Empty Deck space of the removed section
TNPWFL: Truck for NP Waiting For Loading
TNPWFU: Truck for NP Waiting For Unloading
Truck for OS travels 10 (5,7,8)
Truck waiting for dump 11
Dumping 13(4,5,7)
Empty truck returns 14 (4,5,7)
Old Section Cycle
Dump Area
Forklift waiting 12

Figure 4.3 MicroCYCLONE Model of the Jacques Cartier Bridge Rehabilitation Project (Zhang et al. 2007)

The real project provides the research with valuable data. Nevertheless, these data cannot predict the performance of different resource combinations and site layout patterns. To further examine the predictive ability of this prototype system and increase confidence in it, a previous study – a MicroCYCLONE model for this case study – is used to compare with the proposed cell-based simulation model. The comparison of analysis results of the two simulation methods is discussed in Section 4.2.7.3.

Figure 4.3 presents the MicroCYCLONE model for this case study and Table 4.1 shows the involved tasks and their durations. As shown in Figure 4.3, active-state rectangular nodes represent tasks and include the triangular time distribution for each task. Idle-state circles represent delays or waiting positions for resource entities; and directional flow arrows represent the path of resource entities as they move between idle and active states. Tasks can be executed only when all the queues (resources) needed are available. For example, task "Dumping" (Task 13) is the task of dumping an old section of the deck in the dump area. It takes an average time of 5 minutes and needs two resources: "Truck waiting for dump" (Task 11) and "Forklift waiting" (Task 12). After the dumping of one section is finished, these two resources are released: the truck will return to the bridge

Table 4.1 Task Durations and IDs Used in MicroCYCLONE Model (Zhang et al. 2007)

Activity	Task ID	Task Description	Triangular Distribution of Durations (min.)
Remove old sections	4	Cut old sections	15, 18, 30
	9	Load old sections	12, 15, 20
	10	Truck with old sections travels to dumping area	5, 7, 8
	13	Dump old sections	4, 5, 7
	14	Empty OS truck returns to bridge	4, 5, 7
Install new panels	22	Load new panels	10, 14, 15
	23	Truck with new panels travels to bridge	6, 7, 8
	28	Install new panels	23, 26, 28
	31	NP truck returns to plant	4, 5, 6
	30	Team repositioning	15, 18, 20

(Task 14) and the forklift will go back to idle state (Task 12). The procedure for modeling these processes involves four basic steps: (1) Flow unit identification; (2) Development of flow unit cycles; (3) Integration of flow unit cycles; and (4) Flow unit initialization.

To investigate the space representation in this model, a conceptual mapping of the space is applied. In this model, spaces are represented as abstract symbols-queues: *Empty Deck (ED) Available* (18), and *Truck Working Space (TWS) Available* (8). The site layout can be conceptually divided into several areas according to the geographic locations, including *Bridge*, *Plant*, and *Dump Area*. There are two spaces implicitly represented as queues in this model: (1) Empty deck space of a removed section; and (2) Truck working space. Other spaces, such as the moving paths of trucks, are considered to be available all the time and are not represented in this system. The spatial conflicts can not be discovered in MicroCYCLONE.

4.2.4 Previous Study – a Cell-based Model

Zhang et al. (2007) proposed a cell-based model for the same project as shown in Figure 4.4. It was the first time that the cell-based method was used for construction simulation. The previous efforts showed the feasibility and advantages of cell-based simulation.

Nevertheless, since the previous cell-based model did not implement the three phases of cell-based simulation discussed in Chapter 3 as well as its simplified structure, it has limitations in terms of function and structure:

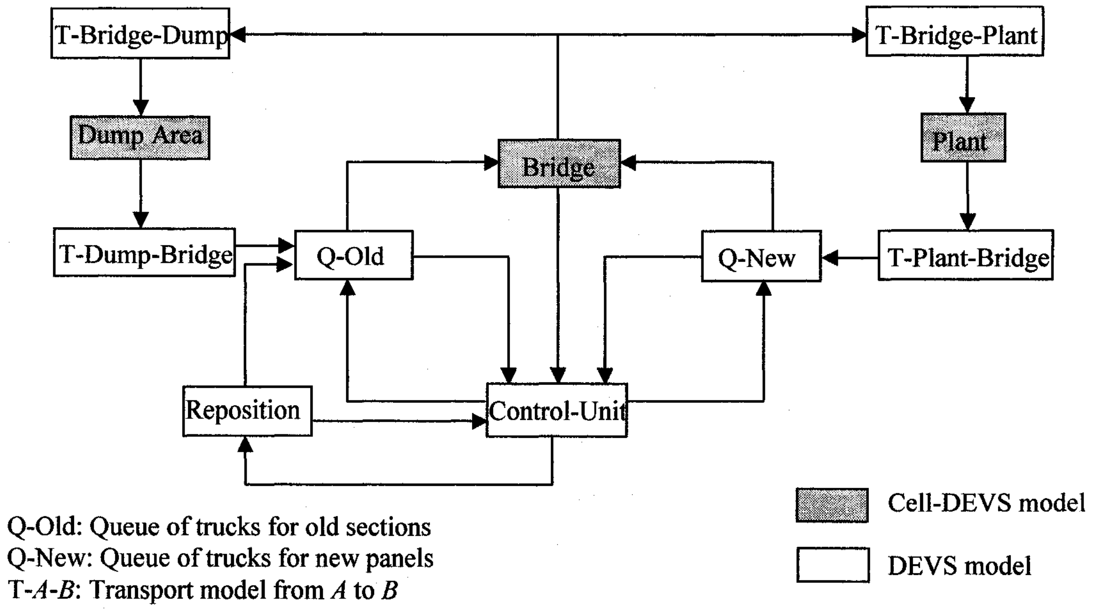


Figure 4.4 Relationships between Cell-based Models of the Jacques Cartier Bridge Rehabilitation Project (Zhang et al., 2007)

In function:

(1) The previous model did not consider different site layouts which may cause impracticable result. For example, the trucks will stop at the nearest available space for loading or unloading. Hence, after a span of simulation time, the team near the bridge entry may install much more panels than the team far away from the bridge entry. Consequently, the effects of different site layouts should be investigated and a mechanism to assign trucks to their proper positions should be established to resolve this problem.

(2) The previous model was not general. It was specific to only one resource combination (two work teams, two trucks for carrying old deck and two trucks for carrying new panels); and cannot be applied to other combinations. Because there

are few resources, the simulation results of the previous model were the same as those of the MicroCYCLONE model. The advantages of cell-based simulation were not demonstrated. In addition, without a series of different resource combinations, it is impossible to compare the productivity and find the optimal combination;

(3) Conflicts among moving objects could not be detected/resolved in the previous model. As a result, animation may be interrupted by these conflicts. In stochastic simulation, various conflict patterns may happen between any objects. How to detect the conflict and design the moving path in the case of conflicts become more complex. From this point of view, defining a set of rules to control moving objects and collecting them into groups that perform specific functions become necessary.

(4) The sizes of objects were not realistic. A truck was represented by only one cell which is too small. Using "blocks" – a number of cells to represent objects is necessary and more realistic in some cases.

In structure:

(1) Some resources such as the *saws* were not included in this model.

(2) Some necessary queues and the logical relationships between models were missing in the previous model. For instance, there was only one output port in the *Bridge* model, when trucks were moving out of the bridge at the same time or

they have overlaps, only the one through this output port could be checked but the others would be ignored.

Because of these limitations, the present research improved and extended the previous study, fully implementing the cell-based simulation methodology. A new simulation model was established to investigate the effects of spatial conflicts on construction productivity and highlight the advantages of this method.

4.2.5 Pre-processing

4.2.5.1 Analyzing Construction Activities and Identifying Involved Resources

The construction activities discussed in Section 4.2.1 was abstracted into several sequential processes: (1) At the beginning of simulation, *Saws* are used to cut the deck into small old sections and work teams (each including two cranes) move beside an old section; (2) Empty trucks for carrying old sections (OS Truck) move on the bridge, stop to load old sections removed by work teams and go off the bridge to the dump area; (3) The trucks loaded with new panels (NP Truck) move on the bridge, stop to unload at the place where the old section has been removed and then go to the plant; (4) The work team moves to the next position and another cycle of activities starts; (5) The OS trucks are unloaded at the dump area and then sent back to the bridge; (6) The empty NP trucks are reloaded with new panels at the plant and sent back to the bridge. Based on the above analysis, the resources involved in the processes are: Teams, Saws, OS Trucks and NP Trucks.

4.2.5.2 Building Cell-based Simulation Models

In this case study, the *Bridge* model is of the most importance of spatial representation because it has space constraints and most of spatial conflicts occur on it. Every detail of the operations on the bridge needs to be visualized. As a result, the *Bridge* should be defined as a Cell-DEVS model. Queues have no spatial constraints and cannot be represented by cells. Therefore, they can only be defined as DEVS models. Transport models from *Bridge* to *Dump Area* or *Plant*, forward or backward, as well as the *Dump Area* model and the *Plant* model have few spatial constraints. Therefore, it is optional to define these models using either of the two types. If spatial problems occur in these models or they are important for the final results, they should be defined as Cell-DEVS models; otherwise, they should be defined as DEVS models. To emphasize the *Bridge* model and speed up the simulation, only the *Bridge* was defined as Cell-DEVS model in this case study.

Based on the analysis in 4.2.4.1, seven models were defined to represent the six processes in the construction. They are: *Controller, Delay-queue, OS Queue, NP Queue, Dump Area, Plant* and *Bridge*. Each model might be composed of several sub-models as shown in Figure 4.5. The functions of each model were explained as follows:

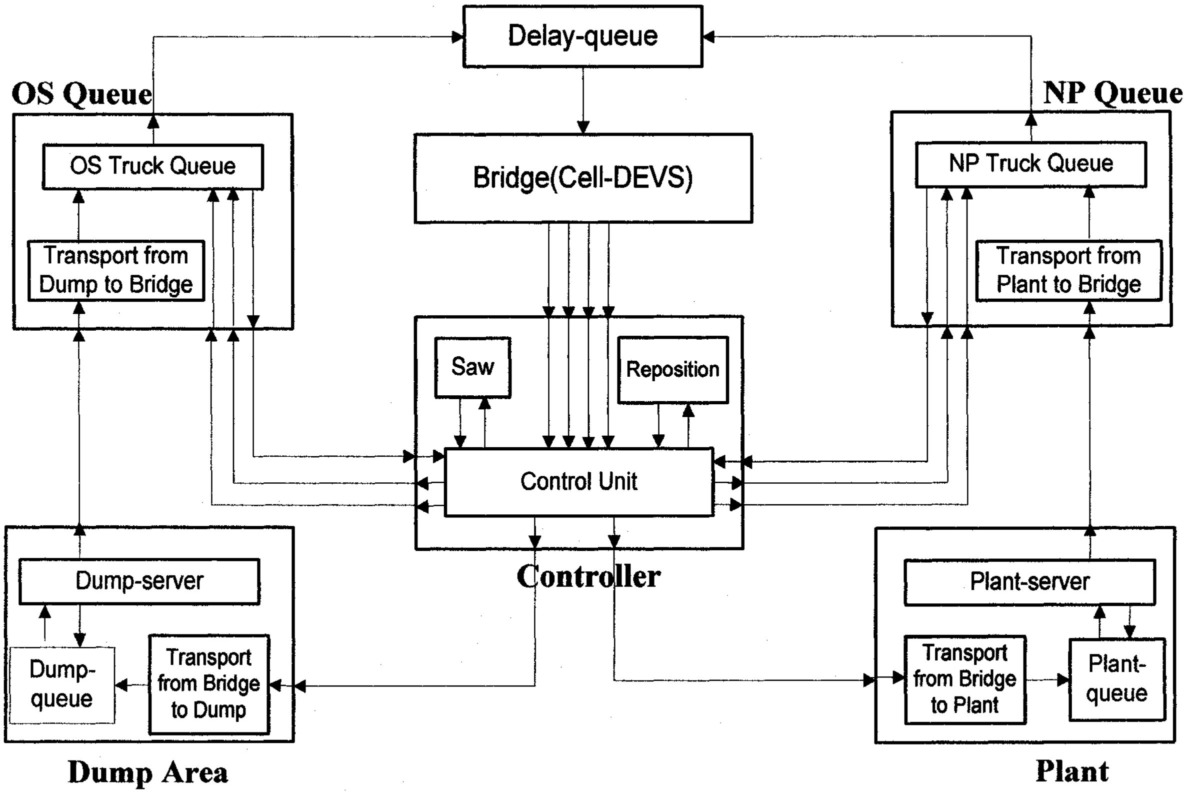


Figure 4.5 Cell-based Simulation Model of the Jacques Cartier Bridge Rehabilitation Project

(1) **Controller.** As shown in Figure 4.5, the *Controller* is the "administrative" model of the system, playing the role of a work site manager, to control OS trucks and NP trucks' receiving and dispatching. This model has three sub-models. *Saw* is used to cut the bridge deck into small old sections. *Reposition* represents a time delay for the reposition of work teams. *Control Unit* is the brain of the system which controls and coordinates the operations of other models. For example, it monitors working state on the bridge by receiving trucks going out of the bridge; it dispatches trucks to *Dump Area* or *Plant* models according to the truck types (NP truck and OS truck); it sends trucks' IDs to *NP Truck Queue* or *OS Truck Queue* and triggers these queues to send a truck to the bridge; it tells *Reposition* to

start a new activity cycle and it controls *Saw* to cut the deck and count the cut sections. Since *Saw* and *Reposition* have a close tie with *Control Unit* and have no communication with other models, they are combined with *Control Unit* to form a coupled DEVS model "*Controller*".

(2) **Dump Area.** *Dump Area* is a coupled DEVS model that consists of *Dump-server, Dump-queue* and *Transport from Bridge to Dump. Dump-server* is the core of this model. *Dump-server* is a workstation where OS trucks loaded with old sections are unloaded by a forklift. When *Dump-server* is idle, it sends a message to *Dump-queue*, asking for the next OS truck. *Dump-queue* stores loaded OS trucks from the sub-model *Transport from Bridge to Dump* in the order of their arrivals. *Transport from Bridge to Dump* represents the duration for OS trucks traveling from *Bridge* to *Dump Area.*

(3) **OS Queue.** This DEVS model is composed of two sub-models: *OS Truck Queue* and *Transport from Dump to Bridge*. *OS Truck Queue* is the focus of this model. *Transport from Dump to Bridge* represents the duration for trucks traveling from *Dump Area to Bridge*. *OS Truck Queue* and *Control Unit* form a mechanism to control OS trucks' dispatching. *OS Truck Queue* receives OS trucks from *Transport from Dump to Bridge* and stores these available OS trucks in a queue. Once it receives an OS truck, *OS Truck Queue* notifies *Control Unit* that an OS truck is ready to be sent, and the latter determines whether or not this truck can be sent out under the current working conditions on the bridge. If the conditions on the bridge are satisfied, *Control Unit* will send a truck ID number, which is determined by the available work teams, to *OS Truck Queue* via the start port,

indicating an OS truck can be sent to the bridge. *OS Truck Queue* will send a truck to *Delay-Queue* with this ID number. After an OS truck is sent, the *OS Truck Queue* sends a feedback message to *Control Unit* to update status on the bridge. Sometimes, there are no available work teams ready for loading when an OS truck arrives in *OS Truck Queue*. The OS truck will wait in the queue until *Control Unit* detects available work teams on the bridge and notify *OS Truck Queue* to send the residing OS truck.

(4) **Delay-queue.** *Delay-queue* is an atomic DEVS model. We suppose that there is only one lane open for trucks to access to the bridge. *Delay-Queue* is used to check and keep the interval between trucks. It separates trucks when they are sent to the bridge at the same time or too close.

(5) **NP Queue**. This model is similar to *OS Queue*. The main difference is that it is used specially for NP trucks' dispatching.

(6) **Plant.** *Plant* is a coupled DEVS model that consists of *Plant-server*, *Plant-queue* and *Transport from Bridge to Plant*. Similar to *Dump Area*, *Plant-server* is a workstation where empty NP trucks are loaded with new panels by a forklift. When *Plant-server* is idle, it sends a message to *Plant-queue*, asking for the next NP truck. *Plant-queue* stores NP trucks from the *Transport from Bridge to Plant* in the order of their arrival. *Transport from Bridge to Plant* represents the duration for NP trucks traveling from *Bridge* to *Plant*.

(7) **Bridge.** The *Bridge* model represents the real bridge workspace where the workers remove old sections and install new panels, and where trucks travel and

stop for loading/unloading. This model should provide workspace for equipment, show the location of resources as well as detect and solve spatial conflicts. It is this model that illustrates the construction spatio-temporal constraints.

A summary of these models and the task durations are presented in Table 4.2.

4.2.5.3 Defining Relationships between Models

The arrows in Figure 4.5 show the directions in which messages are sent. For example, on the bridge, there are six lanes. It is supposed that both the side lanes were reserved for emergency and the other four lanes were used for hauling. Then, four output ports in the *Bridge* model and four input ports in the *Controller* should be defined. In Figure 4.5, the four arrows from *Bridge* to *Controller* mean that four links between *Bridge* and *Controller* are established. Trucks can move out of the bridge through any of the four lanes.

4.2.5.4 Deciding the Suitable Size of Cells and the Dimensions of Each Cell-DEVS Model

Considering the factors discussed in Section 3.4.4, in the case study, the size of cell was defined as 3 x 3 meters. The deck of the main span of the bridge is approximately represented by 200 x 6 cells. Cranes are represented by 1 x 3 cells located beside the old sections which are also represented by 1 x 3 cells. A truck is represented by 4 x 1 cells.

Table 4.2 Summary of Models of Case Study I

Model Type	Coupled Model	Atomic Model	Model Description	Task Duration			Corresponding Task ID in Table 4.1
				Distribution	Mean (min.)	Std. Deviation (min.)	
DEVS	Controller	*Saws*	Cut old sections	normal	18	3	4
		Reposition	Team repositioning	normal	18	3	30
		Control Unit	Controls and coordinates the operations	N/A			N/A
	Dump Area	*Dump-server*	Dump old sections	normal	5	2	13
		Transport from Bridge to Dump	Loaded OS truck travels from the bridge to the dumping area	normal	7	2	10
		Dump-queue	Stores loaded OS trucks	N/A			N/A
	OS Queue	*Transport from Dump to Bridge*	Empty OS truck returns from the dumping area to the bridge	normal	5	2	14
		OS Truck Queue	Stores OS trucks available to be sent to the bridge	N/A			N/A
	Plant	*Plant-server*	Load new panels	normal	14	4	22
		Transport from Bridge to Plant	Empty NP truck returns from the bridge to the plant	normal	5	1	31
		Plant-queue	Stores loaded NP trucks	N/A			N/A
	NP Queue	*Transport from Plant to Bridge*	Loaded NP truck travels from the plant to the bridge	normal	7	1	23
		NP Truck Queue	Stores NP trucks available to be sent to the bridge	N/A			N/A
	N/A	*Delay-queue*	Separate close trucks	N/A			N/A
Cell-DEVS	*Bridge*	*Each cell*	Remove old sections	normal	15	5	9
			Install new panels	normal	26	3	28

4.2.5.5 Defining the Layers and Codes for Each Cell-CEVS Model

This case study adopted the three-layer structure introduced in Section 3.4.5 for the *Bridge* model: the *Occupancy* layer, the *Control* layer and the *ID* layer as shown in Figures 4.6. Some predefined codes are assigned to the cells of each layer as shown in Table 4.3 - Table 4.5.

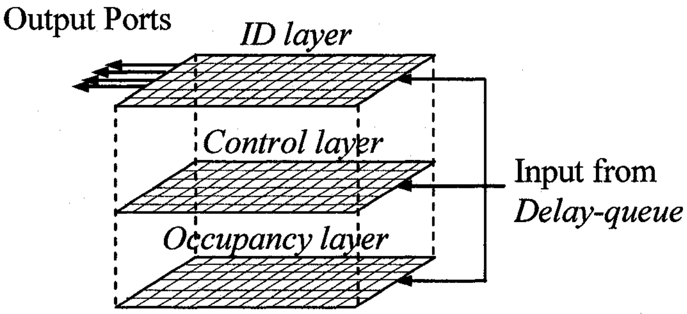


Figure 4.6 The Three Layers Used to Model *Occupancy, Control*, and *ID* Attributes

(1) **Occupancy layer**: This layer has the attribute of the equipment type occupying a cell as shown in Table 4.3. For example, providing that the value of the occupancy layer of a specific cell is "8", we will know that a NP truck carrying new panels is holding this cell. The codes "10"-"15" are used temporarily as signals to inform the slave cells (which will be discussed in Section 4.2.5.7) to change their states.

(2) **Control layer**: This layer controls the movements of mobile objects through changing the states of cells. Codes in this layer indicate the moving state of an object (moving or static) and moving direction (south, north or west). In this way, a cell can detect spatial conflicts between objects through communicating with its

neighbors. Moreover, a set of priority rules were defined on this layer to resolve spatial conflicts as we use traffic lights to control the traffic. Table 4.4 shows the codes used in this layer. For example, the code “6” is used for a mobile object to temporarily stop to avoid a collision.

(3) **ID layer**: This layer contains the ID number of each piece of equipment. This ID number can be used to identify and trace a specific object. Table 4.5 shows the codes used on this layer. The ID layer receives messages from *Delay-queue* and conveys the information about a certain time, location (cell) and ID of objects to *Control Unit*. It should be noted that the ID number of an object is different from the code assigned to a cell. As discussed in Section 4.2.1, the maximum number of teams on the bridge is ten. Accordingly, the ID numbers assigned to teams start from 91 to 99, which are the same as their codes. However, a truck is represented by four cells. To manipulate the state of each cell and trace a specific truck easily, four codes are used for one truck, but its ID number is the fourth code of the truck divided by 4. For example, the codes “5” – “8” represent an OS truck. Its ID number is 2 which equals to “8 / 4”. In addition, to simplify the development of rules, the team’s ID is given to only one of the cells occupied by a team, as show in Figure 4.8.

Collecting the attribute values on each layer, the state of a specific cell can be defined using a triplet *<Occupancy, Control, ID>*. For example, if a cell occupied by an empty OS truck with ID code “1” is moving west on the bridge before loading old sections, its state can be represented as <1, 4, 1>.

Table 4.3 Codes Used in the *Occupancy* Layer of Case Study I

Model	Code	Descriptions
Bridge	0	Empty cell
	1	An OS truck before loading old sections
	2	Crane
	3	Old sections
	4	An OS truck after loading old sections
	5	Empty space after removing the old sections
	6	New installed panel
	8	An NP truck before unloading
	9	An NP truck after unloading
	10 -15	For temporary use

Table 4.4 Codes Used in the *Control* Layer of Case Study I

Model	Code	Descriptions
Bridge	0	Empty cell
	1	Truck moves north
	2	Truck moves south
	4	Truck moves west
	5	Static object
	6	Truck temporarily stops – waiting delay

Table 4.5 Codes used in the *ID* Layer of Case Study I

Model	Code	Descriptions
Bridge	0	Empty cell
	1 - 40	ID of an OS truck
	41 - 80	ID of an NP truck
	91 -99	ID of a work team

4.2.5.6 Defining the Zones of Each Cell-DEVS Model

In the case study, the cellular space could be divided into two zones. The major difference of the rules applied to these two zones is that a truck takes different moving directions when it meets an obstacle. For example, when a spatial conflict between a truck and a work team occurs, the direction of the truck will change to south or north to avoid the obstacle depending on whether the truck is in zone-1 or zone-2, respectively (Figure 4.7). The local rules specific to a zone allow exceptions to common situations.

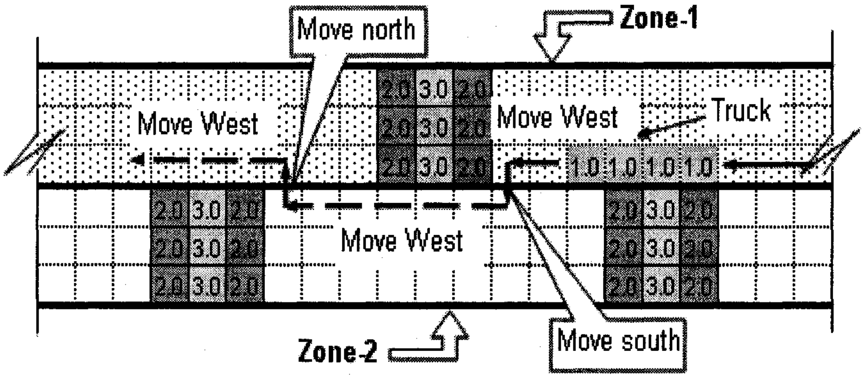


Figure 4.7 Zone Division and the Rules for Changing Directions in Zone-1

4.2.5.7 Developing Rules for Each Cell-DEVS Model

Section 3.4.7 explained how to define rules and utilize them to control the state of cells. In practice, rules can be divided into several types to improve the reusability. For example, in this case study, two types of rules are defined. The first type is regular rules which control the object to move or to turn north and south under regular conditions (no conflicts) as shown in Figure 4.8(a). Regular rules can be easily reused for other applications. The other type is priority rules which regulate the movement of objects when conflicts occur. For example, in Figures 4.8(b) - (f), when *Truck-15* is moving west, it meets an obstacle (Team 92) and needs to turn north to avoid it. However, it has overlaps with *Truck-1* and *Truck-2* respectively, which are also trying to move north. Conflicts between moving objects occur. In order to resolve these conflicts, the following priority rules are defined for this case study.

- When two trucks are moving on the same lane, the interval between them is at least two cells; otherwise, the latter has to stop temporarily, giving the priority to the former one.
- When two trucks are moving on different lanes (e.g., *Truck-1* and *Truck-15* in Figures 4.8(c) and (d)), if the truck on the upper lane is behind the truck on the lower lane but they have an overlap, the truck on the upper lane has the priority. The same rules are applied to the case shown in Figure 4.8(e).
- When two trucks are moving on different lanes (e.g., *Truck-1* and *Truck-2* in Figures 4.8(c) and (d)), if the truck on the upper lane is behind the truck on the lower lane without any overlap, the truck on the lower lane has the priority.
- When a truck is taking the priority to turn, other mobile objects that may cause spatial conflicts with this truck have to keep their positions until the truck leaves (Figures 4.8(c) and (d)).

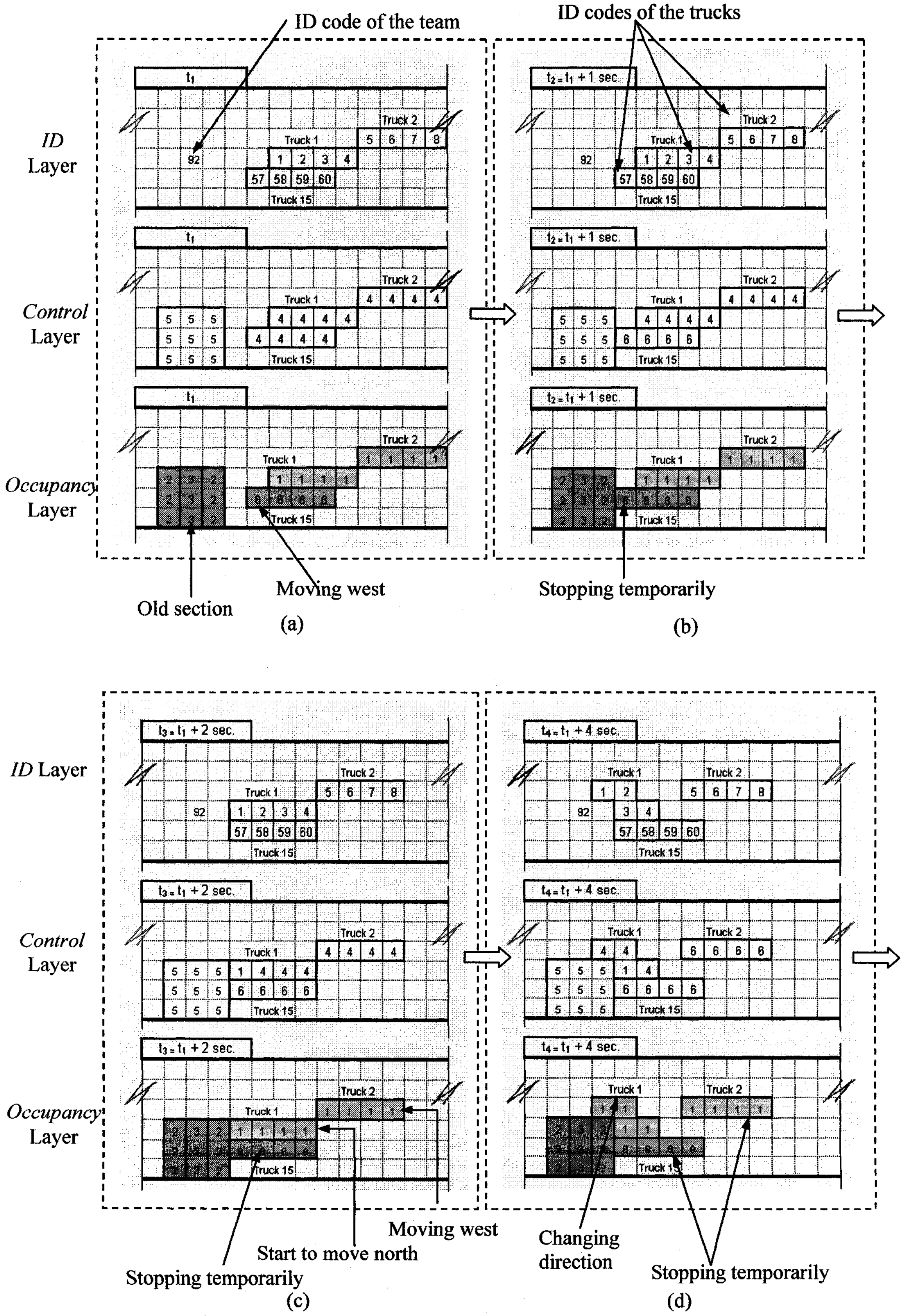


Figure 4.8 Examples of Spatial Conflicts

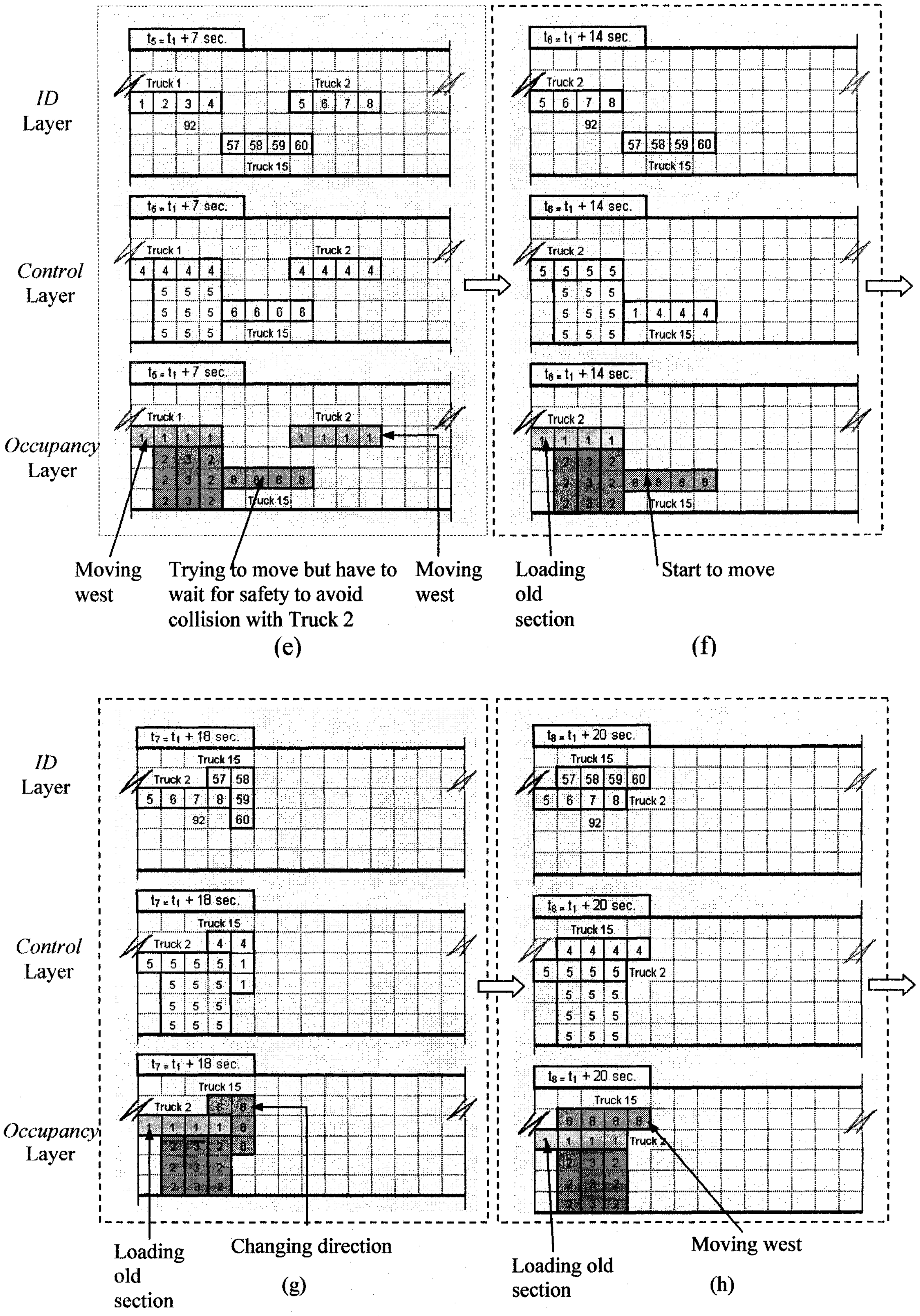


Figure 4.8 Example of Spatial Conflicts (cont'd)

Defining a set of rules to regulate the behavior of objects is the major job of cell-based modeling. In some cases, multiple cells are required to represent an object for a more accurate description. However, in this way, a great number of rules might be needed with increasing the number of cells under control. For example, in this case study, four cells are used to represent a truck. Thus, more rules have to be defined for the four cells since each cell is independent from the others. To reduce the effort of developing rules, the master-slave concept is provided based on the present research. This concept is to assign one cell of an object's group as the master cell and the others as slave cells which follow the same actions as the master cell during simulation. In this way, the user can only focus on the behaviors of the master cell and the slave cells will follow each other one by one. For example, in this case study, the first cell of the truck is defined as the master cell and

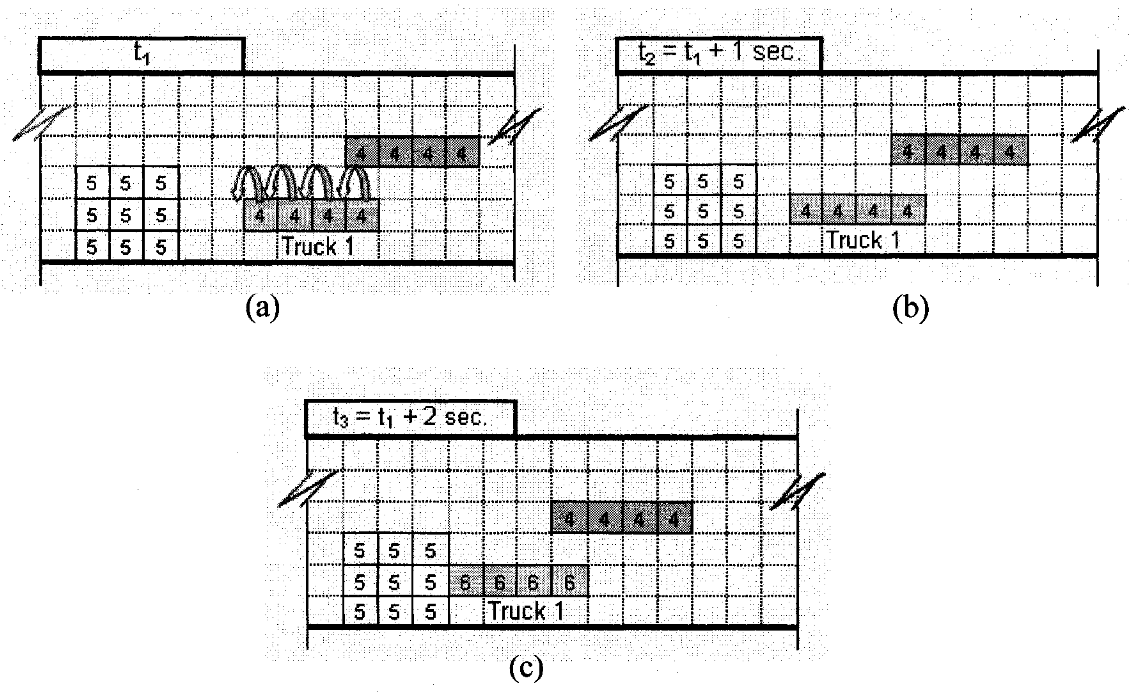


Figure 4.9 Examples of Master-Slave Cells
(*Control* Layer)

the other three cells as slave cells. As shown in Figure 4.9(a), at time t1, *Truck-1* is moving west (the direction code is "4"). At the next time t2, the truck continues moving west as shown in Figure 4.9(b). Comparing Figures 4.9(a) and (b), each slave cell of the truck took its preceding cell's direction code. The second cell took the master's direction code; the third cell took the second one's direction code and so on. In this case, the rule for the master cell must be defined solely while the rules for all the three slave cells could be derived from the rule of the master cell. Figure 4.10 shows the rules that control the master cell and slave cells on the *Control* layer, which lead to the movement of objects as shown in Figures 4.9(a) and (b) . As a result, the master-slave method will greatly reduce the efforts needed to define rules.

Rule for the master cell (*control* layer):

```
rule:4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-1,2,-1) != 3 and
(-2,-1,-1) != 3}
```

Note: if the master cell of a truck detects no obstacles, it will continue to move west at the next step.

Rule for the slave cells (*Control* layer):

```
rule:{(0,0,0)} 1 {((0,1,-1) = 1 or (0,1,-1) = 8 or (0,1,-1) = 4
or (0,1,-1) = 9 ) and (0,1,0) = 4 and remainder((0,1,1),4) != 1}
```

Note: the slave cells will take the preceding cell's direction code at the next step.

Figure 4.10 Example of Rules for Master-slave cells

However, there are some exceptions that should be taken into consideration when applying this method. This method cannot be used for priority rules. For example, in Figure 4.9(c), at time t3, *Truck-1* will stop. That means both the master cell and slave cells have to change their state from "4" (moving west) to "6" (temporarily stop) at the same time. In this case, four rules have to be defined to control the four cells respectively. This concept has been applied in the case study. A detailed definition of the rules used for this case study can be found in Appendix B.

4.2.5.8 Developing DEVS Models

In Section 4.2.5.2, the functions of the DEVS models have been discussed. These DEVS models are implemented by programming using C++. When programming, it is desirable to inherit the cods of existing models. Considering the reusability, this research provides some basic models that can be reused later and stores them in a "Modeling Element Library". For example, in the case study, *Queues* are used as containers to store units (e.g., trucks) in an order. In the *Queues*, the "List" containers are employed to store each of the elements they contain. The ordering is kept by the association to each element of a link to the element preceding it and a link to the element following it. "Lists" perform greatly in inserting, extracting and moving elements in any position within the container. Storage is handled automatically by the class, allowing lists to be expanded and contracted as needed. The user can easily use the "public member functions" provided to access the current size of the container and each element. As a result, many other *Queues* can be defined based on a basic one with minor modifications according to different requirements. An example of a DEVS model is shown in Appendix C.

4.2.6 MAIN-PROCESSING

4.2.6.1 Site Layout Patterns

Figure 4.11 shows a schematic representation of the worksite layout during the deck replacement. Most of the time, two teams worked in parallel at different positions of the bridge. Each team included two telescopic cranes located at both sides of the section to be replaced. Two types of semi-trailer trucks – OS truck and NP truck – were used to transport the old sections and new panels respectively.

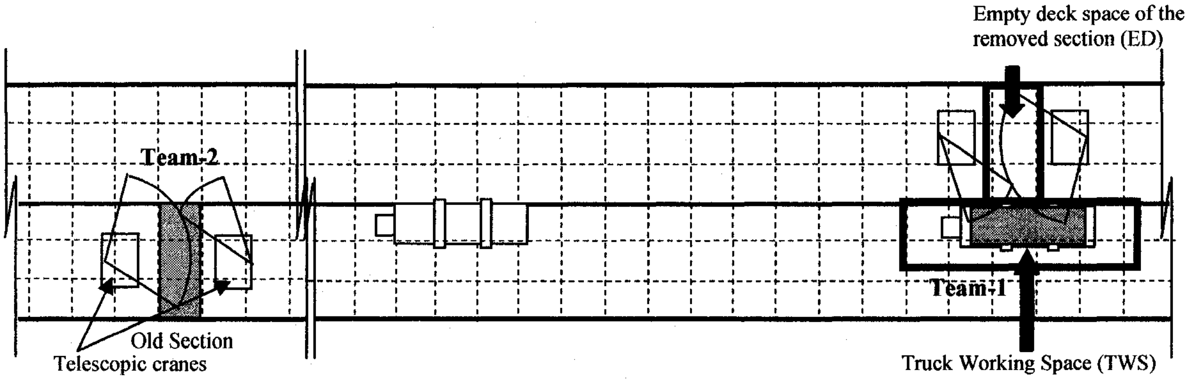
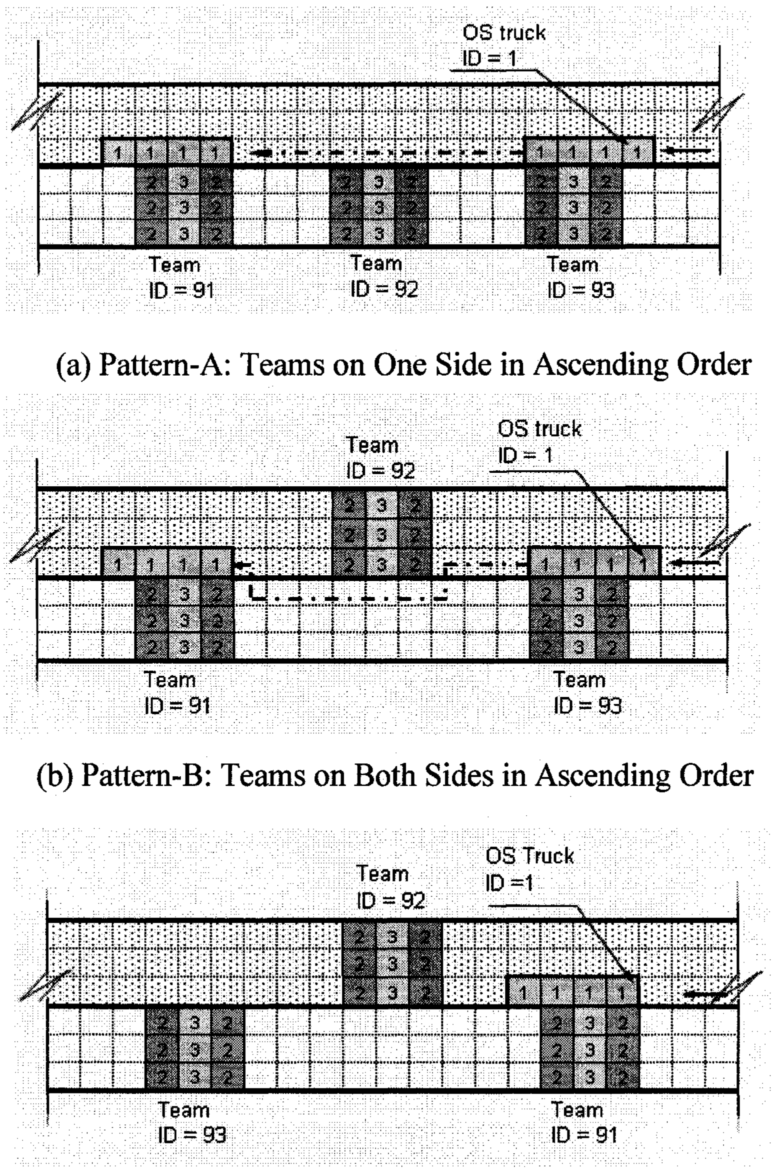

Empty deck space of the removed section (ED)
Team-2
Team-1
Old Section
Telescopic cranes
Truck Working Space (TWS)

Figure 4.11 Worksite Layout of the Bridge Rehabilitation Project

To get in-depth insight into this case study, this site layout was extended to accommodate more resource combinations and to investigate the effects of different site layout patterns. Actually, various patterns can be applied. In this case study, three representative patterns of site layout was chosen as shown in Figure 4.12. In pattern-A, teams are on one side of the bridge and their ID numbers are in ascending order; in pattern-B, teams are on both sides of the bridge in ascending order, while in pattern-C, teams are on both sides of the bridge in descending order.

Since the worksite is explicitly represented by cells, the objects can be controlled by rules and each object can be identified by its ID, a mechanism can be established using the ID attributes. When a truck is moving from west to east, it checks if it is passing a team and if this team is its corresponding team. If the truck finds its corresponding team, i.e., the



(a) Pattern-A: Teams on One Side in Ascending Order

(b) Pattern-B: Teams on Both Sides in Ascending Order

(c) Pattern-C: Teams on Both Sides in Descending Order

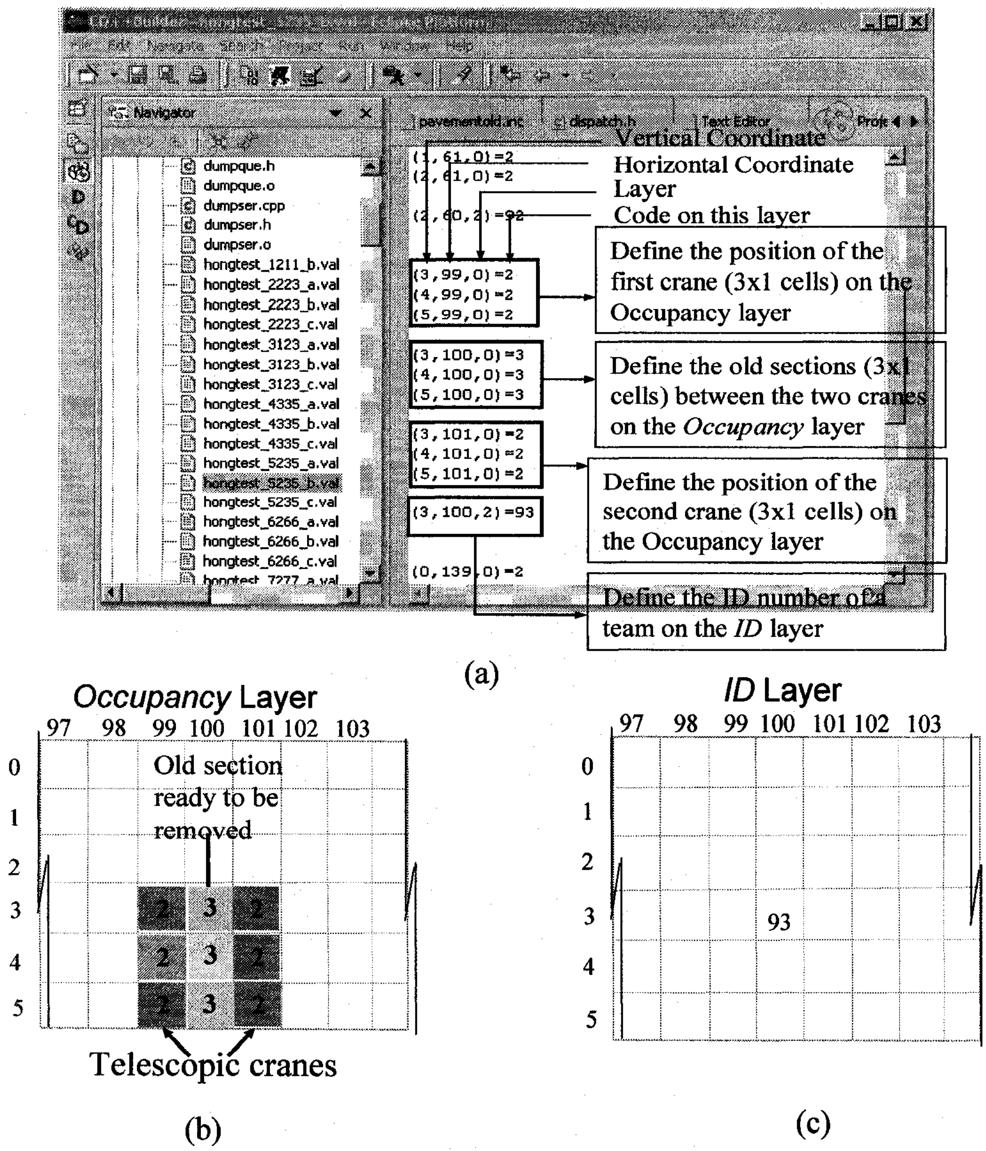
Figure 4.12 Possible Patterns of Teams' Order and Layout

truck's ID number matches that of the team, it will stop for loading old sections or unloading new panels; otherwise, it will move on. In this way, different site layout patterns and the moving path of objects can be considered. For example, the OS truck with the ID number of 1 will pass through any other teams until it stops at its corresponding team, which has the ID number of 91, for loading old sections. The OS truck with ID number of 2 will stop at the team 92, and so on. In this way, the team layout and the order of the team's ID numbers determine where the trucks should stop. Consequently, the results of different worksite layouts can be compared.

4.2.6.2 Initializing Resources

Before simulation, all OS trucks and NP trucks are listed in the *Dump Area* and *Plant* respectively by programming. For the *Bridge* model, resources are defined using a text file with an extension name (.val). Teams can be placed on the bridge at the beginning of the simulation. The surface of the bridge is represented by 6 x 200 cells. The position of the top-left corner is (0,0) and the bottom-right corner is (5,199) on each layer. Figure 4.3 demonstrates the resource initialization. For example, Figure 4.13 (a) defined the team with ID number of 93. It consists of two telescopic cranes which are located in the lower part of the bridge (Zone-2) with their horizontal coordinate values of 99 and 101 respectively. The code on the *Occupancy* layer was "2", which meant the object was a crane. Between the two cranes is an old section to be removed, represented by the code"3". These occupation attributes should be defined on the *Occupancy* layer (Figure 4.13 (b)) and the ID number of this team should be assigned on the *ID* layer (Figure 4.13 (c)).

For simplicity, in the following sections, the acronym TSON is used to indicate resource combinations, e.g., TSON 5235 means that 5 Teams are working on the bridge, 2 Saws are used to cut the old deck, 3 OS Trucks are used for carry old sections and 5 NP Trucks are used to carry new panels.



```
(1,61,0)=2
(2,61,0)=2

(2,60,2)=92

(3,99,0)=2
(4,99,0)=2
(5,99,0)=2
(3,100,0)=3
(4,100,0)=3
(5,100,0)=3
(3,101,0)=2
(4,101,0)=2
(5,101,0)=2
(3,100,2)=93

(0,139,0)=2
```

(a)

(b)

(c)

Figure 4.13 Initializing Resources in the Cell-DEVS Model

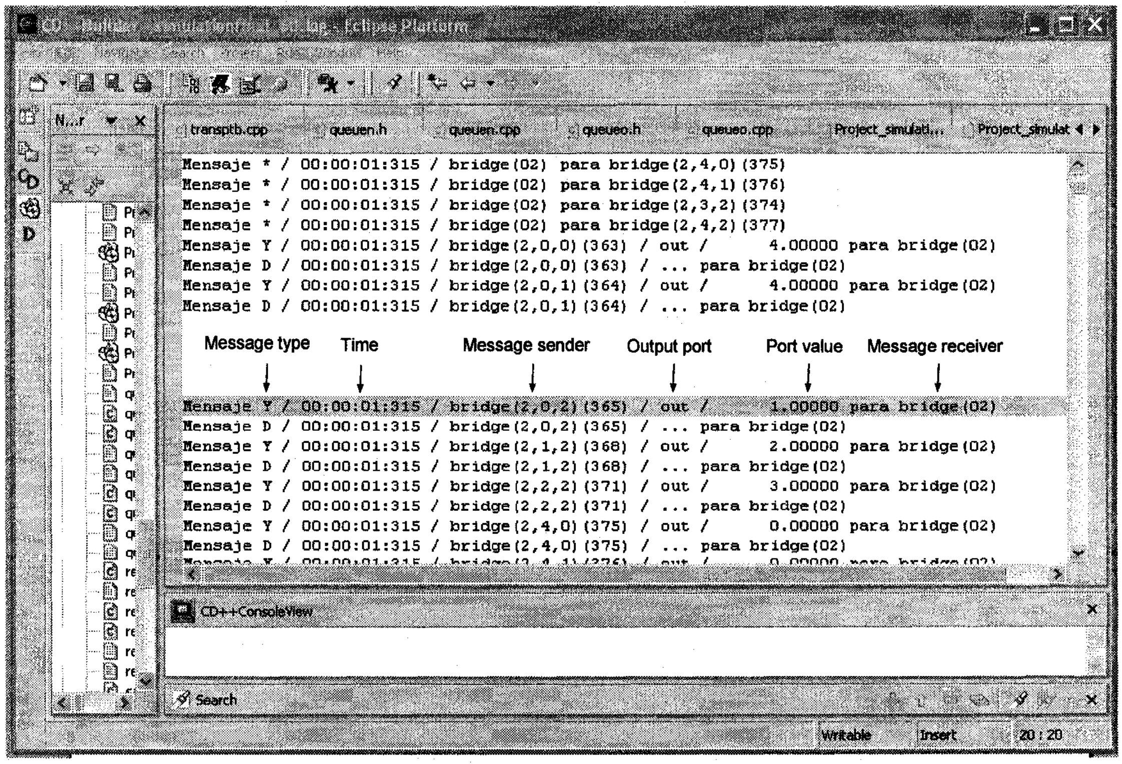
4.2.7 POST-PROCESSING

4.2.7.1 Verification

As mentioned in Section 3.6.1, the process of verification is exercised through examination of output files. This section explains how to examine these files using case study of the Jacques Cartier Bridge Rehabilitation Project. In order to show the simulation results, a minimized model with a space of 6 x 60 cells was used. The resource combination is TSON 5235 (5 teams, 2 saws, 3 OS trucks and 5 NP trucks) and the site layout pattern is Pattern-B as shown in Figure 4.11(b).

(1) (.log) file

This is the most important output file of CD++ since it keeps a record of all the messages sent between models and the ones between cells in the order of simulation



```
Mensaje * / 00:00:01:315 / bridge(02) para bridge(2,4,0)(375)
Mensaje * / 00:00:01:315 / bridge(02) para bridge(2,4,1)(376)
Mensaje * / 00:00:01:315 / bridge(02) para bridge(2,3,2)(374)
Mensaje * / 00:00:01:315 / bridge(02) para bridge(2,4,2)(377)
Mensaje Y / 00:00:01:315 / bridge(2,0,0)(363) / out /      4.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,0,0)(363) / ... para bridge(02)
Mensaje Y / 00:00:01:315 / bridge(2,0,1)(364) / out /      4.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,0,1)(364) / ... para bridge(02)

Mensaje Y / 00:00:01:315 / bridge(2,0,2)(365) / out /      1.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,0,2)(365) / ... para bridge(02)
Mensaje Y / 00:00:01:315 / bridge(2,1,2)(368) / out /      2.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,1,2)(368) / ... para bridge(02)
Mensaje Y / 00:00:01:315 / bridge(2,2,2)(371) / out /      3.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,2,2)(371) / ... para bridge(02)
Mensaje Y / 00:00:01:315 / bridge(2,4,0)(375) / out /      0.00000 para bridge(02)
Mensaje D / 00:00:01:315 / bridge(2,4,0)(375) / ... para bridge(02)
Mensaje Y / 00:00:01:315 / bridge(2,4,1)(376) / out /      0.00000 para bridge(02)
```

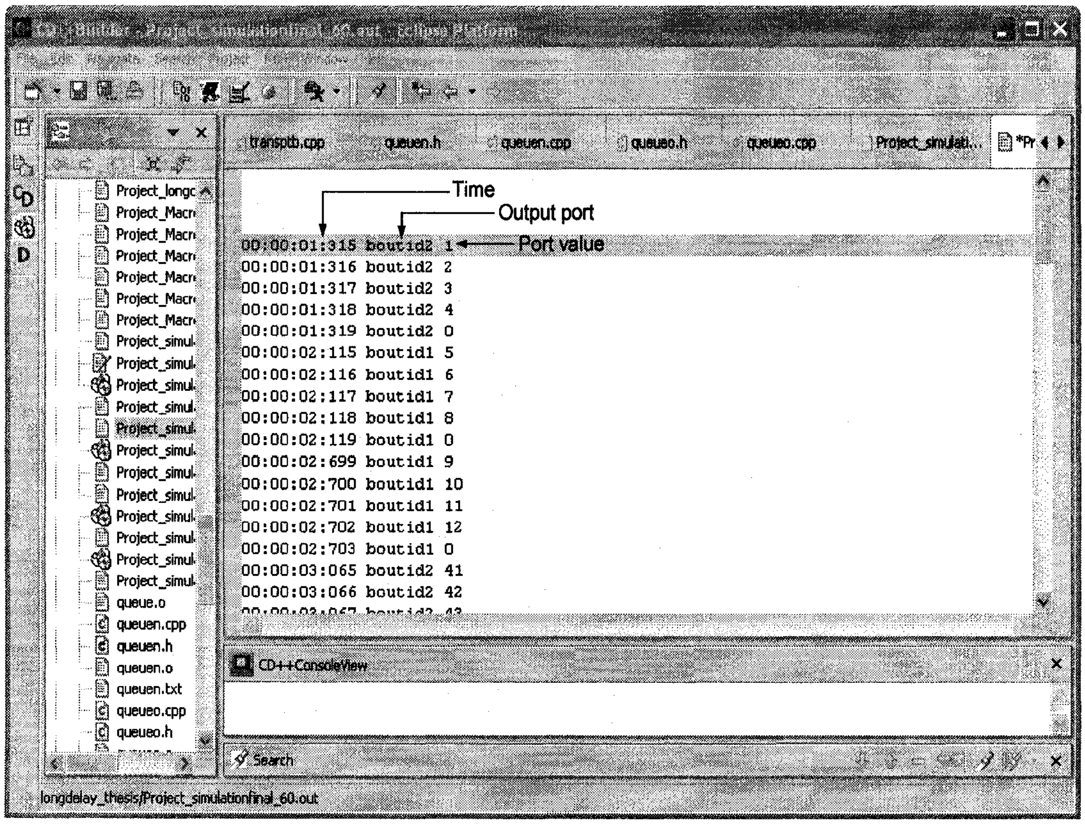
Figure 4.14 Example of (.log) Files

time as shown in Figure 4.14. For example, the highlighted message in Figure 4.14 states that at the time 01:315, an output message (type Y) with its value 1 (the ID code of the first cell of OS *Truck-1*) is sent from the cell (2,0,2) to the second output port (value = 2) of the Bridge model. This simulation state indicates that the OS *Truck-1* is leaving the bridge on the Lane 2.

The (.log) file presents event messages from multiple perspectives and at the most detailed level so that it provides a foundation for very detailed examination of simulation states. However, because of this, a large amount of data may be produced over a long span of simulation time, which would be extremely cumbersome to check in detail for correctness. To make the examination practical, it is necessary to observe the simulation results from other specific points of view as follows.

(2) (.out) file

This file records the output events that are defined in the (.ma) file. It is generated by the simulator automatically. It has a standard format which is composed of simulation time, output port and the port value as shown in Figure 4.15. For example, the message highlighted in Figure 4.15 represents the state at time 01:315. The first cell (port value = 1) of the OS *Truck-1* is at the second output port of the *Bridge* model (boutid2). The (.out) file reflects the simulation performance from the perspective of output message and gives a clear description over simulation time.

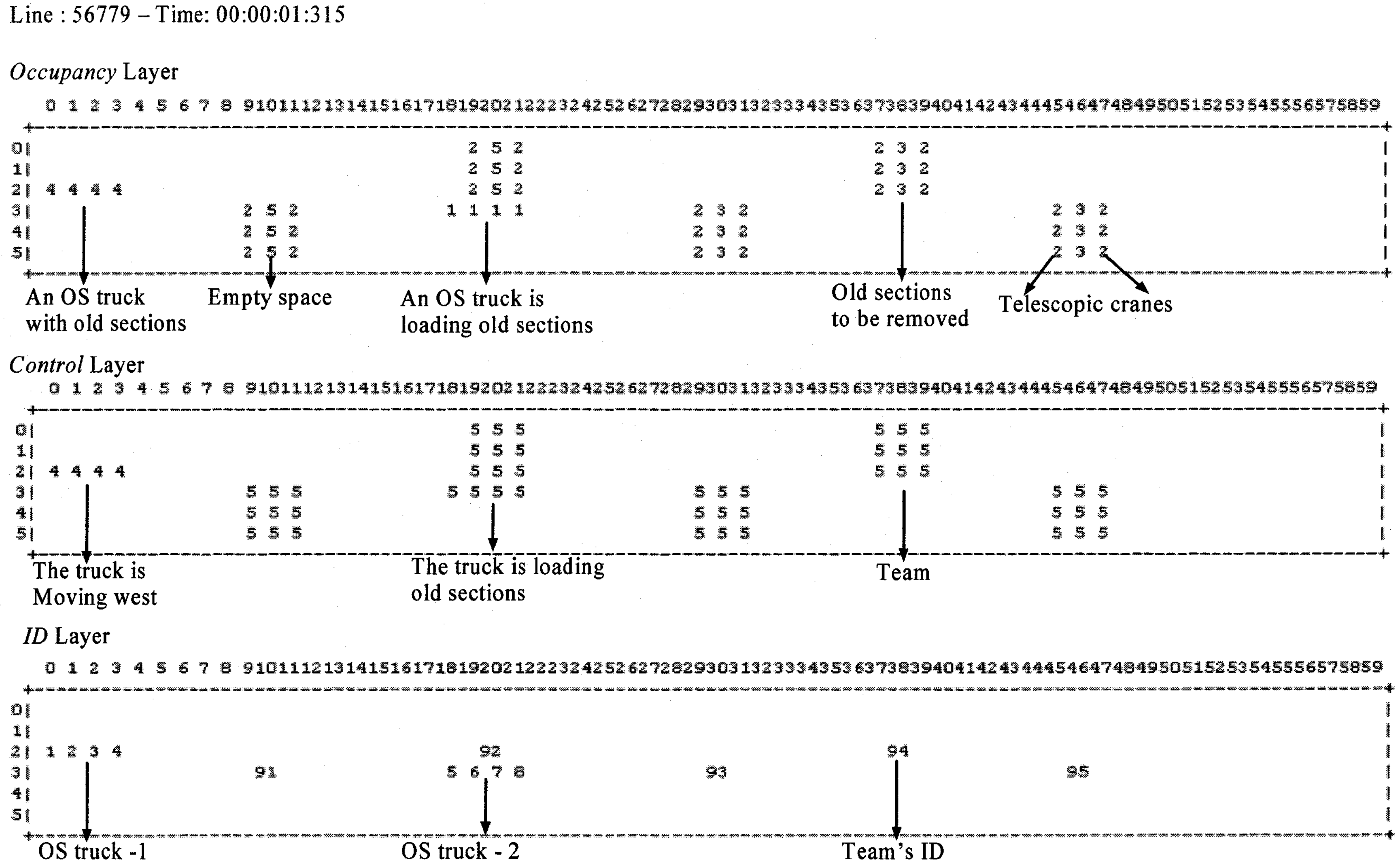


```
00:00:01:315 boutid2 1
00:00:01:316 boutid2 2
00:00:01:317 boutid2 3
00:00:01:318 boutid2 4
00:00:01:319 boutid2 0
00:00:02:115 boutid1 5
00:00:02:116 boutid1 6
00:00:02:117 boutid1 7
00:00:02:118 boutid1 8
00:00:02:119 boutid1 0
00:00:02:699 boutid1 9
00:00:02:700 boutid1 10
00:00:02:701 boutid1 11
00:00:02:702 boutid1 12
00:00:02:703 boutid1 0
00:00:03:065 boutid2 41
00:00:03:066 boutid2 42
```

Figure 4.15 Example of (.out) Files

(3) (.drw) file

This file is used to view the state of Cell-DEVS models in a simpler way. It can be automatically generated using the drawlog tool after simulation. As shown in Figure 4.16, it explicitly demonstrates the coordinates of the cellular space, the event time and the codes on the three layers. Referring to Tables 4.3- 4.5 for the codes used in the three layers, it is easy to understand the simulation state.



Line : 56779 – Time: 00:00:01:315

Occupancy Layer

```
   0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829303132333435363738394041424344454647484950515253545556575859
 +--------------------------------------------------------------------------------------------------------------------------+
0|                                       2 5 2                               2 3 2                                        |
1|                                       2 5 2                               2 3 2                                        |
2| 4 4 4 4                               2 5 2                               2 3 2                                        |
3|                   2 5 2             1 1 1 1               2 3 2                           2 3 2                        |
4|                   2 5 2                                   2 3 2                           2 3 2                        |
5|                   2 5 2                                   2 3 2                           2 3 2                        |
 +--------------------------------------------------------------------------------------------------------------------------+
```

An OS truck with old sections

Empty space

An OS truck is loading old sections

Old sections to be removed

Telescopic cranes

Control Layer

```
   0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829303132333435363738394041424344454647484950515253545556575859
 +--------------------------------------------------------------------------------------------------------------------------+
0|                                       5 5 5                               5 5 5                                        |
1|                                       5 5 5                               5 5 5                                        |
2| 4 4 4 4                               5 5 5                               5 5 5                                        |
3|                   5 5 5             5 5 5 5               5 5 5                           5 5 5                        |
4|                   5 5 5                                   5 5 5                           5 5 5                        |
5|                   5 5 5                                   5 5 5                           5 5 5                        |
 +--------------------------------------------------------------------------------------------------------------------------+
```

The truck is Moving west

The truck is loading old sections

Team

ID Layer

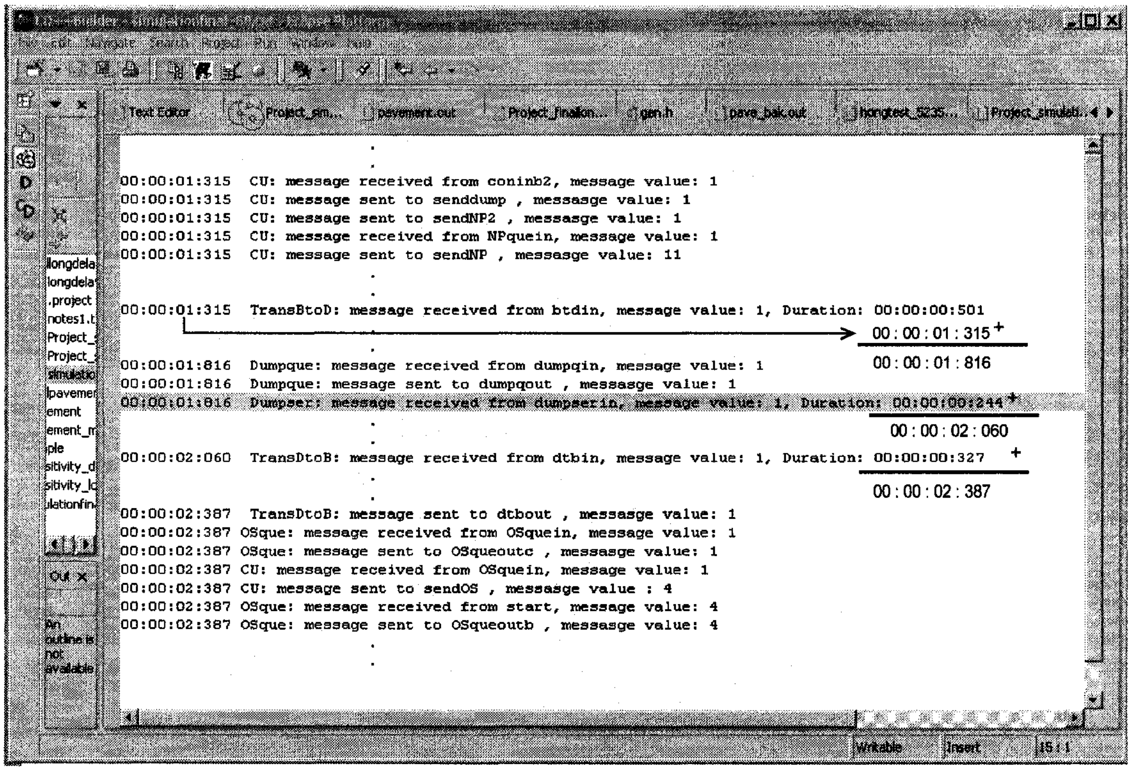
```
   0 1 2 3 4 5 6 7 8 91011121314151617181920212223242526272829303132333435363738394041424344454647484950515253545556575859
 +--------------------------------------------------------------------------------------------------------------------------+
0|                                                                                                                        |
1|                                                                                                                        |
2| 1 2 3 4                               92                                  94                                         |
3|                   91                5 6 7 8               93                              95                           |
4|                                                                                                                        |
5|                                                                                                                        |
 +--------------------------------------------------------------------------------------------------------------------------+
```

OS truck -1

OS truck - 2

Team's ID

Figure 4.16 Example of (.drw) Files (Minimized model)



```
                         .
                         .
00:00:01:315  CU: message received from coninb2, message value: 1
00:00:01:315  CU: message sent to senddump , messasge value: 1
00:00:01:315  CU: message sent to sendNP2 , messasge value: 1
00:00:01:315  CU: message received from NPquein, message value: 1
00:00:01:315  CU: message sent to sendNP , messasge value: 11
                         .
                         .
00:00:01:315  TransBtoD: message received from btdin, message value: 1, Duration: 00:00:00:501
                         .
                         .
00:00:01:816  Dumpque: message received from dumpqin, message value: 1
00:00:01:816  Dumpque: message sent to dumpqout , messasge value: 1
00:00:01:816  Dumpser: message received from dumpserin, message value: 1, Duration: 00:00:00:244
                         .
                         .
00:00:02:060  TransDtoB: message received from dtbin, message value: 1, Duration: 00:00:00:327
                         .
                         .
00:00:02:387  TransDtoB: message sent to dtbout , messasge value: 1
00:00:02:387 OSque: message received from OSquein, message value: 1
00:00:02:387 OSque: message sent to OSqueoutc , messasge value: 1
00:00:02:387 CU: message received from OSquein, message value: 1
00:00:02:387 CU: message sent to sendOS , messasge value : 4
00:00:02:387 OSque: message received from start, message value: 4
00:00:02:387 OSque: message sent to OSqueoutb , messasge value: 4
                         .
                         .
```

Figure 4.17 Example of (.txt) Files

(4) (.txt) file

This file is designed specially for DEVS models. Since the DEVS model is programmable, when writing codes, the user can output any value of interest to a (.txt) file with a user-defined format. This file overcomes the redundancy in the (.log) file and offers a concise way for information output. Therefore, it is very useful to trace specific entities. A trace in a simulation program would give the value of selected variables each time an event occurred. Figure 4.17 shows a simple example of tracing the OS *Truck-1* by outputting the external and output messages of each model in Figure 4.4 (the relationship of models). From the (.drw) file (Figure 4.16), it has been known that at time 01:315, the OS *Truck-1* began to leave the *Bridge* model. After leaving the *Bridge*, this truck entered into *Control Unit*. *Control Unit* got the truck ID number, listing the ID and sending the OS truck to the model *Transport from Bridge to Dump*. At the same time, *Control Unit* asked the *NP Truck Queue* if there were NP trucks available. In this case, the answer was "yes" (the value of NPquein = 1) and a NP truck with its ID number 11 was sent out. After the duration needed for the transport from *Bridge* to *Dump Area,* this truck passed through *Dump-queue*, *Dump Server*, *Transport from Dump to Bridge* and finally it entered into the *OS Truck Queue*, preparing for a new cycle. At time 02:387, when *OS Truck Queue* received an OS truck, it would ask *Control Unit* if this truck could be sent out right away. *Control Unit* calculates if there was available space on the bridge. If the answer was "yes", *Control Unit* would feedback with an ID number to *OS Truck Queue* (in this case, the ID number is 4). The latter received this message and sent out the OS truck with this ID number (since the combination is TSON 5235, there were only three OS trucks. When the OS truck-1 form the first cycle was sent out for the second cycle, its ID number was changed from 1 to 4 to load the old sections at Team 94).

These output files provide a way to observe the simulation state at a specific point of time from different perspectives. For example, all the figures from Figure 4.14 to 4.17 present the state at the time 01:315. When associating the four output files with Figure 4.5 and Table 4.2 together, the user will have a whole picture of the simulation states over time and can easily check the logic and accuracy of this simulation system.

4.2.7.2 Visualizing the Cell-DEVS Models

As an example, Figure 4.18 shows the animation snapshots of the *Occupancy* layer of the *Bridge* model at different time points. With animation, the user can easily identify each object and its state.

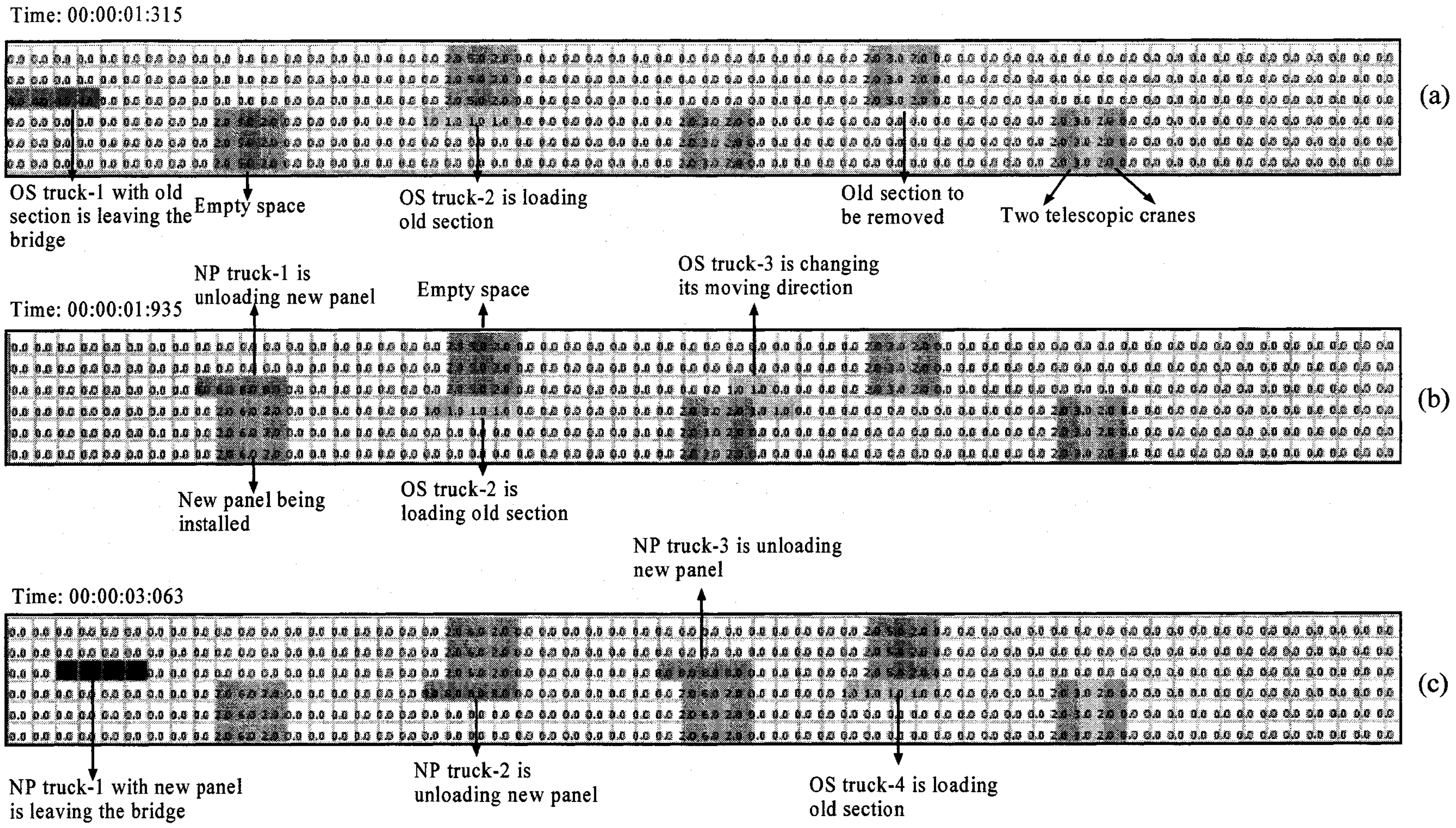

Time: 00:00:01:315
(a)
OS truck-1 with old section is leaving the bridge
Empty space
OS truck-2 is loading old section
Old section to be removed
Two telescopic cranes
Time: 00:00:01:935
NP truck-1 is unloading new panel
Empty space
OS truck-3 is changing its moving direction
(b)
New panel being installed
OS truck-2 is loading old section
Time: 00:00:03:063
NP truck-3 is unloading new panel
(c)
NP truck-1 with new panel is leaving the bridge
NP truck-2 is unloading new panel
OS truck-4 is loading old section

Figure 4.18 Snapshots of the *Occupancy* Layer of the *Bridge* Model (Minimized model)

4.2.7.3 Analyzing Simulation Results

(1) Assumptions

To compare the cell-based simulation method with MicroCYCLONE, the analysis of simulation results are based on the following assumptions: (1) The simulation time is 9 hours to represent the real operation shift from 8:30 pm to 5:30 am of the next day; (2) For simplicity, this case study concentrates on the spatial conflicts occurring in the *Bridge* model and ignore the spatial constraints in other models. It is acceptable because all the heavy equipment and trucks are working on the bridge within a limited workspace so most of the spatial conflicts are expected to occur on the bridge; (3) Both the OS trucks and NP trucks move on the bridge at a constant speed of 10 km/h. Therefore, it takes about one second for a truck to move from one cell to the next; (4) There is only one lane for the trucks to access to the bridge and four lanes for them to go off it; and (5) For economic analysis, we assigned the cost to each resource based on the real requirements. Each team including two telescopic cranes on *Bridge* for removing the old sections and installing the new panels costs \$450/ hour; the saw for cutting the old deck costs \$100/ hour; each truck, no matter OS truck or NP truck, costs \$50/ hour; and the forklift in the *Dump Area* and the small crane in the *Plant* cost \$150/hour each.

(2) Types of Delays Resulting from Spatial Conflicts

The emphasis of this case study is to investigate the effects of spatial conflicts on productivity. Therefore, it is important to analyze the type of conflicts on the bridge as well as the delays resulting from these conflicts. As presented in Tables 4.3 - 4.5, some codes are used to indicate spatial conflicts on the *Control* layer. Assume that if all the trucks on the bridge always move straight on an east-west axis, there will be no spatial conflicts or delays. However, trucks on the bridge may encounter various spatial conflicts. They may change directions to turn around obstacles such as cranes or other trucks, which results in changing direction delays. In other cases, a truck might stop temporarily and gives the moving priority to another one, which results in waiting delays. The codes "1" and "2" signify the "changing direction delay" because it takes time for a truck to move north or south; the code "6" signifies the "waiting delay" that a truck needs to stop temporarily. Occasionally, two or more trucks may get to the bridge at the same time or very close. The *Delay-queue* model will separate them for a short interval and the delays due to the separation can be recorded.

(3) Quantifying delays resulting from spatial conflicts

As discussed above, three types of delays, the "changing direction delays", "waiting delay" and the delays due to separation, are considered in this case study. Since the (.drw) file keeps a detailed simulation result of the *Bridge* model, the total delays resulting from spatial conflicts can be quantified by reading this file and picking out the codes that indicate delays and add the accumulated delays for separation.

In this case study, all of the three types of delays were counted for 45 resource combinations of each site layout pattern to investigate the impact of different resource combinations and site layout patterns on the delays resulting from spatial conflicts. As shown in Figure 4.19, it is found: (1) For Pattern-B and Pattern-C, the delays are sensitive to the number of teams. With the increase of the number of teams, the delays increase significantly. The number of trucks also influences the delays. For Pattern-A, the influence of different resource combinations is much less; and (2) The delay of Pattern-A is significantly less than that of Pattern-B or Pattern-C for each resource combination, while the difference between Pattern-B and Pattern-C is subtle. Therefore, from the perspective of less spatial conflicts or delays, Pattern-A is better than the others, which is consistent with our expectation since there are less directional changes in Pattern-A. This result indicates that the spatial delays can be decreased by properly arranging the resources and selecting resource combinations.

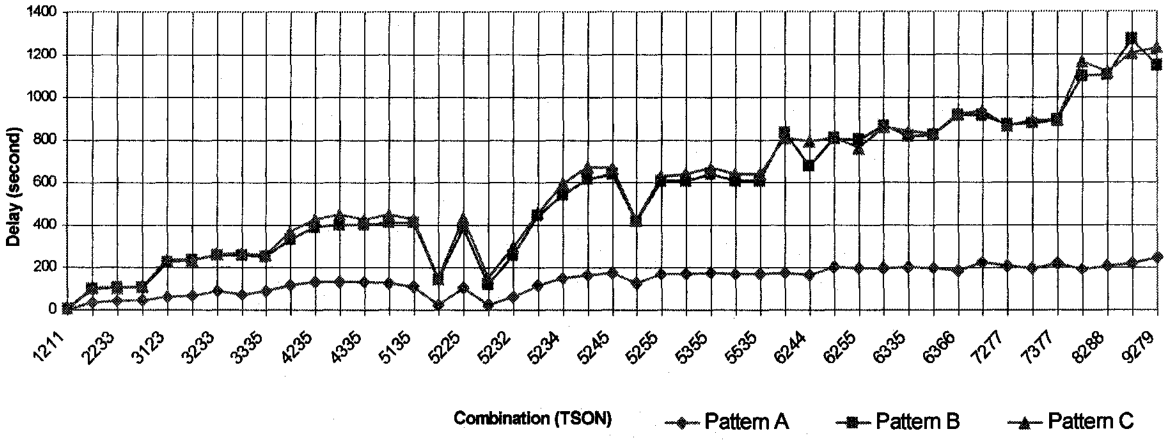


Figure 4.19 Delays of Different Site Layout Patterns

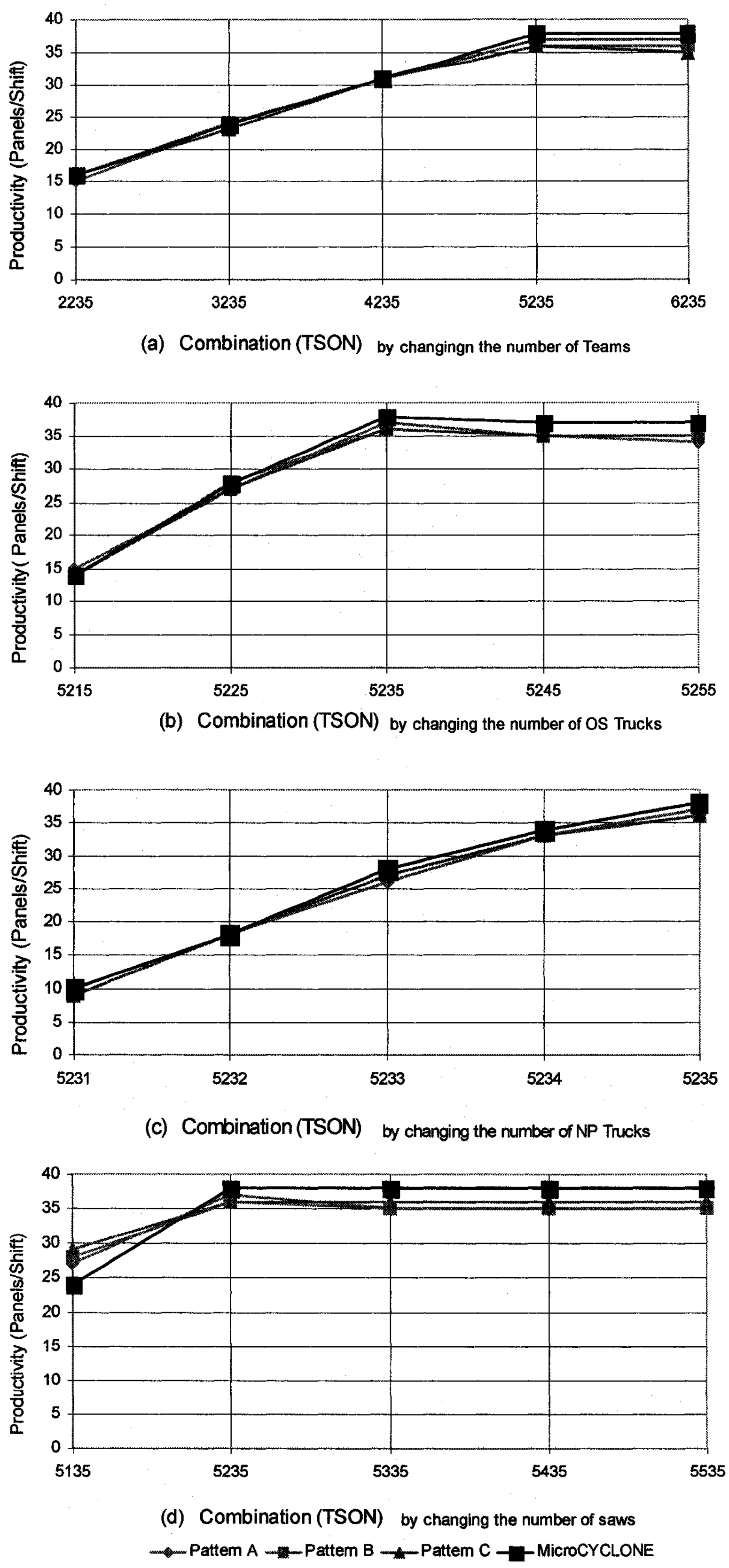
(4) Sensitivity Analysis Based on Resource Combinations

The sensitivity analysis is performed by changing the number of each resource for each site layout pattern. For the MicroCYCLONE model (Figure 4.3), the generated number of resource combinations is 1296 by increasing the number of each resource from 1 to 6. When the number of teams was increased above 6, the productivity was not beyond what was achieved using 5 teams. For the cell-based simulation model, the results for the three patterns and resource combinations (Figure 4.20) demonstrate the same trends.

More resource combinations are applied to the two simulation methods for further comparing the productivity results between the MicroCYCLONE model and the cell-based simulation model. As shown in Figure 4.21, based on the 45 combinations of the three site layout patterns, it is found that: (1) As mentioned in Section 4.2.1, the contractor was able to replace eight panels per crew per night with the resource combination of TSON 2223. The simulation results show that the total number of installed panels of this resource combination is 16, which matches the productivity of the real project; (2) The results of both modeling techniques are similar, which indicates that the cell-based simulation model predicts the performance of different resource combinations and site layout patterns properly; (3) In most combinations, the productivity of MicroCYCLONE model is higher than that of the cell-based simulation method by about 3-5% (1-2 panels). This indicates that the influence of space constraints is small when the speed of trucks is constant at 10 km/h; and (4) In most combinations, the productivity of Pattern-A is a little higher than the other two patterns. This is because Pattern-A has less delays resulting from spatial conflicts as we mentioned above.

(5) Optimal resource combination

The results of sensitivity analysis can be used for selecting the optimal resource combination roughly. Those that have a higher productivity and lower cost are selected since they dominate the other combinations that have a similar or lower productivity with a higher cost/panel. The unit cost ($/Panel) is an important index that links the productivity with cost. As shown in Figure 4.22, for MicroCYCLONE model, the lowest unit cost can be easily found at the combination TSON 5235 with the value $746.45. However, for the cell-based simulation method, the selection of the optimal resource combination should consider different site layout patterns. As discussed above, Pattern-A has a higher productivity than the other two patterns. This difference will be reflected in deciding the optimal combination. In Figure 4.22, for Pattern-B and Pattern-C, the lowest unit cost is $783.87 at the combination TSON 4235, while the unit cost of the combination TSON 5235 is $787.50; for Pattern-A, the lowest unit cost is $766.22 at the combination TSON 5235, while the unit cost of the combination TSON 4235 is still $783.87. Since Pattern-A has a lower unit cost and less spatial delays (Figure 4.12), the optimal resource combination can be roughly confirmed as the combination TSON 5235 with Pattern-A, from the perspective of productivity and cost. Through the above analysis, it can be seen that: (1) The unit costs of different resource combinations using the cell-based simulation method are higher than those using MicroCYCLONE due to their lower productivities; and (2) For the cell-based simulation method, the optimal resource combination of a specific the site layout pattern may be different from that of other patterns. This result again reflects the impact of the site layout patterns.



(a) Combination (TSON) by changingn the number of Teams

(b) Combination (TSON) by changing the number of OS Trucks

(c) Combination (TSON) by changing the number of NP Trucks

(d) Combination (TSON) by changing the number of saws

Figure 4.20 Sensitivity Analysis Using MircoCYCLONE and Cell-DEVS

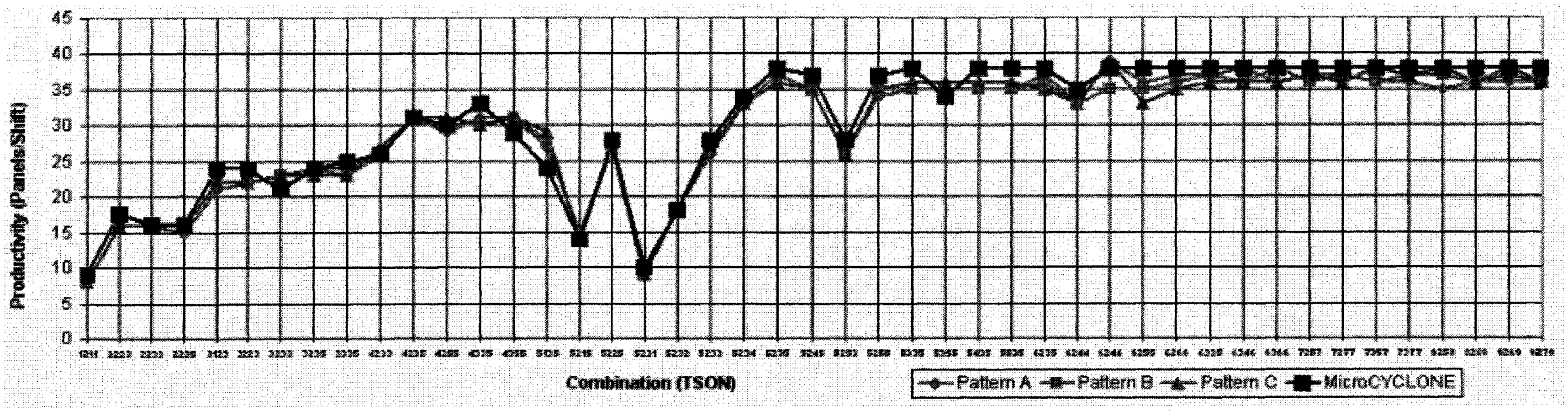


Figure 4.21 Comparison of Productivity between MicroCYCLONE and Cell-based Simulation Method

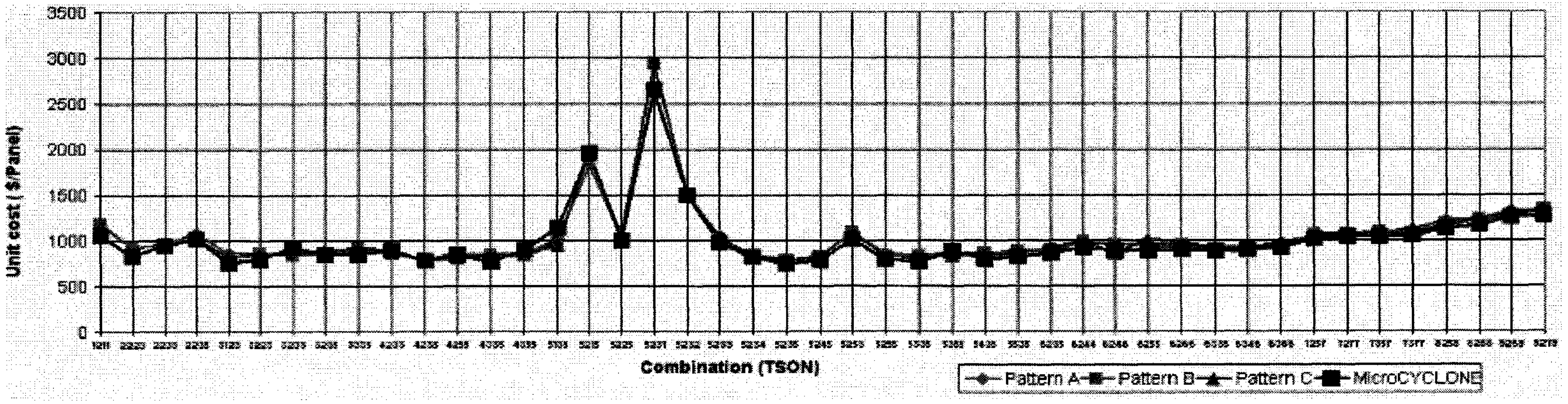


Figure 4.22 Comparison of Unit Cost between MicroCYCLONE and Cell-based Simulation Model

(6) Effects of Reduced Turning Speed

In the above sections, the simulation runs based on the assumption that trucks travel at a constant speed of one cell one second (10km/h). This is not realistic, In fact, when a truck is changing direction, it has to slow down and longer delays should be taken into account. The reduced moving speed may help to examine the effect of the extended delays. Nine combinations were selected and Pattern-B (Figure 4.12(b)) for this test. Figure 4.23 shows the comparison of productivity at average speed (10km/h) and a slower turning speed (1km/h) only when a truck is changing its direction and the other conditions were not changed. The productivity of slower turning speed is lower than that of average speed. The productivity rate dropped to 87% with TSON 4235 and 89% with TSON 5235, which indicates that when considering the changeable moving speed, the effects should not be ignored in some cases.

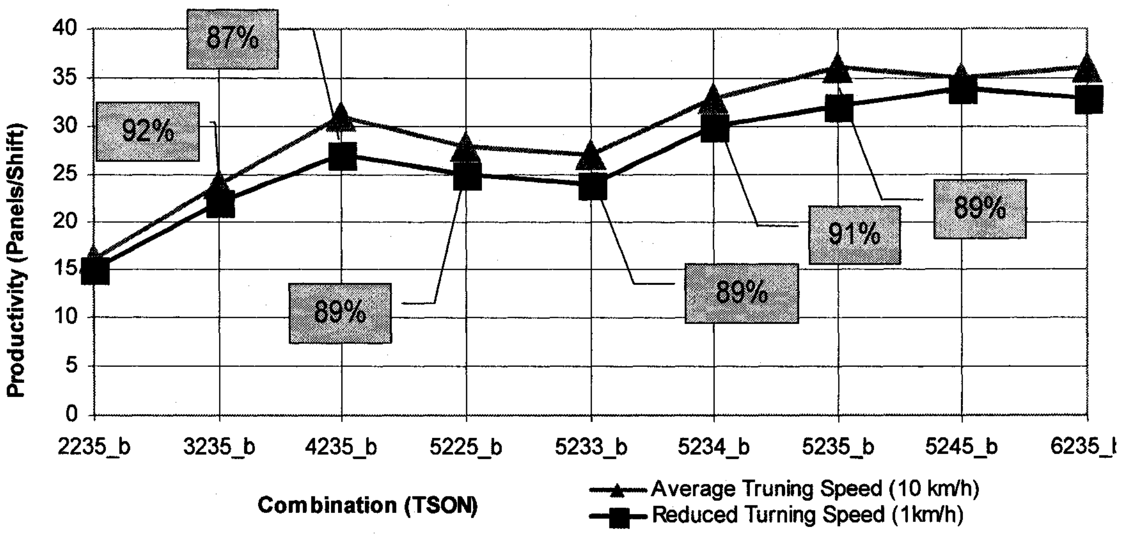


Figure 4.23 Effects of Slower Turning Speed on Productivity

4.3 CASE STUDY II – Analysis of Effect of Work Zones on Traffic Flow

4.3.1 Introduction

The cell-based simulation method is a generic technique that could be used for broad domains where spatial constraints are crucial to the project. In order to demonstrate the applicability of the proposed cell-based simulation to different types of construction projects, another case study about the traffic flow during asphalt pavement construction is discussed in this section. Every year, numerous road rehabilitation projects spread in Montreal. The picture in Figure 4.24 was taken in front of the Engineering, Computer Science and Visual Arts Complex in the summer of 2006.

Figure 4.24 Asphalt Pavement Project

There are some challenges in modeling paving operations. Paving operations often require one or more lanes closure in order to perform the operations. As a result, many closure scenarios, such as daytime closure, nighttime closure, one-lane closure or one-way closure, should be considered. On the other hand, the paving operation will impact the traffic flow. The effects of the closure strategies on traffic flow under different closure scenarios should be compared. In modern urban centers, analyzing the interference between construction operations and traffic flow has great significance to transportation agencies and construction managers.

4.3.2 Developing the Cell-based Simulation Model

In this case study, it is assumed that the paving operations are being done on only one side of a six-lane street and the closure strategies influence only one direction traffic as shown in Figure 4.24. Therefore, the observed section can be simplified as a section of a three-lane one-way street. Based on the methodology dicussed in Chapter 3, a cell-based simulation model is developed as shown in Figure 4.25.

In Figure 4.25, *Generator* generates cars and trucks according to the current traffic volume and the truck/car rate. *Sender* receives the generated cars or trucks, counts and then distributes them randomly to the three *Queues* through the three output ports. Since the emphasis of this case study is to investigate the influence of the asphalt pavement construction on the traffic flow under different lane-closure scenarios, it is necessary to monitor the entry of each lane and decide if a vehicle can enter the observed road section. As a result, three *Control Units* are used to check the first four cells of each lane to determine if a car or a truck can come in. For example, if the first three cells are empty and the fourth cell is not being occupied by a stopping vehicle, a truck can enter.

Otherwise, the truck cannot enter into the observed section because of the traffic jam will be stored in the three corresponding *Queues*, waiting for the next possible chance. If the condition is satisfied, the vehicle will be sent into the *Road* from the *Queue*. *Road* is a Cell-DEVS model where the interferences between construction operations and traffic occur. *Road* represents a section of the road with three lanes. When a vehicle is sent to a lane of the observed section, it will check if this lane has been reserved for construction. If not, the vehicle will go ahead; otherwise, it has to make a lane-changing maneuver. Unlike the case study of the Jacques Cartier Bridge rehabilitation project in which resources remain within the system, in this case study, the vehicles traverse the observed section only once and then exit, forming a noncyclic system.

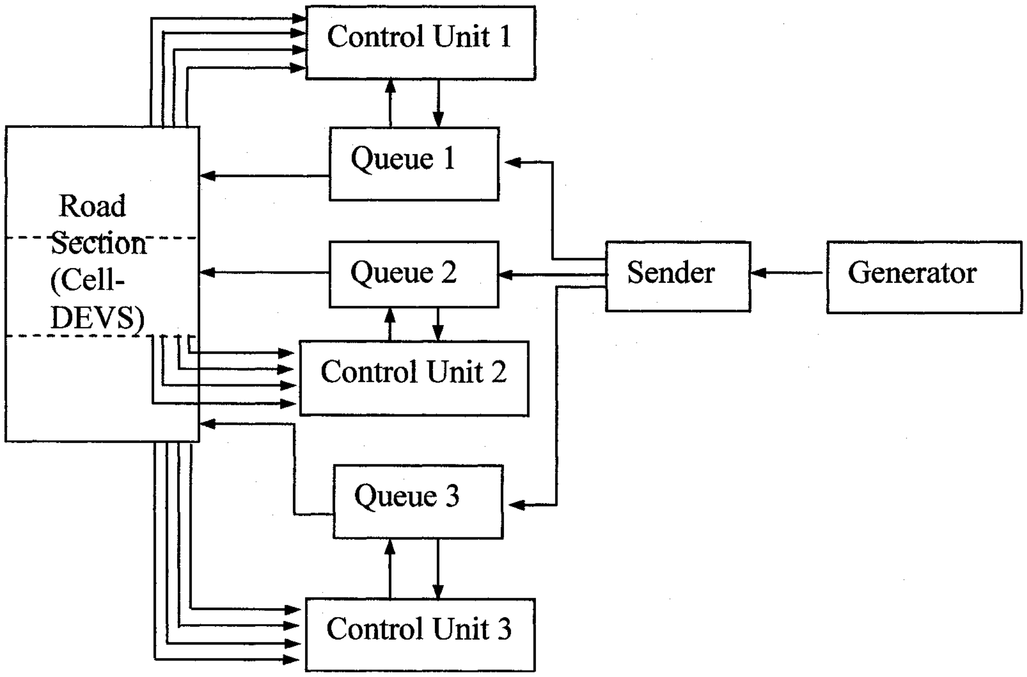


Figure 4.25 Simulation Model of the Traffic Flow under Asphalt Pavement Construction

4.3.3 Deciding the Size of Cells, the Layers and Codes for Each Cell-DEVS Model

The size of cells is taken as 3m x 3m (which can be adjusted according to the accuracy requirments). The surface of the observed road section is represented by 60 x 3 cells. A car is represented by one cell and a truck is represented by 3 x 1 cells (equivalent to 3 vehicles). This case study also adopted the three-layer structure used in Case Study I for the *Road* model as shown in Figures 4.6: the *Occupancy* layer, the *Control* layer and the *ID* layer. Predefined codes are assigned to the cells of each layer as shown in Table 4.6.

Table 4.6 Codes Used in the three Layers of Case Study II

Model		Code	Descriptions
Road	*Occupancy Layer*	0	Empty cell
		1	A car
		2	A truck
	Control Layer	0	Empty cell
		1	Moving north
		2	Moving south
		4	Moving west
		5	Static
		6	Temporarily stops
	ID Layer	0	Empty cell
		1	A car
		2 - 4	A truck

4.3.4 Defining the Zones of Each Cell-DEVS Model

The reserved zone for the paving operations could occupy different numbers of the lane. Therefore, the performance of alternative closure strategies could be compared. In this case study, the paving oprations could reserve one or two lanes and the other lanes could be used for traffic. Accordingly, the surface of the *Road* are devided into two zones to adopt different rules for the work zones and traffic lanes, respetively, as shown in Figure 4.26.

4.3.5 Developing Rules for the Cell-DEVS Model and Developing DEVS models

In the case study of the Jacques Cartier Bridge rehabilitation project, a basic "Modeling Element Library" has been set up. It includes sets of rules that can perform specific actions such as moving straight or turning north and south. It also provides several DEVS models such as *Queues* that can be used for other applications. These advantages of the elements in the library are used to implement this case study. The priority rules are defined as the following:

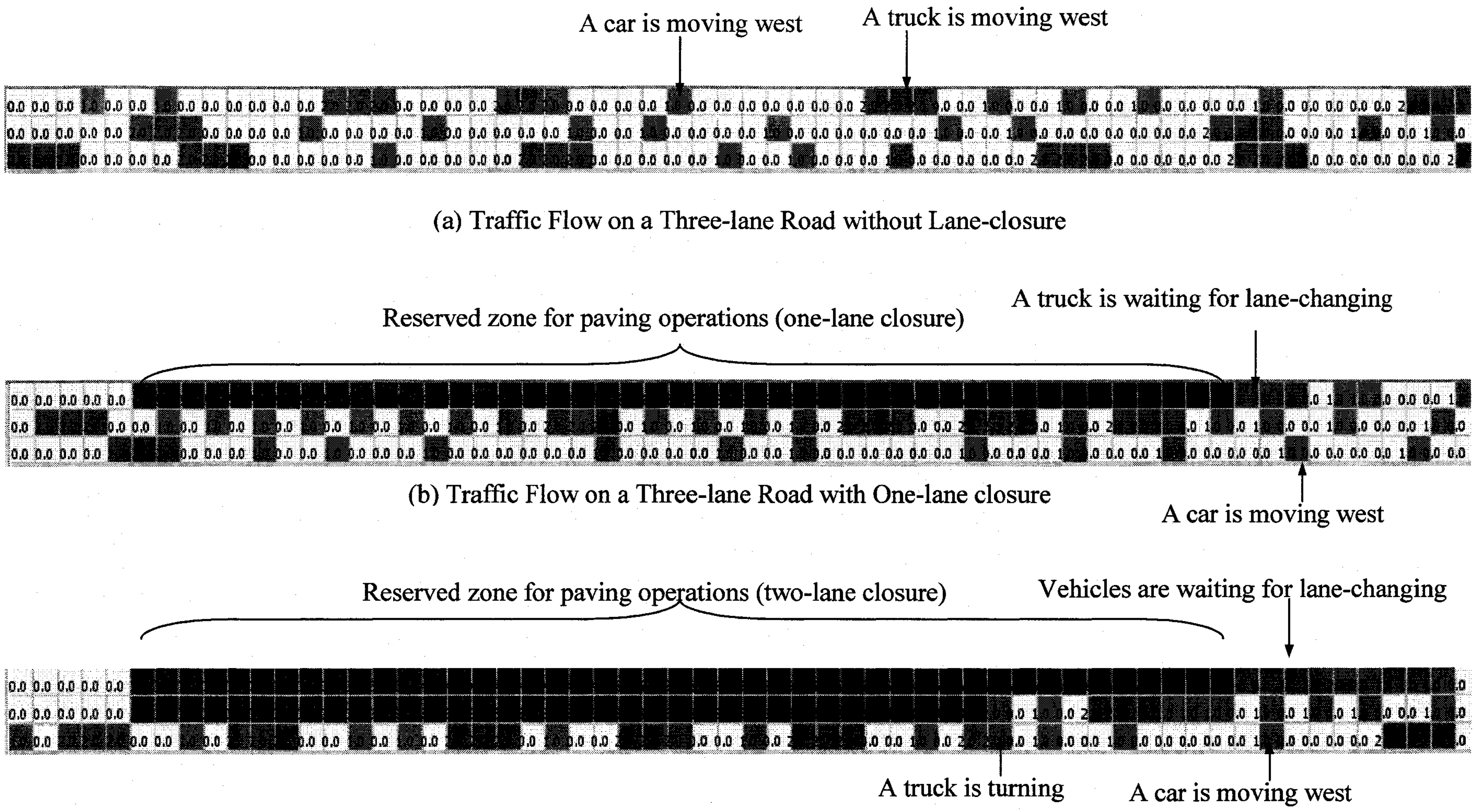
- When two vehicles are moving on the same lane, the interval between them is at least one cell; otherwise, the latter has to stop temporarily, giving the priority to the former one. When vehicles are stopping temporarily, no space is kept between them.
- When two vehicles are moving on different lanes (e.g., the vehicles in Figures 4.26(b) and (c)), the vehicle on the lower lane has the priority. If the vehicle on the upper lane is ahead of the vehicle on the lower lane but they have an overlap, the vehicle on the upper lane has to stop temporarily.

- When two vehicles are moving on different lanes, if the vehicle on the upper lane is ahead of the vehicle on the lower lane without any overlap, the vehicle on the upper lane has the priority.

- When a vehicle is taking the priority to turn, other mobile objects that may cause spatial conflicts with this vehicle have to keep their positions until the vehicle leaves.

4.3.6 Visualizing the Cell-DEVS Model

Figure 4.26 shows the animation snapshots of the *Occupancy* layer of the *Road* model with different lane-closure scenarios. Through animation, the user can identify the interferences between the operations and traffic flow — how vehicles made turns and were jammed because of the lane closure.


A car is moving west
A truck is moving west
Reserved zone for paving operations (one-lane closure)
A truck is waiting for lane-changing
A car is moving west
Reserved zone for paving operations (two-lane closure)
Vehicles are waiting for lane-changing
A truck is turning
A car is moving west

(a) Traffic Flow on a Three-lane Road without Lane-closure

(b) Traffic Flow on a Three-lane Road with One-lane closure

(c) Traffic Flow on a Three-lane Road with Two-lane closure

Figure 4.26 Snapshots of the Animation of the Traffic Flow under different Lane-closure Scenarios

4.3.7 Simulation Results and Discussions

For comparison, three scenarios are considered — traffic flow on a three-lane road without lane closure, with one-lane closure and two-lane closure, as shown in Figure 4.26. Suppose the traffic volume is 5040 equivalent vehicles/hour, 2520 equivalent vehicles/hour and 1680 equivalent vehicles/hour, and these vehicles are sent to one of the three lanes randomly. The number of trucks accounts for 20% of the total vehicles that are coming to this observed road section. The simulation period is eight minutes. The average speed is assumed as 10 km/hour. Table 4.7 shows the simulation results.

Table 4.7 Simulation Results of Traffic Flow under Different Lane-closure Scenarios

(a) Under the Traffic Volume of 5040 equivalent vehicles/hour

Number of closed lanes	Total coming vehicles	Total vehicles in the Queues	Total vehicles passed through the Road Section	Total vehicles on the Road Section
0	673 (100%)	287 (43%)	335 (50%)	52 (8%)
1	673 (100%)	314 (47%)	309 (46%)	50 (7%)
2	673 (100%)	427 (63%)	204 (30%)	42 (6%)

(b) Under the Traffic Volume of 2520 equivalent vehicles/hour

Number of closed lanes	Total coming vehicles	Total vehicles in the Queues	Total vehicles passed through the Road Section	Total vehicles on the Road Section
0	337 (100%)	69 (20%)	232 (69%)	36 (11%)
1	337 (100%)	69 (20%)	235 (70%)	34 (10%)
2	337 (100%)	84 (25%)	217 (64%)	36 (11%)

(c) Under the Traffic Volume of 1680 equivalent vehicles/hour

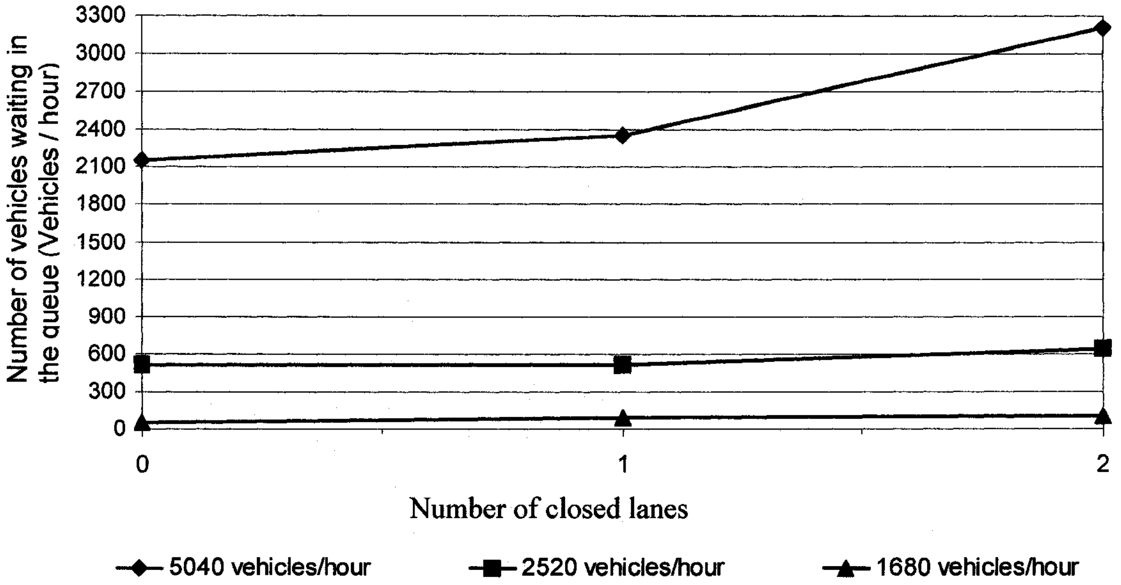
Number of closed lanes	Total coming vehicles	Total vehicles in the Queues	Total vehicles passed through the Road Section	Total vehicles on the Road Section
0	225 (100%)	7 (3%)	190 (84%)	28 (12%)
1	225 (100%)	11 (5%)	188 (83%)	27 (12%)
2	225 (100%)	14 (6%)	181 (80%)	31 (14%)

In Table 4.7 (a), when the traffic volume is 5040 equivalent vehicles/hour, there are 287 (43%) of the total vehicles waiting in the *Queues* and 335 (50%) vehicles passed through the observed road section. Under the one-lane closure scenario, the number of vehicles which passed through the observed section reduced to 309 (46%). This is not a significant drop. The results mean that under this traffic volume, the one-lane closure scenario will not influence the traffic flow greatly. Under the two-lane closure scenario, the vehicles passed through the observed section dropped to 204 (30%) and 427 (63%) of the total vehicles are waiting in the *Queues*. The results indicate that under this the traffic volume, the two-lane closure scenario has a greater influence on the traffic flow.

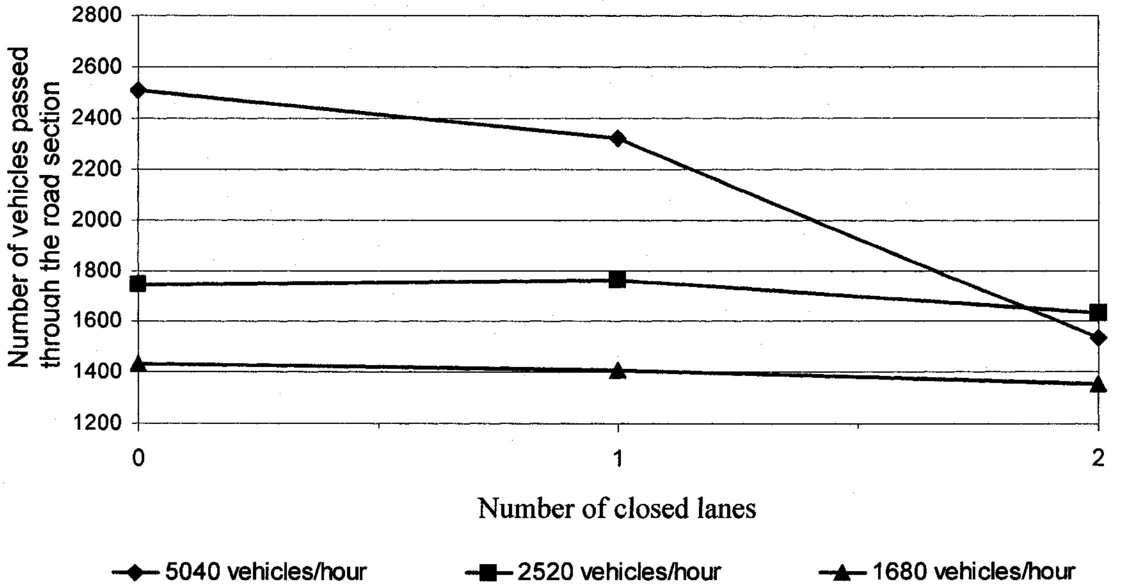
In Table 4.7 (b), when the traffic volume is 2520 equivalent vehicles/hour, the results of one-lane closure are almost the same as those without lane-closure. They both have about 20% of the vehicles in the *Queues* and 70% of the vehicles which passed through the observed section. The results mean that under this the traffic volume, the one-lane closure scenario has little influence on the traffic flow. Under the two-lane closure scenario, the number of vehicles passed through the observed section dropped to 217 (64%). The results indicate that under this the traffic volume, the two-lane closure scenario does not have significant influence on the traffic flow.

In Table 4.7 (c), when the traffic volume is 1680 equivalent vehicles/hour, the results show there is little difference among the three scenarios, meaning that under this the traffic volume, the paving operations has little influence on the traffic flow under either closure scenario.

Based on the simulation results, the trendlines under the three traffic volumes could be drawn as shown in Figure 4.27 to provide a better illustration for the construction manager. Since the objective of this case study is to show the applicability of the proposed simulation method, only three traffic volumes and three scenarios were tested. For a more complete analysis, more scenarios and traffic volumes might be needed. This case study helps the user to predict the effects of the paving operations on the traffic flow and then to adjust the lane-closure strategies according to the traffic volume.



(a) Comparison of the Number of Vehicles Waiting in the Queue under Different Lane-closure Scenarios



(b) Comparison of the Number of Vehicles Passed through the Road Section under Different Lane-closure Scenarios

Figure 4.27 Comparisons of Traffic Flow under Different Lane-closure Scenarios

4.4 SUMMARY AND CONCLUSIONS

This chapter described the implementation of the proposed method discussed in Chapter 3. Two case studies were presented to demonstrate the applicability and effectiveness of the cell-based simulation method for construction operations.

Following the proposed method, a new cell-based simulation system was developed for the case study of the Jacques Cartier Bridge rehabilitation project. Since in cell-based simulation the surface of the construction site was represented by cells, the positions and the states of objects could be shown explicitly, which provides the foundation to analyze the simulation results from multiple perspectives. According to the objectives and requirements, three major delays caused by spatial conflicts were identified and quantified. Different site layouts and resource combinations were examined and the spatial conflicts can be detected and resolved during simulation. The simulation results show that: (1) The site layouts might influence the delays resulting from spatial conflicts, the optimal resource combination and the productivity. These results illustrate the importance of considering site layouts in simulation; (2) In most cases, the productivity of the cell-based simulation was lower than that of MicroCYCLONE because of the consideration of spatial constraints; and (3) The effect of changeable moving speed could be analyzed in cell-based simulation. In the case study of Jacques Cartier Bridge rehabilitation project, the simulation results showed that the impact of additional delays due to the slower turning speed might lead to a more than 10% drop on the productivity.

The case study of the traffic flow analysis considering work zones presented a non-cyclic simulation system. Through the evaluation of the conditions on the observed road section and the control of the vehicles' movement, the effects of construction operations on the traffic flow under different scenarios were investigated.

The two case studies show the advantages and the applicability of the proposed cell-based simulation method. In addition, sets of rules and models have been defined which can be reused in other simulation studies.

CHAPTER 5

SUMMARY, CONTRIBUTIONS AND FUTURE WORK

5.1 SUMMARY AND CONCLUSIONS

This research is an attempt to apply the cell-based method to construction simulation, trying to incorporate site layout planning, simulation and animation, to detect and resolve spatial conflicts, and to analyze the effects of spatial constraints.

The literature review of the construction simulation tools, the site layout planning approaches as well as the animation techniques has revealed some improved implementations in this field. During the past three decades, these tools opened the door for the construction simulation research and pushed it forward constantly. Nevertheless, the fact that workspace was not considered and spatial conflicts were not detected/resolved in construction simulation limited the application of these tools. Explicit representation of space will contribute to a more realistic simulation result. In addition, an adequate space representation can bridge the gap between site layout planning and simulation, allowing examination of the various site layout patterns in simulation. Moreover, a conflict-free and uninterrupted animation, which is helpful in site layout planning and visualizing the activity sequence, may be achieved by introducing the space into construction simulation.

This research proposed a cell-based construction simulation method using the Cell-DEVS technique. In this method, the spatial constraints, site layout patterns and resource allocations can be explicitly represented. Consequently, spatial conflicts can be detected and resolved. The effects of spatial constraints can be analyzed based on site layout patterns, resource combinations and changeable moving speeds. Two case studies were used to implement and demonstrate the advantages and applicability of this method.

The discussions and comparisons in this research demonstrate some distinguishing characteristics of the cell-based simulation method:

(1) It is a general-purpose simulation tool for broad domains and can be used to simulate complex construction operations, especially when spatial constraints are crucial to the project.

(2) Space is represented explicitly so that the occupation of the workspace and other spatial information about the construction environment can be understood more easily than the abstract symbols in simulation networks.

(3) Physical construction resources can be allocated considering different site layout patterns as if the user were planning a real project.

(4) The programmability and the capability of defining rules to control entities make the cell-based simulation systems more flexible and capable of detecting and resolving spatial conflicts during the simulation.

(5) An uninterrupted conflict-free animation provides a better means for the practitioners to check the simulation results and compare various resolutions.

(6) More information (occupancy, moving direction and ID, etc.) is integrated in the cell-based model, which makes it easy to identify and trace a specific object. Furthermore, the detailed output helps to verify the outcome of the system.

(7) Space can be considered as an independent construction resource.

5.2 CONTRIBUTIONS

The contributions of this research are grouped into the following areas:

(1) It is the first time to propose a construction simulation method which integrates site layout planning, simulation and animation. This cell-based, space-involved and conflict-resolving simulation method will help the decision makers or the field personnel to easily understand, predict and compare the performances of construction operations with different resource combinations considering site layout patterns. This innovative simulation method provides multiple perspectives to examine the construction operations in which space is explicitly represented and serves as an independent construction resource.

(2) Designed a general process to apply the cell-based method to the construction simulation. It consists of the Pre-processing, Main-processing and Post-processing phases, involving system modeling, preparing data input, simulation running, verification and validation, animation and result analysis. Through the three phases, a whole cell-based simulation system can be developed and implemented.

(3) Applied two case studies to demonstrate the advantages and applicability of the proposed method. A new cell-based model was developed for the case study of the Jacques Cartier Bridge rehabilitation project to consider different site layouts and investigate the effects of spatial constraints. This model overcame the limitations of the previous study. It took into account all necessary construction resources and used a group of cells to represent objects. Through a mechanism designed for the trucks to find their corresponding teams, different site layout patterns could be compared. The improvement in this research makes the model more practical and closer to the real conditions of the construction operations. The case study of the traffic flow analysis considering work zones presented a non-cyclic simulation system. Through the evaluation of the conditions on the observed road section and the control of the vehicles' movement, the effects of construction operations on the traffic flow under different scenarios were investigated.

(4) Defined sets of rules for the Cell-DEVS models and related DEVS models to establish basic modules of the "Modeling Elements Library" that can be reused in the future. More than 500 rules are coded to control the movement of objects and explore the possibilities of detecting and resolving spatial conflicts. These rules have been tested over 200 successful tests for different resource combinations and site layout patterns. Based on the experience, the modeling concepts of "master-slave" cells and the "macros" were created, which greatly reduce the efforts of developing rules.

(5) Analyzed the simulation results according to different requirements. The effects of spatial constraints were quantified and the results were analyzed from the aspects of site layout patterns, resource combinations and the changeable moving speed. The simulation result shows that: (1) Pattern-A has less spatial constraints than the other two patterns; (2) The productivities of the cell-based method are lower than those of MicroCYCLONE by 3-5% (1-2 panels) when the object's moving speed was constant; and (3) When the turning speed dropped from 10 km/h to 1km/h, the productivity might drop over 10% for some resource combinations (Figure 4.23). Through the comparisons with those of MicroCYCLONE, the results of the cell-based method reflect the effects of the spatial constraints. This difference illustrates that in some cases the impact of spatial constraints is significant and should not be neglected.

Although the manner and degree to which spatial constraints affect the construction operations may be different from one project to another, the perspectives to analyze simulation results provided in this chapter could be used for other projects to analyze the impact of spatio-temporal constraints.

5.3 LIMITATIONS AND FUTURE WORK

While pursuing this research, several limitations have been identified related to the requirements and the performance of the developed method. In order to enhance the capabilities of the cell-based simulation method, the following points should be explored in the future research.

(1) More case studies should be investigated to further validate this cell-based method, and to accumulate more experiences in building models and developing rules, making this method more practical and easier for more applications.

(2) In the cell-based method, the simulator needs to scan each rule and related cells to check their activation conditions. When the amount of rules or cells is very large and the compositions of the entities such as resources and logical dependencies to start activities are complex, the simulation will become slow. A solution is Parallel CD++ (Glinsky and Wainer 2006; Liu and Wainer 2007). Using this parallel computing algorithm, the performance of the cell-based method can considerably speed up by distributing calculation over multiple possessors.

(3) More issues, such as site layout patterns and the size of cells, have to be considered in the proposed method. With the increase of the amount of entities, modeling may become complicated. A graphical modeling interface or other techniques that can facilitate modeling and data input should be developed.

(4) Based on the DEVSView (Khan et al. 2005), a 3D animation could be realized. On the other hand, since the cell-based simulation method provides a very detailed data output, the potential to link VITASCOPE could be also investigated.

REFERENCES

AbouRizk, S.M., Ruwanpura, J.Y., Er, K.C., and Fernando, S. (1999). Special Purpose Simulation Template for Utility Tunnel Construction, Proceedings of the 1999 Winter Simulation Conference, pp. 948-955.

Ahmad, S., and Simonovic, S.P. (2004). Spatial System Dynamics: New Approach for Simulation of Water Resources Systems, Journal of Computing in Civil Engineering, ASCE, Vol. 18, No. 4, pp. 331-340.

Anumba, C., and Bishop, G. (1997). Importance of Safety Considerations in Site Layout and Organization." Canadian Journal of Civil Engineering, Vol. 24 No. 2, pp. 229–236.

Balci, O. (1988). The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-level Languages, Proceedings of the 1988 Winter Simulation Conference, San Diego, California, pp. 287-295.

Banks, J., and Carson, J.S. (1984). Discrete-Event System Simulation, Prentice-Hall, Inc., New Jersey.

Booch, G. (1991). Object Oriented Design with Applications, Benjamin/Cummings Publishing Company, Inc., Redwood City, California.

Caron, F., Marchet, G. and Perego, A. (2000). Layout Design in Manual Picking Systems: A Simulation Approach, Integrated Manufacturing Systems, Vol. 11No. 2, pp. 94-104.

Chang, D. Y., and Carr, R. I. (1987). RESQUE: A Resource Oriented Simulation System for multiple Resource Constrained Processes, Proceedings of the PMI Seminar / Symposium, Milwaukee, Wisconsin, pp. 4-19.

Cheng, M.Y, and Yang, S.C. (2001). GIS-Based Cost Estimates Integrating with Material Layout Planning, Journal of Construction Engineering and Management, ASCE, Vol. 127, No. 4, pp. 291-299.

Cor, H. (1998). Using Simulation to Quantify the Impacts of Changes in Construction Work, Master thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Davidson, A. and Wainer, G. (2006). ATLAS: a Specification Language for Traffic Modelling and Simulation, Simulation, Practice and Experience, Elsevier. Vol. 14, No. 3, pp. 317-337.

Dawood, N., and Marasini, R. (2001). Stockyard Layout Management for Precast Concrete Products Using Simulation, Proceedings of CIB-W78 International Concrete Conference, IT in Construction in Africa, Mpumalanga, South Africa, Vol. 26, pp. 1-13.

Elbeltagi, E., Hegazy, T., and Eldosouky, A. (2004). Dynamic Layout of Construction Temporary Facilities Considering Safety, Journal of Construction Engineering and Management, ASCE, Vol. 130, No. 4, pp. 534-541.

El-Rayes, K. and Khalafallah, A. (2005). Trade-off between Safety and Cost in Planning Construction Site Layout, Journal of Construction Engineering and Management, ASCE, Vol. 131, No. 11, pp. 1186-1195.

Giambiasi, N. and Wainer, G. (2005) Cell-DEVS/GDEVS for Complex Continuous Systems. SIMULATION: Transactions of the Society for Modeling and Simulation International, Vol. 81, No. 2, pp. 137-151.

Gonzalez-Quevedo, A. A., AbouRizk, S. M., Iseley, D. T., and Halpin, D. W. (2002). Comparison of Two Simulation Methodologies in Construction, Journal of Construction Engineering and Management, ASCE, Vol. 119, No. 3, pp. 573-589.

Guo, S. (2002). Identification and Resolution of Workspace Conflicts in Building Construction, Journal of Construction Engineering and Management, ASCE, Vol. 128, No. 4, pp. 287-295.

Glinsky, E. and Wainer, G. (2006). New Parallel simulation techniques of DEVS and Cell-DEVS in CD++, Proceedings of the 38th IEEE/SCS Annual Simulation Symposium, Huntsville, AL. 2006.

Hajjar, D., and AbouRizk, S.M. (1998). Modeling and Analysis of Aggregate Production Operations, Journal of Construction Engineering and Management, ASCE, Vol. 124, No. 5, pp. 390-401.

Hajjar, D., and AbouRizk, S.M. (1999). Simphony: an Environment for Building Special Purpose Construction Simulation Tools, Proceedings of the 1999 Winter Simulation Conference, pp. 998-1006.

Hajjar, D., and AbouRizk, S.M. (2002). Unified Modeling Methodology for Construction Simulation, Journal of Construction Engineering and Management, ASCE, Vol. 128, No. 2, pp. 174-185.

Halpin, D.W. (1973). An Investigation of the Use of Simulation Networks for Modeling Construction Operations, PhD dissertation, University of Illinois at Urbana-Champain, Illinois.

Halpin, D.W. (1977). CYCLONE: Method for Modeling of Job Site Processes, Journal of the Construction Division, ASCE, Vol. 103, No. 3, pp. 489-499.

Halpin, D.W., and Riggs, L.S. (1992). Planning and Analysis of Construction Operations, Wiley Interscience, New York, 1992.

Hamiani, A. (1989). Knowledge Representation for the Site Layout Problem, Proceedings of the 6th Conference on Computing in Civil Engineering, New York, pp. 283-289.

Harmanani, H., Zouein, P., and Hajar, A. (2000). An Evolutionary Algorithm for Solving the Geometrically Constrained Site Layout Problem, Proceedings of 8th ASCE International Conference on Computing in Civil and Building Engineering, (ICCCBE-VIII), Stanford University, 1442–1449.

Hegazy, T. M., and Elbeltagi, E. (1999). EvoSite: Evolution-Based Model for Site Layout Planning, Journal of Computing in Civil Engineering, Vol. 13, No. 3, pp. 198-206.

Henriksen, J. O. (1998). Window-based Animation with PROOF, Proceedings of the 1998 Winter Simulation Conference, San, Diego, California, pp. 241-247.

Hooper, J. W. (1986). Strategy Related Characteristics of Discrete-event Languages and Models, Simulation, Vol. 46, No. 4, pp. 153-159.

Huang, R. Y., and Halpin, D. W. (1994). Graphical-based Method for Transient Evaluation of Constructin Operations, Journal of Construction Engineering and Management, ASCE, Vol. 121, No. 2, pp. 222-229.

Ioannou, P. G., and Martinez, J. C. (1996). Animation of Complex Construction Simulation Models, Proceedings of 3rd Congress on Computing in Civil Engineering, ASCE, Reston, VA, pp. 620-626.

Ioannou, P. G., and Martinez, J. C. (1996a). Comparison of Construction Alternatives Using Matched Simulation Experiments, Journal of Construction Engineering and Management, ASCE, Vol. 122, No. 3, pp. 222-229.

Islier, A. A. (1998). A Genetic Algorithm Approach for Multiple Criteria Facility Layout Design, International Journal of Production Research, Vol. 36, No. 6, 1549-1569.

Jen, H. (2005). Web-based Construction Process Simulation Framework, Doctoral dissertation, School of Civil Engineering, Purdue University, W. Lafayette, Indiana.

Kamat, V. R., and Martinez, J. C. (2001). Visualizing Simulated Construction Operations in 3D, Journal of Computing in Civil Engineering, ASCE, Vol. 15, No. 4, pp. 329-337.

Kamat, V. R. (2003). VITASCOPE – Extensible and Scalable 3D Visualization of Simulated Construction Operations, Doctoral dissertation, Department of Civil Engineering, Virginia Polytechnic Institute and State University, Blacksburg, Virginia.

Kamat, V.R., and Martinez, J.C. (2004). Practical 3D Animation of Multiply Articulated Construction Equipment, Proceedings of the 2004 Winter Simulation Conference, pp.1229-1237.

Kamat, V.R., and Martinez, J. C. (2006). Interactive Collision Detection in Three-Dimensional Visualizations of Simulated Construction Operations, Engineering with Computers, Springer, London, UK.

Khan, A., Venhola, W., Wainer, G. and Jemtrud. M. (2005). Advanced DEVS Model Visualization. Proceedings of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation 2005, Paris, France. 2005. VI.

Khoury, H. M., Kamat V. R., and Ioannou P. G. (2006). Simulation and Visualization of Air-side Operations at Detroit Metropolitan Airport, Proceedings of 2006 Winter Simulation Conference, Monterey, California, pp. 2029-2038.

Kumara, S. R. T., Kashyap, R. L., and Moodie, C. L. (1988). Application of Expert Systems and Pattern Recognition Methodologies to Facilities Layout Planning, International Journal Production Research, Vol. 26, No. 5, pp. 905-930.

Lee, E. B., Harvey, J. T., and Thomas, D. (2005a). Integrated Design / Construction / Operations Analysis for Fast-Track Urban Freeway Reconstruction, Journal of Construction Engineering and Management, ASCE, Vol. 131, No. 12, pp. 1283-1291.

Lee, E. B., and Ibbs, C. W. (2005b). Computer Simulation Model: Construction analysis for Pavement Rehabilitation Strategies, Journal of Construction Engineering and Management, ASCE, Vol. 131, No. 4, pp. 449-458.

Lee, E. B., Thomas, D., and Bloomberg, L. (2005b). Planning Urban Highway Reconstruction with Traffic Demand Affected construction Schedule, Journal of Construction Engineering and Management, ASCE, Vol. 131, No. 10, pp. 752-761.

Li, H., and Love, P.E.D. (1998). Site-level Facilities Layout Using Genetic Algorithms, Journal of Computing in Civil Engineering, ASCE, Vol. 12, No. 4, pp. 227-231.

Liu, L. Y. and Ioannou, P.G. (1992). Graphical Object-Oriented Simulation System for Construction Process Modeling, Proceedings of the Eighth Conference on Computing in Civil Engineering, ASCE, Dallas, Texas, pp. 1139-1146.

Liu, Q. and Wainer, G. (2007) Improving CD++ Parallel Simulation Engine, Proceedings of the 39th IEEE/SCS Annual Simulation Symposium. Norfolk, VA, USA. 2007.

Lluch, J. F., and Halpin, D. W. (1981). Analysis of Construction Operations Using Microcomputers, Journal of the Construction Division, ASCE, Vol. 108, No. CO1, pp. 129-145.

Lutz, J. D., Halpin, D. W., and Wilson, J. R. (1994). Simulation of Learning Development in Repetitive Construction. Journal of Construction Engineering and Management, ASCE, Vol. 120, No. 4, pp. 753-773.

Madhoun, R., and Wainer, G. (2007). Studying the Impact of Web-Services Implementation of Distributed Simulation of DEVS and Cell-DEVS Models, Proceedings of 2007 DEVS Integrative M&S Symposium (DEVS'07), Norfolk, Virginia, USA.

Marasini, R., and Dawood, N. (2002). Simulation Modeling and Optimization of Stockyard Layout for Precast Concrete Products, Proceedings of the 2002 Winter Simulation Conference, pp. 1731-1736.

Martinez, J.C. and P.G. Ioannou (1994). General Purpose Simulation with Stroboscope, Proceedings of the 2004 Winter Simulation Conference, Lake Buena Vista, Florida, pp. 1159-1166.

Martinez, J. C., and Ioannou, P. G. (1995). Advantages of the Activity Scanning Approach in the Modeling of Complex Construction Processes, Proceedings of 1995 Winter Simulation Conference, San Diego, California, pp. 1024-1031.

Martinez, J. C. (1996). STROBOSCOPE: State and Resource Based Simulation of Construction Processes, Doctoral dissertation, University of Michigan, Ann Arbor, Michigan.

Martinez, J.C. (1998). EarthMover – Simulation Tool for Earthwork Planning, Proceedings of the 1998 CIB W78 Conference, Department of Construction Management and Economics, Royal Institute of Technology, Stockholm, Sweden.

Martinez, J. C., and Ioannou, P. (1999). General–purpose Systems for Effective Construction Simulation, Journal of Construction Engineering and Management, ASCE, Vol. 125, No. 4, pp. 265-276.

Marzouk, M. and Moselhi, O. (2003). Object-Oriented Simulation Model for Earthmoving Operations, Journal of Construction Engineering and Management, ASCE, Vol. 129, No. 2, pp. 173-181.

Mawdesley, M.J., Al-jibouri, S.H., and Yang H. (2002). Genetic Algorithms for Construction Site Layout in Project Planning, Journal of Construction Engineering and Management, ASCE, Vol. 128, No. 5, pp. 418-426.

Muzy, A., E. Innocenti, A. A, Santucci, J. F., Santoni, P. and Hill, D. (2005) Modelling and Simulation of Ecological Propagation Processes: Application to Fire Spread, Environmental Modeling and Software, Vol. 20, pp. 827-42.

Naji, K. K. (1997). The Development of A Virtual Environment for Simulating Equipment-based Operations in Real-time Object-oriented Systems, Doctoral dissertation, University of Florida, Gainesville, Florida.

Nassar, K., Thabet, W., and Beliveau, Y. (2003). Simulation of Asphalt Paving Operations under Lane Closure Conditions, Automation in Construction, Vol. 12, pp. 527-541.

Odeh, A. M., Tommelein I. D., and Carr R. I. (1992). Knowledge-based Simulation of Construction Plans, Proceedings of the Eighth Conference on Computing in Civil Engineering, ASCE, Dallas, Texas, pp. 1042-1049.

Oloufa, A. A. (1992). Feedback Mechanisms for Operational Simulation, Journal of Computing in Civil Engineering, ASCE, Vol. 6, No. 2, pp. 161-177.

Oloufa, A. A. (1993). Modeling Operational Activities in Object-Oriented Simulation, Journal of Computing in Civil Engineering, ASCE, Vol. 7, No. 1, pp. 94-106.

Palaniappan, S., Sawhney, A. and Sarjoughian, H. S. (2006). Application of the DEVS Framework in Construction Simulation, Proceedings of 2006 Winter Simulation Conference, Monterey, California, pp. 2077-2086.

Pang, H., Zhang, C., and Hammad, A. (2006). Sensitivity Analysis of Construction Simulation Using Cell-DEVS and MicroCYCLONE, Proceedings of 2006 Winter Simulation Conference, Monterey, California, pp. 2021-2028.

Pang, H., Zhang, C., and Hammad, A. (2007). Sensitivity Analysis of Construction Simulation Considering Site Layout Patterns, International Workshop on Computing in Civil Engineering, Pittsburgh, Pennsylvania, July 2007.

Paulson, B. C., Jr. (1987). Interactive Graphics for Simulating Construction Operations, Journal of the Construction Division, ASCE, Vol. 104, No. 1, pp. 69-76.

Pidd, M. (1998). Computer Simulation in Management Science, Wiley, Chichester, U.K.

PJCCI 2004: The official website of Jacques Cartier and Champlain Bridges Incorporated [online]. Available from http://www.pjcci.ca/ [cited on August 18, 2004].

Randhawa, S.U. and Sharoff, R. (1995), Simulation Base Design Evaluation of Unit Load Automated Storage/ Retrieval Systems, Computers and Industrial Engineering, Vol. 28, No. 1, pp. 71-79.

Riley, D. R., and Sanvido, V. E. (1995). Patterns of Construction-Space Use in Multistory Buildings, Journal of Construction Engineering and Management, ASCE, Vol. 121, No. 4, pp. 464-473.

Riley, D. R., and Sanvido, V. E. (1997). Space Planning Method for Multistory Building Construction, Journal of Construction Engineering and Management, ASCE, Vol, 123, No. 2, pp. 171-180.

Shang, H. and Wainer, G. (2005). A Model of Virus Spreading Using Cell-DEVS, Proceedings of the International Conference on Computational Science, Atlanta, GA.

Shang, H. and Wainer, G. (2006). A Simulation Algorithm for Dynamic Structure DEVS Modeling, Proceedings of 2006 Winter Simulation Conference, Monterey, California, pp. 815-822.

Shi, J. (1999). Activity-based Construction (ABC) Modeling and Simulation Method, Journal of Construction Engineering and Management, ASCE, Vol, 125, No. 5, pp. 354-360.

Shi, J., and Zhang, H.(1999). Iconic Animation of Construction Simulation, Proceedings of the 1999 Winter Simulation Conference, Vol, 2, pp. 992-997.

Shi, J. (2000). Object-oriented Technology for Enhancing Activity-based Modeling Functionality, Proceedings of the 2000 Winter Simulation Conference, pp. 1938-1944.

Tam, C.M., Tong, T.K.L., and Chan, W.K.W. (2001). Genetic Algorithm for Optimizing Supply Locations around Tower Crane, Journal of Construction Engineering and Management, ASCE, Vol. 127, No. 4, pp. 315-321.

Tartaro, M.L., Torres, C., and Wainer, G. (2001). Defining Models of Urban Traffic Using the TSC Tool, Proceedings of the 2001 Winter Simulation Conference, Arlington, VA, USA, pp. 1056-1063.

Tawfik, H. M., and Fernando, T. (2001). A Simulation Environment for Construction Site Planning, Proceedings of 5th International Conference, Information Visualization, London, U.K., pp. 199-204.

Thabet, W. Y., and Beliveau, Y. J. (1994). Modeling Work Space to Schedule Repetitive Floors in Multistory Buildings, Journal of Construction Engineering and Management, ASCE, Vol. 120, No. 1, pp. 96-116.

Tocher, K. T. (1963). The Art of Simulation, English Universities Press, London.

Tommelein, I. D., Levitt, R. E., and Hayes-Roth, B. (1992). Site-layout Modeling: How Can Artificial Intelligence Help? Journal of Construction Engineering and Management, ASCE, Vol. 118, No. 3, pp. 594-611.

Touran, A., and Asai, T. (1987). Simulation of Tunneling Operations, Journal of Construction Engineering and Management, ASCE, Vol. 113, No. 4, pp. 554-568.

Vanegas, J. A., and Halpin, D. W. (1993). Simulation Technologies for Planning Heavy Construction processes, Journal of Construction Engineering and Management, ASCE, Vol. 119, No. 2, pp. 336-354.

VITASCOPE 2003 [online]. Available from http://pathfinder.engin.umich.edu [cited on November 12, 2005].

Wainer, G. (1998). Discrete-events Cellular Models with Explicit Delays, Doctoral dissertation, Université d'Aix-Marseille III.

Wainer, G., and Giambiasi. N. (2001). Application of the Cell-DEVS Paradigm for Cell Spaces Modeling and Simulation, SIMULATION, Transactions of the SCS, Vol. 76, No. 1, pp. 22-39.

Wainer, G. (2002). CD++: a Toolkit to Define Discrete-event Models, Software, Practice and Experience, Wiley. Vol. 32, No.3, pp. 1261-1306.

Wainer, G., and Giambiasi, N. (2002). N-Dimensional Cell-DEVS, Discrete Events Systems: Theory and Applications, Kluwer. Vol. 12, No. 1, pp. 135-157.

Wainer, G. (2006). Applying Cell-DEVS Methodology for Modeling the Environment, SIMULATION, Transactions of the SCS, Vol. 82, No. 10, pp. 635-660.

Wandha, L.C. (2000), Optimizing Deployment of Shiploaders at Bulk Export Terminal, Journal of Waterway, Port, Coastal and Ocean Engineering , Nov/Dec, 2000, pp.297-304.

Wolfram, S. (1986). Theory and Application of Cellular Automata, Vol. 1, Advances Series on Complex Systems, World Scientific, Singapore.

Yeh, I.-C. (1995). Construction-site Layout Using Annealed Neural Network, Journal of Computing in Civil Engineering, ASCE, Vol. 9, No. 3, pp. 201-208.

Zaki, A. R., and Mailhot, G. (2003). Deck Reconstruction of Jacques Cartier Bridge Using Precast Prestressed High Performance Concrete Panels, PCI Journal.

Zayed, T. M. and Halpin, D. (2001). Simulation of Concrete Batch Plant Production, Journal of Construction Engineering and Management, ASCE, Vol. 127, No. 2, pp. 132-141.

Zeigler, B.P. (1976). Theory of Modelling and Simulation, Wiley-Interscience Publication.

Zhang, C., Hammad, A., Zayed, T.M., and Wainer, G., (2005). Representation and Analysis of Spatial Resources in construction Simulation, Proceedings of 2005 Winter Simulation Conference, Orlando, Florida, pp. 1541-1548.

Zhang, C., Hammad, A., Zayed, T.M., Wainer, G., and Pang, H. (2007). Cell-based Representation and Analysis of Spatial Resources in Construction Simulation, Automation in Construction, Vol. 16, No. 4, pp. 436-448.

Zhang, H., Shi, J.J., and Tam, C. M. (2002). Iconic Animation for Activity-based Construction Simulation, Journal of Computing in Civil Engineering, ASCE, Vol. 16, No. 3, pp. 157-164.

Zhang, H., Tam, C.M., and Li, H. (2005). Activity Object-Oriented Simulation Strategy for Modeling Construction Operations, Journal of Computing in Civil Engineering, ASCE, Vol. 19, No. 3, pp. 313-322.

Zhong, D., Li, J., Zhu, H., and Song, L. (2004). Geographic Information System-Based Visual Simulation Methodology and Its Application in Concrete Dam Construction Processes, Journal of Construction Engineering and Management, ASCE, Vol. 130, No. 5, pp. 742-750.

Zouein, P. P., and Tommelein, I. D. (1999). Dynamic Layout planning Using a Hybrid Incremental Solution Method, Journal of Construction Engineering and Management, ASCE, Vol. 125, No. 6, pp. 400-408.

Zouein, P. P., Harmanani, H. and Hajar, A. (2002). Genetic Algorithm for Solving Site Layout Problem with Unequal-Size and Constrained Facilities. Journal of Computing in Civil Engineering, ASCE, Vol. 16, No. 2, pp. 143-151.

APPENDIX A: Source Codes of (.ma) File

```
#include(Project_finallongdelay_10.inc)
[top]

components : bridge dump plant qnew qold controller deque@Deque

out : boutid1 boutid2 boutid3 boutid4

Link : sendold@controller start@qold
Link : repdone@controller newstart@qold
Link : sendnew@controller start@qnew
Link : sendnew2@controller newstart@qnew
Link : qoldoutc@qold qoldin@controller
Link : qnewoutc@qnew qnewin@controller
Link : boutid1@bridge coninb1@controller
Link : boutid2@bridge coninb2@controller
Link : boutid3@bridge coninb3@controller
Link : boutid4@bridge coninb4@controller
Link : senddump@controller btdin@dump
Link : sendplant@controller btpin@plant
Link : dumpout@dump dtbin@qold
Link : plantout@plant ptbin@qnew
Link : qoldoutb@qold qoldin@deque
Link : qnewoutb@qnew qnewin@deque
Link : out@deque bin@bridge
Link : boutid1@bridge boutid1
Link : boutid2@bridge boutid2
Link : boutid3@bridge boutid3
Link : boutid4@bridge boutid4


[dump]
components : dumpser@Dumpser transbtd@Transbtd dumpque@Dumpque

in : btdin
out : dumpout
Link : btdin btdin@transbtd
Link : out@transbtd dumpqin@dumpque
Link : dumpqout@dumpque dumpserin@dumpser
Link : done@dumpser ready@dumpque
Link : dumpout@dumpser dumpout

[plant]
components : plantser@Plantser transbtp@Transbtp plantque@Plantque
in : btpin
out : plantout
Link : btpin btpin@transbtp
Link : out@transbtp plantqin@plantque
Link : plantqout@plantque plantserin@plantser
Link : done@plantser ready@plantque
Link : plantout@plantser plantout

[qold]
components : queold@Queueo transdtb@Transdtb
```

```
in : start newstart dtbin
out : qoldoutc qoldoutb
Link : start start@queold
Link : newstart newstart@queold
Link : dtbin dtbin@transdtb
Link : out@transdtb truckin@queold
Link : qoldoutb@queold qoldoutb
Link : qoldoutc@queold qoldoutc


[qnew]
components : quenew@Queuen transptb@Transptb

in : start newstart ptbin
out : qnewoutc qnewoutb
Link : start start@quenew
Link : newstart newstart@quenew
Link : ptbin ptbin@transptb
Link : out@transptb truckin@quenew
Link : qnewoutb@quenew qnewoutb
Link : qnewoutc@quenew qnewoutc

[controller]
components : cu@ControlUnit reposition@Reposition saw@Saw

in : coninb1 coninb2 coninb3 coninb4 qoldin qnewin
out : sendold repdone sendnew sendnew2 senddump sendplant
Link : coninb1 coninb1@cu
Link : coninb2 coninb2@cu
Link : coninb3 coninb3@cu
Link : coninb4 coninb4@cu
Link : qoldin qoldin@cu
Link : qnewin qnewin@cu
Link : sawout@cu in@saw
Link : out@saw sawin@cu
Link : repout@cu in@reposition
Link : out@reposition repin@cu
Link : sendold@cu sendold
Link : repdone@cu repdone
Link : sendnew@cu sendnew
Link : sendnew2@cu sendnew2
Link : senddump@cu senddump
Link : sendplant@cu sendplant

[Dumpser]
distribution : normal
mean : 0.300
deviation : 0.120

[Plantser]
distribution : normal
mean : 0.840
deviation : 0.120

[Transptb]
distribution : normal
mean : 0.420
```

```
deviation : 0.060


[Transdtb]
distribution : normal
mean : 0.300
deviation : 0.120

[Transbtd]
distribution : normal
mean : 0.420
deviation : 0.120

[Transbtp]
distribution : normal
mean : 0.300
deviation : 0.060


[Deque]
delay : 00:00:00:010

[Reposition]
distribution : normal
mean : 1.080
deviation : 0.180


[Saw]
distribution : normal
mean : 1.080
deviation : 0.180

[cu]
num_old : 4
num_emp : 0
num_plate : 400
num_saw : 2
OS_ID : 4
NP_ID : 4
delay : 00:00:00:010

[Dumpque]

OS_NUM : 5

[Plantque]
NP_NUM : 5

[bridge]
type : cell
dim : (6,200,3)
delay : inertial
defaultDelayTime : 1
border : nowrapped
```

```
neighbors : bridge(0,-1,2)      bridge(0,0,2)    bridge(0,1,2)
            bridge(0,2,2)
neighbors : bridge(0,-1,1)      bridge(0,0,1)    bridge(0,1,1)
            bridge(0,2,1)
neighbors : bridge(0,-1,0)      bridge(0,0,0)    bridge(0,1,0)
            bridge(0,2,0)
neighbors : bridge(0,-1,-1)     bridge(0,0,-1)   bridge(0,1,-1)
            bridge(0,2,-1)
neighbors : bridge(0,-1,-2)     bridge(0,0,-2)   bridge(0,1,-2)
neighbors : bridge(0,-2,1)      bridge(0,-2,0)   bridge(0,-3,0)
            bridge(0,-2,2)  bridge(0,3,2)
neighbors : bridge(0,-5,0)      bridge(0,-5,1)   bridge(0,-5,-1)
            bridge(0,-4,-1)
neighbors : bridge(0,4,0)       bridge(0,3,0)    bridge(0,-2,-1)
            bridge(0,-3,-1)
neighbors : bridge(0,-4,0)      bridge(0,-4,1)   bridge(0,-3,0)
            bridge(0,-3,1)
neighbors : bridge(0,-6,0)      bridge(0,-6,1)   bridge(0,-6,-1)
            bridge(0,-5,-1)
neighbors : bridge(0,-6,-1)     bridge(0,-7,-1)  bridge(0,-8,-1)
            bridge(0,3,1)
neighbors : bridge(0,4,1)       bridge(0,4,2)
neighbors : bridge(1,-1,2)      bridge(1,0,2)    bridge(1,1,2)
            bridge(1,4,0)
neighbors : bridge(1,-1,-1)     bridge(1,0,-1)   bridge(1,1,-1)
            bridge(1,2,-1)
neighbors : bridge(1,3,1)       bridge(1,-3,0)   bridge(1,-4,1)
            bridge(1,-2,-1)
neighbors : bridge(1,-3,-1)     bridge(1,-4,-1)  bridge(1,-4,0)
            bridge(1,-5,0)
neighbors : bridge(1,-6,0)      bridge(1,3,-1)   bridge(1,3,0)
            bridge(1,4,-1)
neighbors : bridge(1,-5,-1)     bridge(1,-5,1)   bridge(1,-2,2)
            bridge(1,-2,0)
neighbors : bridge(1,-6,-1)     bridge(1,-6,1)   bridge(1,2,-2)
            bridge(1,1,-2)
neighbors : bridge(1,-2,1)      bridge(1,-3,1)   bridge(1,0,-2)
            bridge(1,-1,-2)
neighbors : bridge(1,2,2)       bridge(1,2,0)    bridge(1,3,-2)
            bridge(1,4,-2)
neighbors : bridge(1,-1,0)      bridge(1,0,0)    bridge(1,1,0)
            bridge(1,4,1)
neighbors : bridge(1,-1,1)      bridge(1,0,1)    bridge(1,1,1)
            bridge(1,2,1)
neighbors : bridge(1,3,2)       bridge(1,-7,-1)  bridge(1,-8,-1)
            bridge(1,4,2)
neighbors : bridge(-1,-1,2)     bridge(-1,0,2)    bridge(-1,1,2)
            bridge(-1,4,0)
neighbors : bridge(-1,-1,-1)    bridge(-1,0,-1)   bridge(-1,1,-1)
            bridge(-1,2,-1)
neighbors : bridge(-1,-2,1)     bridge(-1,-2,-1)  bridge(-1,-3,1)
            bridge(-1,-3,-1)
neighbors : bridge(-1,3,0)      bridge(-1,3,1)    bridge(-1,-3,0)
            bridge(-1,-4,-1)
neighbors : bridge(-1,-4,1)     bridge(-1,2,2)    bridge(-1,2,0)
            bridge(-1,-4,0)
neighbors : bridge(-1,-5,-1)    bridge(-1,-5,1)   bridge(-1,-5,0)
```

```
            bridge(-1,4,-1)
neighbors : bridge(-1,-6,-1)    bridge(-1,-6,1)   bridge(-1,-6,0)
            bridge(-1,-2,2)
neighbors : bridge(-1,3,-1)     bridge(-1,-2,0)   bridge(-1,2,-2)
            bridge(-1,1,-2)
neighbors : bridge(-1,0,-2)     bridge(-1,-1,-2)  bridge(-1,3,-2)
            bridge(-1,4,-2)
neighbors : bridge(-1,-1,0)     bridge(-1,0,0)    bridge(-1,1,0)
            bridge(-1,4,1)
neighbors : bridge(-1,-1,1)     bridge(-1,0,1)    bridge(-1,1,1)
            bridge(-1,2,1)
neighbors : bridge(-1,3,2)      bridge(-1,-7,-1)  bridge(-1,-8,-1)
            bridge(-1,4,2)

neighbors : bridge(2,-1,-1)   bridge(2,-1,1)     bridge(2,-2,1)
            bridge(2,-2,-1)
neighbors : bridge(2,-3,-1)   bridge(2,-3,1)     bridge(2,-4,-1)
            bridge(2,-4,1)
neighbors : bridge(2,-5,1)    bridge(2,-5,-1)    bridge(2,-6,1)
            bridge(2,-6,-1)
neighbors : bridge(2,-6,0)    bridge(2,-5,0)     bridge(2,-4,0)
            bridge(2,-3,0)
neighbors : bridge(2,4,0)     bridge(2,4,-1)     bridge(2,3,0)
            bridge(2,3,-1)
neighbors : bridge(2,2,0)     bridge(2,2,-1)     bridge(2,1,0)
            bridge(2,1,-1)
neighbors : bridge(2,-1,0)    bridge(2,0,-2)     bridge(2,0,-1)
            bridge(2,0,0)
neighbors : bridge(2,-2,0)    bridge(2,0,2)      bridge(2,1,2)
            bridge(2,0,1)
neighbors : bridge(2,1,1)     bridge(2,-2,-2)    bridge(2,-7,-1)
            bridge(2,-8,-1)
neighbors : bridge(2,-1,2)    bridge(2,2,2)      bridge(2,2,1)

neighbors : bridge(-2,4,0)    bridge(-2,4,-1)    bridge(-2,3,0)
            bridge(-2,3,-1)
neighbors : bridge(-2,2,0)    bridge(-2,2,-1)    bridge(-2,1,0)
            bridge(-2,1,-1)
neighbors : bridge(-2,-1,-1)  bridge(-2,-1,1)    bridge(-2,-1,0)
            bridge(-2,0,-2)
neighbors : bridge(-2,-2,-1)  bridge(-2,-2,1)    bridge(-2,-2,0)
            bridge(-2,0,-1)
neighbors : bridge(-2,-3,-1)  bridge(-2,-3,1)    bridge(-2,-3,0)
            bridge(-2,0,0)
neighbors : bridge(-2,-4,-1)  bridge(-2,-4,1)    bridge(-2,-4,0)
            bridge(-2,0,2)
neighbors : bridge(-2,-5,-1)  bridge(-2,-5,1)    bridge(-2,-5,0)
            bridge(-2,1,2)
neighbors : bridge(-2,-6,-1)  bridge(-2,-6,1)    bridge(-2,-6,0)
            bridge(-2,0,1)
neighbors : bridge(-2,1,1)    bridge(-2,-2,-2)   bridge(-2,-7,-1)
            bridge(-2,-8,-1)
neighbors : bridge(-2,-1,2)   bridge(-2,2,2)     bridge(-2,2,1)

neighbors : bridge(3,0,0)    bridge(3,0,1)     bridge(3,4,0)
            bridge(3,4,-1)
neighbors : bridge(3,3,0)    bridge(3,3,-1)    bridge(3,2,0)
```

```
            bridge(3,2,-1)
neighbors : bridge(3,1,0)    bridge(3,1,-1)    bridge(3,0,1)
            bridge(3,0,-1)
neighbors : bridge(3,-1,-1)    bridge(3,-1,1)   bridge(3,-1,0)
            bridge(3,0,2)
neighbors : bridge(3,-2,-1)    bridge(3,-2,1)   bridge(3,-2,0)
            bridge(3,1,1)
neighbors : bridge(3,-3,-1)    bridge(3,-3,1)   bridge(3,-3,0)
            bridge(3,1,2)
neighbors : bridge(3,-4,-1)    bridge(3,-4,1)   bridge(3,-4,0)
neighbors : bridge(3,-5,-1)    bridge(3,-5,1)   bridge(3,-5,0)
neighbors : bridge(3,-6,-1)    bridge(3,-6,1)   bridge(3,-6,0)
neighbors : bridge(-3,0,0)    bridge(-3,0,1)     bridge(-3,4,0)
            bridge(-3,4,-1)
neighbors : bridge(-3,3,0)    bridge(-3,3,-1)    bridge(-3,2,0)
            bridge(-3,2,-1)
neighbors : bridge(-3,1,0)    bridge(-3,1,-1)    bridge(-3,0,1)
            bridge(-3,0,-1)
neighbors : bridge(-3,-1,-1)    bridge(-3,-1,1)   bridge(-3,-1,0)
            bridge(-3,0,2)
neighbors : bridge(-3,-2,-1)    bridge(-3,-2,1)   bridge(-3,-2,0)
            bridge(-3,1,1)
neighbors : bridge(-3,-3,-1)    bridge(-3,-3,1)   bridge(-3,-3,0)
            bridge(-3,1,2)
neighbors : bridge(-3,-4,-1)    bridge(-3,-4,1)   bridge(-3,-4,0)
neighbors : bridge(-3,-5,-1)    bridge(-3,-5,1)   bridge(-3,-5,0)
neighbors : bridge(-3,-6,-1)    bridge(-3,-6,1)   bridge(-3,-6,0)


neighbors : bridge(-4,0,1)      bridge(-4,0,2)
neighbors : bridge(4,0,1)      bridge(4,0,2)

initialvalue : 0

initialCellsValue : hongtest_5235_b.val

in : bin
out : boutid1
out : boutid2
out : boutid3
out : boutid4
link : bin in@bridge(2,199,0)
link : bin in@bridge(2,199,1)
link : bin in@bridge(2,199,2)

link : out@bridge(1,0,2) boutid1
link : out@bridge(2,0,2) boutid2
link : out@bridge(3,0,2) boutid3
link : out@bridge(4,0,2) boutid4

localtransition : move-rule
zone : move-rule1 { (0,0,0)..(2,199,0) }
zone : move-rule2 { (3,0,0)..(5,199,0) }
zone : move-check-rule1 { (0,0,1)..(2,199,1) }
zone : move-check-rule2 { (3,0,1)..(5,199,1) }
zone : move-id-rule { (0,0,2)..(5,199,2) }
portInTransition : in@bridge(2,199,0) gentruck-rule
portInTransition : in@bridge(2,199,1) gendir-rule
portInTransition : in@bridge(2,199,2) genid-rule

[move-rule]

rule : 0 1 { t }

[move-rule1]

#Macro(MoveRule_WaitingDelay)
#Macro(MoveRule_LongDelay)
#Macro(MoveRule1_WorkTeamAction)
#Macro(MoveRule1_AfterLoading)
#Macro(MoveRule_BeforeLoading)
#Macro(MoveRule_AtLoading)
#Macro(MoveRule_SouthAndNorth)
#Macro(MoveRule1_GenTruck)
#Macro(Reset)

[move-rule2]

#Macro(MoveRule_WaitingDelay)
#Macro(MoveRule_LongDelay)
#Macro(MoveRule2_WorkTeamAction)
#Macro(MoveRule2_AfterLoading)
#Macro(MoveRule_BeforeLoading)
#Macro(MoveRule_AtLoading)
#Macro(MoveRule_SouthAndNorth)
#Macro(Reset)

[move-check-rule1]

#Macro(MoveCheckRule_WorkTeamAction)
#Macro(MoveCheckRule_LongDelay)
#Macro(MoveCheckRule1_WaitingDelay_12)
#Macro(MoveCheckRule1_WaitingDelay_13)
#Macro(MoveCheckRule1_WaitingDelay_14)
#Macro(MoveCheckRule1_WaitingDelay_23)
#Macro(MoveCheckRule1_WaitingDelay_24)
#Macro(MoveCheckRule1_WaitingDelay_21)
#Macro(MoveCheckRule_WaitingDelay_end)
#Macro(MoveCheckRule1_WaitingDelay_SameLane1)
#Macro(MoveCheckRule1_WaitingDelay_SameLane2)
#Macro(MoveCheckRule_AtLoading)
#Macro(MoveCheckRule1_GoAroundTeam)
#Macro(MoveCheckRule_AfterLoading)
#Macro(MoveCheckRule_BeforeLoading)
#Macro(MoveCheckRule_WaitingDelay_3)
#Macro(MoveCheckRule1_GenTruck)
#Macro(Reset)

[move-check-rule2]

#Macro(MoveCheckRule_WorkTeamAction)
#Macro(MoveCheckRule_LongDelay)
#Macro(MoveCheckRule2_WaitingDelay_43)
#Macro(MoveCheckRule2_WaitingDelay_42)
```

```
#Macro(MoveCheckRule2_WaitingDelay_41)
#Macro(MoveCheckRule2_WaitingDelay_32)
#Macro(MoveCheckRule2_WaitingDelay_31)
#Macro(MoveCheckRule2_WaitingDelay_34)
#Macro(MoveCheckRule_WaitingDelay_end)
#Macro(MoveCheckRule2_WaitingDelay_SameLane1)
#Macro(MoveCheckRule2_WaitingDelay_SameLane2)
#Macro(MoveCheckRule_AtLoading)
#Macro(MoveCheckRule2_GoAroundTeam)
#Macro(MoveCheckRule_AfterLoading)
#Macro(MoveCheckRule_BeforeLoading)
#Macro(MoveCheckRule_WaitingDelay_3)
#Macro(Reset)

[gendir-rule]

#Macro(Gendir_Rule)

[gentruck-rule]

#Macro(Gentruck_Rule)

[genid-rule]

#Macro(Genid_Rule)
[move-id-rule]
#Macro(MoveIdRule_WaitingDelay)
#Macro(MoveIdRule_LongDelay)
#Macro(MoveIdRule_BeforeLoading)
#Macro(MoveIdRule_AtLoading)
#Macro(MoveIdRule_AfterLoading)
#Macro(MoveIdRule_WorkTeamAction)
#Macro(MoveIdRule_SouthAndNorth)
#Macro(MoveIdRule_GenTruckID)
#Macro(Reset)

[sendid-rule]
#Macro(Sendid_Rule)

[sendtruck-rule]
#Macro(SendTruck_Rule)
```

APPENDIX B: Source Codes of (.inc) File

This file includes the rules for the *Bridge* model of the Jacques Cartier Bridge Rehabilitation Project. The following is only part of this file that shows the rules used for Zone-1 on the *Control* layer.

```
[move-check-rule1]

#BeginMacro(MoveCheckRule_WorkTeamAction)

rule : 5 1 { (0,0,0) = 0 and ((0,0,-1) = 2 or (0,0,-1) = 3 or (0,0,-1)
           = 5 or (0,0,-1) = 6) }
#EndMacro

#BeginMacro(MoveCheckRule_LongDelay)

rule : {(0,0,0)} 1 { (0,0,-1 ) = 12 }
rule : {(0,0,0)} 1 { (0,0,-1 ) = 13 }
rule : {(0,0,0)} 1 { (0,0,-1 ) = 14 }
rule : {(0,0,0)} 1 { (0,0,-1 ) = 15 }

#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_12)

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and (0,0,-1) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1  and (2,-1,-1) = 3
           and ((2,-1,1) - trunc((0,1,1)/4 ) = 91 ) and (((1,4,-1) != 0
           and (1,4,0) = 4 ) or ((1,0,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 2
           and (0,1,-1) = 1 and (2,-2,-1) = 3 and ((2,-2,1) -
           trunc((0,1,1)/4 ) = 91 ) and (((1,3,-1) != 0 and (1,3,0) = 4 )
           or ((1,-1,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 3
           and (0,1,-1) = 1  and (2,-3,-1) = 3 and ((2,-3,1) -
           trunc((0,1,1)/4 ) = 91 ) and (((1,2,-1) != 0 and (1,2,0) = 4 )
           or ((1,-2,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 0
           and (0,1,-1) = 1 and (2,-4,-1) = 3 and ((2,-4,1) -
           trunc((0,1,1)/4 - 1 ) = 91 ) and (((1,1,-1) != 0 and (1,1,0)
           = 4 ) or ((1,-3,-1) != 0  ))}

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and (0,0,-1) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (2,-1,-1) = 5
           and ((2,-1,1) - trunc((0,1,1)/4 ) = 81 ) and (((1,4,-1) != 0
           and (1,4,0) = 4 ) or ((1,0,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 2
           and (0,1,-1) = 8 and (2,-2,-1) = 5 and ((2,-2,1) -
          trunc((0,1,1)/4 ) = 81 ) and (((1,3,-1) != 0 and (1,3,0) = 4 )
          or ((1,-1,-1) != 0  )) }
rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 3
          and (0,1,-1) = 8 and (2,-3,-1) = 5 and ((2,-3,1) -
          trunc((0,1,1)/4 ) = 81 ) and (((1,2,-1) != 0 and (1,2,0) = 4 )
          or ((1,-2,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,1,0) = 4 and remainder((0,1,1),4) = 0
          and (0,1,-1) = 8  and (2,-4,-1) = 5 and ((2,-4,1) -
          trunc((0,1,1)/4 - 1 ) = 81 ) and (((1,1,-1) != 0 and (1,1,0)
          = 4 ) or  ((1,-3,-1) != 0  ))}

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 1
          and (0,0,-1) = 1 and (2,-1,-1) = 3 and ((2,-1,1) -
          trunc((0,0,1)/4 ) = 91 ) and (((1,3,-1) != 0 and (1,3,0) = 4 )
          or ((1,0,-1) != 0  )) }
rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 2
          and (0,0,-1) = 1 and (2,-2,-1) = 3 and ((2,-2,1) -
          trunc((0,0,1)/4 ) = 91 ) and (((1,2,-1) != 0 and (1,2,0) = 4 )
          or ((1,-1,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 3
          and (0,0,-1) = 1 and (2,-3,-1) = 3 and ((2,-3,1) -
          trunc((0,0,1)/4 ) = 91 ) and (((1,1,-1) != 0 and (1,1,0) = 4 )
          or ((1,-2,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 0
          and (0,0,-1) = 1 and (2,-4,-1) = 3 and ((2,-4,1) -
          trunc((0,0,1)/4 - 1 ) = 91 ) and (((1,0,-1) != 0 and (1,0,0)
          = 4 ) or  ((1,-3,-1) != 0  ))}

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 1
          and (0,0,-1) = 8 and (2,-1,-1) = 5 and ((2,-1,1) -
          trunc((0,0,1)/4 ) = 81 ) and (((1,3,-1) != 0 and (1,3,0) = 4 )
          or ((1,0,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 2
          and (0,0,-1) = 8 and (2,-2,-1) = 5 and ((2,-2,1) -
          trunc((0,0,1)/4 ) = 81 ) and (((1,2,-1) != 0 and (1,2,0) = 4 )
          or  ((1,-1,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 3
          and (0,0,-1) = 8  and (2,-3,-1) = 5 and ((2,-3,1) -
          trunc((0,0,1)/4 ) = 81 ) and (((1,1,-1) != 0 and (1,1,0) = 4 )
          or ((1,-2,-1) != 0  )) }

rule : 6 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 0
          and (0,0,-1) = 8 and (2,-4,-1) = 5 and ((2,-4,1) -
          trunc((0,0,1)/4 - 1 ) = 81 ) and (((1,0,-1) != 0 and (1,0,0)
          = 4 ) or  ((1,-3,-1) != 0  ))}

rule : 2 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 1
          and (0,0,-1) = 1 and (2,-1,-1) = 3 and ((2,-1,1) -
          trunc((0,0,1)/4 ) = 91 ) and (1,3,-1) = 0  and (1,0,-1) = 0  }
```

```
rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 2
           and (0,0,-1) = 1 and (2,-2,-1) = 3 and ((2,-2,1) -
           trunc((0,0,1)/4 ) = 91 ) and
                    (1,2,-1) = 0 and (1,-1,-1) = 0  }

rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 3
           and (0,0,-1) = 1 and (2,-3,-1) = 3 and ((2,-3,1) -
           trunc((0,0,1)/4 ) = 91 ) and
                    (1,1,-1) = 0 and (1,-2,-1) = 0 }

rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 0
           and (0,0,-1) = 1 and (2,-4,-1) = 3 and ((2,-4,1) -
           trunc((0,0,1)/4 - 1 ) = 91 ) and
                    (1,0,-1) = 0 and (1,-3,-1) = 0 }

rule : 2 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 1
           and (0,0,-1) = 8 and (2,-1,-1) = 5 and ((2,-1,1) -
           trunc((0,0,1)/4 ) = 81 ) and
                    (1,3,-1) = 0 and (1,0,-1) = 0  }


rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 2
           and (0,0,-1) = 8 and (2,-2,-1) = 5 and ((2,-2,1) -
           trunc((0,0,1)/4 ) = 81 ) and
                    (1,2,-1) = 0 and (1,-1,-1) = 0  }

rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 3
           and (0,0,-1) = 8 and (2,-3,-1) = 5 and ((2,-3,1) -
           trunc((0,0,1)/4 ) = 81 ) and
                    (1,1,-1) = 0 and (1,-2,-1) = 0 }

rule : 4 1 {cellpos(0)= 1 and (0,0,0) = 6 and remainder((0,0,1),4) = 0
           and (0,0,-1) = 8 and (2,-4,-1) = 5 and ((2,-4,1) -
           trunc((0,0,1)/4 - 1 ) = 81 ) and
                    (1,0,-1) = 0 and (1,-3,-1) = 0 }

#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_14)

rule : 6 1 { cellPos(0) = 1 and (0,0,-1) = 0 and (0,1,0) = 4 and
           remainder((0,1,1),4) = 1 and (0,1,1) < 91 and (0,-1,-1) = 2
           and (-1,-1,-1) = 2 and (1,-1,-1) = 2 and (2,-1,0) = 5 and
           cellPos(1) != 0 and
             (((2,-4,-1) = 10 or (2,-4,-1) = 11) or
           ((3,4,-1) != 0 and (3,4,0) = 4 ) or
           ((2,4,-1) != 0 and (2,4,0) = 4 ) or
              ((1,4,-1) != 0 and (1,4,0) = 4 ) or
           ((1,0,-1) != 0 and (1,0,1) < 91) or
           ((2,0,-1) != 0 and (2,0,1) < 91 ) or
           ((3,0,-1) != 0 and (3,0,1) < 91 ) or
           ((3,-1,-1)!= 0 and (3,-1,1) < 91 and (3,-1,0) != 5 ) or
           ((3,-2,-1)!= 0 and (3,-2,1) < 91 and (3,-2,0) != 5 ) or
           ((3,-3,-1)!= 0 and (3,-3,1) < 91 and ( (3,-3,0) = 1 or (3,-
           3,0) = 2 or (3,-3,0) = 6 ) ) ) }
```

```
rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
          (1,-2,-1) = 2 and (2,-2,0) = 5 and cellPos(1) != 1 and
            (((2,-5,-1) = 10 or (2,-5,-1) = 11)  or
            ((3,3,-1) != 0 and (3,3,0) = 4 ) or
            ((2,3,-1) != 0 and (2,3,0) = 4 ) or
            ((1,3,-1) != 0 and (1,3,0) = 4 ) or
            ((1,-1,-1) != 0 and (1,-1,1) < 91) or
            ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
            ((3,-1,-1) != 0 and (3,-1,1) < 91 ) or
            ((3,-2,-1) != 0 and (3,-2,1) < 91 and (3,-2,0) != 5 ) or
            ((3,-3,-1) != 0 and (3,-3,1) < 91 and (3,-3,0) != 5 ) or
            ((3,-4,-1) != 0 and (3,-4,1) < 91 and ( (3,-4,0) = 1 or
          (3,-4,0) = 2 or (3,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
          1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) = 5 and
          cellPos(1) != 2 and
             (((2,-6,-1) = 10 or (2,-6,-1) = 11)  or
             ((3,2,-1) != 0 and (3,2,0) = 4 ) or
             ((2,2,-1) != 0 and (2,2,0) = 4 ) or
             ((1,2,-1) != 0 and (1,2,0) = 4 ) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91) or
             ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
             ((3,-2,-1) != 0 and (3,-2,1) < 91 ) or
             ((3,-3,-1) != 0 and (3,-3,1) < 91 and (3,-3,0) != 5 ) or
             ((3,-4,-1) != 0 and (3,-4,1) < 91 and (3,-4,0) != 5 ) or
             ((3,-5,-1) != 0 and (3,-5,1) < 91 and ( (3,-5,0) = 1 or
          (3,-5,0) = 2 or   (3,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,1) < 91 and (0,1,1) != 0 and (0,-6,-1) = 2
          and (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and
          (2,-4,0) = 5  and cellPos(1) != 3 and
             (((2,-7,-1) = 10 or (2,-7,-1) = 11)  or
             ((3,1,-1) != 0 and (3,1,0) = 4 ) or
             ((2,1,-1) != 0 and (2,1,0) = 4 ) or
             ((1,1,-1) != 0 and (1,1,0) = 4 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91) or
             ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
             ((3,-3,-1) != 0 and (3,-3,1) < 91 ) or
             ((3,-4,-1) != 0 and (3,-4,1) < 91 and (3,-4,0) != 5 ) or
             ((3,-5,-1) != 0 and (3,-5,1) < 91 and (3,-5,0) != 5 ) or
             ((3,-6,-1) != 0 and (3,-6,1) < 91 and ( (3,-6,0) = 1 or
            (3,-6,0) = 2 or     (3,-6,0) = 6 ) ) ) }

rule : 6 1  { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
          1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
          (1,-1,-1) = 2 and (2,-1,0) = 5 and cellPos(1) != 0 and
             (((2,-4,-1) = 10 or (2,-4,-1) = 11) or
             ((3,3,-1) != 0 and (3,3,0) = 4 ) or
             ((2,3,-1) != 0 and (2,3,0) = 4 ) or
             ((1,3,-1) != 0 and (1,3,0) = 4 ) or
             ((1,0,-1) != 0 and (1,0,1) < 91) or
             ((2,0,-1) != 0 and (2,0,1) < 91 ) or
```

```
              ((3,0,-1) != 0 and (3,0,1) < 91 ) or
              ((3,-1,-1) != 0 and (3,-1,1) < 91 and (3,-1,0) != 5 ) or
              ((3,-2,-1) != 0 and (3,-2,1) < 91 and (3,-2,0) != 5 ) or
              ((3,-3,-1) != 0 and (3,-3,1) < 91 and ( (3,-3,0) = 1 or
             (3,-3,0) = 2 or (3,-3,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,-1) = 2 and (2,-2,0) = 5 and cellPos(1) != 1 and
              (((2,-5,-1) = 10 or (2,-5,-1) = 11)  or
              ((-3,2,-1) != 0 and (3,2,0) = 4 ) or
              ((2,2,-1) != 0 and (2,2,0) = 4 ) or
              ((1,2,-1) != 0 and (1,2,0) = 4 ) or
              ((1,-1,-1) != 0 and (1,-1,1) < 91) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
              ((3,-1,-1) != 0 and (3,-1,1) < 91 ) or
              ((3,-2,-1) != 0 and (3,-2,1) < 91 and (3,-2,0) != 5 ) or
              ((3,-3,-1) != 0 and (3,-3,1) < 91 and (3,-3,0) != 5 ) or
              ((3,-4,-1) != 0 and (3,-4,1) < 91 and ( (3,-4,0) = 1 or
           (3,-4,0) = 2 or (3,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) = 5 and
           cellPos(1) != 2 and
              (((2,-6,-1) = 10 or (2,-6,-1) = 11)  or
              ((3,1,-1) != 0 and (3,1,0) = 4 ) or
              ((2,1,-1) != 0 and (2,1,0) = 4 ) or
              ((1,1,-1) != 0 and (1,1,0) = 4 ) or
              ((1,-2,-1) != 0 and (1,-2,1) < 91) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
              ((3,-2,-1) != 0 and (3,-2,1) < 91 ) or
              ((3,-3,-1) != 0 and (3,-3,1) < 91 and (3,-3,0) != 5 ) or
              ((3,-4,-1) != 0 and (3,-4,1) < 91 and (3,-4,0) != 5 ) or
              ((3,-5,-1) != 0 and (3,-5,1) < 91 and ( (3,-5,0) = 1 or
           (3,-5,0) = 2 or (3,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and (2,-
           4,0) = 5 and cellPos(1) != 3 and
              (((2,-7,-1) = 10 or (2,-7,-1) = 11)  or
              ((3,0,-1) != 0 and (3,0,0) = 4 ) or
              ((2,0,-1) != 0 and (2,0,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,0) = 4 ) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
              ((3,-3,-1) != 0 and (3,-3,1) < 91 ) or
              ((3,-4,-1) != 0 and (3,-4,1) < 91 and (3,-4,0) != 5 ) or
              ((3,-5,-1) != 0 and (3,-5,1) < 91 and (3,-5,0) != 5 ) or
              ((3,-6,-1) != 0 and (3,-6,1) < 91 and ( (3,-6,0) = 1 or
           (3,-6,0) = 2 or (3,-6,0) = 6 ) ) ) }

rule : 2 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
           (1,-1,-1) = 2 and (2,-1,0) = 5 and cellPos(1) != 0 and
              (2,-4,-1) != 10 and (2,-4,-1) != 11 and
             (3,3,-1) = 0 and
             (2,3,-1) = 0 and
             (1,3,-1) = 0 and
             (1,0,-1) = 0 and
             (2,0,-1) = 0 and
             (3,0,-1) = 0 and
             (3,-1,-1) = 0 and
             (3,-2,-1) = 0 and
             ((3,-3,-1) = 0 or (3,-3,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,-1) = 2 and (2,-2,0) = 5 and cellPos(1) != 1 and
             (2,-5,-1) != 10 and (2,-5,-1) != 11 and
            (3,2,-1) = 0 and
            (2,2,-1) = 0 and
             (1,2,-1) = 0 and
             (1,-1,-1) = 0 and
             (2,-1,-1) = 0 and
             (3,-1,-1) = 0 and
             (3,-2,-1) = 0 and
             (3,-3,-1) = 0 and
             ((3,-4,-1) = 0 or (3,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) = 5 and
           cellPos(1) != 2 and
             (2,-6,-1) != 10 and (2,-6,-1) != 11 and
             (3,1,-1) = 0 and
             (2,1,-1) = 0 and
             (1,1,-1) = 0 and
             (1,-2,-1) = 0 and
             (2,-2,-1) = 0 and
             (3,-2,-1) = 0 and
             (3,-3,-1) = 0 and
             (3,-4,-1) = 0 and
             ((3,-5,-1) = 0 or (3,-5,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and (2,-
           4,0) = 5 and cellPos(1) != 3 and
             (2,-7,-1) != 10 and (2,-7,-1) != 11 and
             (3,0,-1) = 0 and
             (2,0,-1) = 0 and
             (1,0,-1) = 0 and
             (1,-3,-1) = 0 and
             (2,-3,-1) = 0 and
             (3,-3,-1) = 0 and
             (3,-4,-1) = 0 and
             (3,-5,-1) = 0 and
             ((3,-6,-1) = 0  or (3,-6,0) = 4 ) }
#EndMacro
```

```
#BeginMacro(MoveCheckRule1_WaitingDelay_13)

rule : 6 1 { cellPos(0) = 1 and (0,0,-1) = 0 and (0,1,0) = 4 and
          remainder((0,1,1),4) = 1 and (0,1,1) < 91 and (0,-1,-1) = 2
          and (-1,-1,-1) = 2 and (1,-1,-1) = 2 and (2,-1,0) != 5 and
          cellPos(1) != 0 and
             (((2,-4,-1) = 10 or (2,-4,-1) = 11 ) or
             ((2,4,-1) != 0 and (2,4,0) = 4 ) or
             ((1,4,-1) != 0 and (1,4,0) = 4 ) or
             ((1,0,-1) != 0 and (1,0,1) < 91) or
             ((2,0,-1) != 0 and (2,0,1) < 91 ) or
             ((2,-1,-1) != 0 and (2,-1,1) < 91 and (2,-1,0) != 5 ) or
             ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
             ((2,-3,-1) != 0 and (2,-3,1) < 91 and ( (2,-3,0) = 1 or
          (2,-3,0) = 2 or (2,-3,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
          (1,-2,-1) = 2 and (2,-2,0) != 5 and cellPos(1) != 1 and
             (((2,-5,-1) = 10 or (2,-5,-1) = 11)  or
             ((2,3,-1) != 0 and (2,3,0) = 4 ) or
             ((1,3,-1) != 0 and (1,3,0) = 4 ) or
             ((1,-1,-1) != 0 and (1,-1,1) < 91) or
             ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
             ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
             ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
             ((2,-4,-1) != 0 and (2,-4,1) < 91 and ( (2,-4,0) = 1 or
          (2,-4,0) = 2 or (2,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
          1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) != 5 and
          cellPos(1) != 2 and
             (((2,-6,-1) = 10 or (2,-6,-1) = 11)  or
             ((2,2,-1) != 0 and (2,2,0) = 4 ) or
             ((1,2,-1) != 0 and (1,2,0) = 4 ) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91) or
             ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
             ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
             ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
             ((2,-5,-1) != 0 and (2,-5,1) < 91 and ( (2,-5,0) = 1 or
          (2,-5,0) = 2 or (2,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,1) < 91 and (0,1,1) != 0 and (0,-6,-1) = 2 and
          (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and (2,-
          4,0) != 5 and cellPos(1) != 3 and
             (((2,-7,-1) = 10 or (2,-7,-1) = 11)  or
             ((2,1,-1) != 0 and (2,1,0) = 4 ) or
             ((1,1,-1) != 0 and (1,1,0) = 4 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91) or
             ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
             ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
             ((2,-5,-1) != 0 and (2,-5,1) < 91 and (2,-5,0) != 5 ) or
             ((2,-6,-1) != 0 and (2,-6,1) < 91 and ( (2,-6,0) = 1 or
          (2,-6,0) = 2 or (2,-6,0) = 6 ) ) ) }
```

```
rule : 6 1  { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
           (1,-1,-1) = 2 and (2,-1,0) != 5 and cellPos(1) != 0 and
              (((2,-4,-1) = 10 or (2,-4,-1) = 11)  or
              ((2,3,-1) != 0 and (2,3,0) = 4 ) or
              ((1,3,-1) != 0 and (1,3,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,1) < 91) or
              ((2,0,-1) != 0 and (2,0,1) < 91 ) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 and (2,-1,0) != 5 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and ( (2,-3,0) = 1 or
           (2,-3,0) = 2 or (2,-3,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,-1) = 2 and (2,-2,0) != 5 and cellPos(1) != 1 and
              (((2,-5,-1) = 10 or (2,-5,-1) = 11)  or
              ((2,2,-1) != 0 and (2,2,0) = 4 ) or
              ((1,2,-1) != 0 and (1,2,0) = 4 ) or
              ((1,-1,-1) != 0 and (1,-1,1) < 91) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and ( (2,-4,0) = 1 or
           (2,-4,0) = 2 or (2,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) != 5 and
           cellPos(1) != 2 and
              (((2,-6,-1) = 10 or (2,-6,-1) = 11)  or
              ((2,1,-1) != 0 and (2,1,0) = 4 ) or
              ((1,1,-1) != 0 and (1,1,0) = 4 ) or
              ((1,-2,-1) != 0 and (1,-2,1) < 91) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and ( (2,-5,0) = 1 or
           (2,-5,0) = 2 or (2,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and (2,-
           4,0) != 5 and cellPos(1) != 3 and
              (((2,-7,-1) = 10 or (2,-7,-1) = 11)  or
              ((2,0,-1) != 0 and (2,0,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,0) = 4 ) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and (2,-5,0) != 5 ) or
              ((2,-6,-1) != 0 and (2,-6,1) < 91 and ( (2,-6,0) = 1 or
           (2,-6,0) = 2 or (2,-6,0) = 6 ) ) ) }

rule : 2 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
           (1,-1,-1) = 2 and (2,-1,0) != 5 and cellPos(1) != 0 and
```

```
             (2,-4,-1) != 10 and (2,-4,-1) != 11 and
             (2,3,-1) = 0 and
             (1,3,-1) = 0 and
             (1,0,-1) = 0 and
             (2,0,-1) = 0 and
             (2,-1,-1) = 0 and
             (2,-2,-1) = 0 and
             ((2,-3,-1) = 0 or (2,-3,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
          2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
          (1,-2,-1) = 2 and (2,-2,0) != 5 and cellPos(1) != 1 and
             (2,-5,-1) != 10 and (2,-5,-1) != 11 and
             (2,2,-1) = 0 and
             (1,2,-1) = 0 and
             (1,-1,-1) = 0 and
             (2,-1,-1) = 0 and
             (2,-2,-1) = 0 and
             (2,-3,-1) = 0 and
             ((2,-4,-1) = 0  or (2,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
          3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
          1,-3,-1) = 2 and (1,-3,-1) = 2 and (2,-3,0) != 5 and
          cellPos(1) != 2 and
             (2,-6,-1) != 10 and (2,-6,-1) != 11 and
             (2,1,-1) = 0 and
             (1,1,-1) = 0 and
             (1,-2,-1) = 0 and
             (2,-2,-1) = 0 and
             (2,-3,-1) = 0 and
             (2,-4,-1) = 0 and
             ((2,-5,-1) = 0 or (2,-5,0) = 4 ) }

rule : 4 1 { cellPos(0) = 1 and (0,0,0) = 6 and remainder((0,0,1),4) =
          0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
          (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,-1) = 2 and (2,-
          4,0) != 5 and cellPos(1) != 3 and
             (2,-7,-1) != 10 and (2,-7,-1) != 11 and
             (2,0,-1) = 0 and
             (1,0,-1) = 0 and
             (1,-3,-1) = 0 and
             (2,-3,-1) = 0 and
             (2,-4,-1) = 0 and
             (2,-5,-1) = 0 and
             ((2,-6,-1) = 0 or (2,-6,0) = 4 ) }

#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_23)

rule : 6 1 { cellPos(0) = 2 and (0,0,-1) = 0 and (0,1,0) = 4 and
          remainder((0,1,1),4) = 1 and (0,1,1) < 91 and (0,-1,-1) = 2
          and (-1,-1,-1) = 2 and (1,-1,0) != 5  and cellPos(1) != 0 and
             (((1,4,-1) != 0 and (1,4,0) = 4 ) or
             ((1,0,-1) != 0 and (1,0,1) < 91) or
             ((1,-1,-1) != 0 and (1,-1,1) < 91 and (1,-1,0) != 5 ) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91 and (1,-2,0) != 5 ) or
            ((1,-3,-1) != 0 and (1,-3,1) < 91 and ( (1,-3,0) != 5 or
          (1,-3,0) != 4 ) ) ) }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
          (1,-2,0) != 5 and cellPos(1) != 1 and
             (((1,3,-1) != 0 and (1,3,0) = 4 ) or
             ((1,-1,-1) != 0 and (1,-1,1) < 91) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91 and (1,-2,0) != 5 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91 and (1,-3,0) != 5 ) or
             ((1,-4,-1) != 0 and (1,-4,1) < 91 and ( (1,-4,0) != 5 or
          (1,-4,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
          1,-3,-1) = 2 and (1,-3,0) != 5 and cellPos(1) != 2 and
             (((1,2,-1) != 0 and (1,2,0) = 4 ) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91 and (1,-3,0) != 5 ) or
             ((1,-4,-1) != 0 and (1,-4,1) < 91 and (1,-4,0) != 5 ) or
             ((1,-5,-1) != 0 and (1,-5,1) < 91 and ( (1,-5,0) != 5 or
          (1,-5,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,1) < 91 and (0,1,1) != 0 and (0,-6,-1) = 2 and
          (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) != 5 and
          cellPos(1) != 3 and
             (((1,1,-1) != 0 and (1,1,0) = 4 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91) or
             ((1,-4,-1) != 0 and (1,-4,1) < 91 and (1,-4,0) != 5 ) or
             ((1,-5,-1) != 0 and (1,-5,1) < 91 and (1,-5,0) != 5 ) or
             ((1,-6,-1) != 0 and (1,-6,1) < 91 and ( (1,-6,0) != 5 or
          (1,-6,0) != 4 ) ) )  }

rule : 6 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
          (1,-1,0) != 5  and cellPos(1) != 0 and
             (((1,3,-1) != 0 and (1,3,0) = 4 ) or
             ((1,0,-1) != 0 and (1,0,1) < 91) or
             ((1,-1,-1) != 0 and (1,-1,1) < 91 and (1,-1,0) != 5 ) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91 and (1,-2,0) != 5 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91 and ( (1,-3,0) != 5 or
          (1,-3,0) != 4 ) ) ) }

rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
          (1,-2,0) != 5  and cellPos(1) != 1 and
             (((1,2,-1) != 0 and (1,2,0) = 4 ) or
             ((1,-1,-1) != 0 and (1,-1,1) < 91) or
             ((1,-2,-1) != 0 and (1,-2,1) < 91 and (1,-2,0) != 5 ) or
             ((1,-3,-1) != 0 and (1,-3,1) < 91 and (1,-3,0) != 5 ) or
             ((1,-4,-1) != 0 and (1,-4,1) < 91 and ( (1,-4,0) != 5 or
          (1,-4,0) != 4 ) ) )  }
```

```
rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,0) != 5  and cellPos(1) != 2 and
              (((1,1,-1) != 0 and (1,1,0) = 4 ) or
              ((1,-2,-1) != 0 and (1,-2,1) < 91) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91 and (1,-3,0) != 5 ) or
              ((1,-4,-1) != 0 and (1,-4,1) < 91 and (1,-4,0) != 5 ) or
              ((1,-5,-1) != 0 and (1,-5,1) < 91 and ( (1,-5,0) != 5 or
           (1,-5,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) != 5  and
           cellPos(1) != 3 and
              (((1,0,-1) != 0 and (1,0,0) = 4 ) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91) or
              ((1,-4,-1) != 0 and (1,-4,1) < 91 and (1,-4,0) != 5 ) or
              ((1,-5,-1) != 0 and (1,-5,1) < 91 and (1,-5,0) != 5 ) or
              ((1,-6,-1) != 0 and (1,-6,1) < 91 and ( (1,-6,0) != 5 or
           (1,-6,0) != 4 ) ) )  }

rule : 2 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
           (1,-1,0) != 5  and cellPos(1) != 0 and
              (1,3,-1) = 0 and
              (1,0,-1) = 0 and
              (1,-1,-1) = 0 and
              (1,-2,-1) = 0 and
              ((1,-3,-1) = 0 or (1,-3,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,0) != 5  and cellPos(1) != 1 and
              (1,2,-1) = 0 and
              (1,-1,-1) = 0 and
              (1,-2,-1) = 0 and
              (1,-3,-1) = 0 and
              ((1,-4,-1) = 0 or (1,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,0) != 5  and cellPos(1) != 2 and
              (1,1,-1) = 0 and
              (1,-2,-1) = 0 and
              (1,-3,-1) = 0 and
              (1,-4,-1) = 0 and
              ((1,-4,-1) = 0 or (1,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) != 5  and
           cellPos(1) != 3 and
              (1,0,-1) = 0 and
              (1,-3,-1) = 0 and
              (1,-4,-1) = 0 and
              (1,-5,-1) = 0 and
              ((1,-6,-1) = 0 or (1,-6,0) = 4 ) }
```

```
#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_24)

rule : 6 1 { cellPos(0) = 2 and (0,0,-1) = 0 and (0,1,0) = 4 and
           remainder((0,1,1),4) = 1 and (0,1,1) < 91 and (0,-1,-1) = 2
           and (-1,-1,-1) = 2 and (1,-1,0) = 5 and cellPos(1) != 0 and
              ((1,-4,-1) = 10 or (1,-4,-1) = 11 or
              ((2,4,-1) != 0 and (2,4,0) = 4 ) or
              ((1,4,-1) != 0 and (1,4,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,1) < 91) or
              ((2,0,-1) != 0 and (2,0,1) < 91 ) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 and (2,-1,0) != 5 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and ( (2,-3,0) = 1 or
           (2,-3,0) = 2 or (2,-3,0) = 6 ) ) ) }


rule : 6 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           2 and (0,1,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,0) = 5 and cellPos(1) != 1 and
              ((1,-5,-1) = 10 or (1,-5,-1) = 11 or
              ((2,3,-1) != 0 and (2,3,0) = 4 ) or
              ((1,3,-1) != 0 and (1,3,0) = 4 ) or
              ((1,-1,-1) != 0 and (1,-1,1) < 91) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and ( (2,-4,0) = 1 or
           (2,-4,0) = 2 or (2,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           3 and (0,1,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,0) = 5 and cellPos(1) != 2 and
              ((1,-6,-1) = 10 or (1,-6,-1) = 11 or
              ((2,2,-1) != 0 and (2,2,0) = 4 ) or
              ((1,2,-1) != 0 and (1,2,0) = 4 ) or
              ((1,-2,-1) != 0 and (1,-2,1) < 91) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and ( (2,-5,0) = 1 or
           (2,-5,0) = 2 or (2,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           0 and (0,1,1) < 91 and (0,1,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) = 5 and
           cellPos(1) != 3 and
              ((1,-7,-1) = 10 or (1,-7,-1) = 11 or
              ((2,1,-1) != 0 and (2,1,0) = 4 ) or
              ((1,1,-1) != 0 and (1,1,0) = 4 ) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and (2,-5,0) != 5 ) or
              ((2,-6,-1) != 0 and (2,-6,1) < 91 and ( (2,-6,0) = 1 or
           (2,-6,0) = 2 or (2,-6,0) = 6 ) ) ) }

rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
           (1,-1,0) = 5 and cellPos(1) != 0 and
              ((1,-4,-1) = 10 or (1,-4,-1) = 11 or
              ((2,3,-1) != 0 and (2,3,0) = 4 ) or
              ((1,3,-1) != 0 and (1,3,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,1) < 91) or
              ((2,0,-1) != 0 and (2,0,1) < 91 ) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 and (2,-1,0) != 5 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and ( (2,-3,0) = 1 or
           (2,-3,0) = 2 or (2,-3,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
           (1,-2,0) = 5 and cellPos(1) != 1 and
              ((1,-5,-1) = 10 or (1,-5,-1) = 11 or
              ((2,2,-1) != 0 and (2,2,0) = 4 ) or
              ((1,2,-1) != 0 and (1,2,0) = 4 ) or
              ((1,-1,-1) != 0 and (1,-1,1) < 91) or
              ((2,-1,-1) != 0 and (2,-1,1) < 91 ) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 and (2,-2,0) != 5 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and ( (2,-4,0) = 1 or
           (2,-4,0) = 2 or (2,-4,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
           1,-3,-1) = 2 and (1,-3,0) = 5 and cellPos(1) != 2 and
              ((1,-6,-1) = 10 or (1,-6,-1) = 11 or
              ((2,1,-1) != 0 and (2,1,0) = 4 ) or
              ((1,1,-1) != 0 and (1,1,0) = 4 ) or
              ((1,-2,-1) != 0 and (1,-2,1) < 91) or
              ((2,-2,-1) != 0 and (2,-2,1) < 91 ) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 and (2,-3,0) != 5 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and ( (2,-5,0) = 1 or
           (2,-5,0) = 2 or (2,-5,0) = 6 ) ) ) }

rule : 6 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
           (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) = 5 and
           cellPos(1) != 3 and
              ((1,-7,-1) = 10 or (1,-7,-1) = 11 or
              ((2,0,-1) != 0 and (2,0,0) = 4 ) or
              ((1,0,-1) != 0 and (1,0,0) = 4 ) or
              ((1,-3,-1) != 0 and (1,-3,1) < 91) or
              ((2,-3,-1) != 0 and (2,-3,1) < 91 ) or
              ((2,-4,-1) != 0 and (2,-4,1) < 91 and (2,-4,0) != 5 ) or
              ((2,-5,-1) != 0 and (2,-5,1) < 91 and (2,-5,0) != 5 ) or
              ((2,-6,-1) != 0 and (2,-6,1) < 91 and ( (2,-6,0) = 1 or
           (2,-6,0) = 2 or (2,-6,0) = 6 ) ) ) }
```

```
rule : 2 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
         1 and (0,0,1) < 91 and (0,-1,-1) = 2 and (-1,-1,-1) = 2 and
         (1,-1,0) = 5 and cellPos(1) != 0 and
           (1,-4,-1) != 10 and (1,-4,-1) != 11 and
           (2,3,-1) = 0 and
           (1,3,-1) = 0 and
           (1,0,-1) = 0 and
           (2,0,-1) = 0 and
           (2,-1,-1) = 0 and
           (2,-2,-1) = 0 and
           ((2,-3,-1) = 0 or (2,-3,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
         2 and (0,0,1) < 91 and (0,-2,-1) = 2 and (-1,-2,-1) = 2 and
         (1,-2,0) = 5 and cellPos(1) != 1 and
           (1,-5,-1) != 10 and (1,-5,-1) != 11 and
           (2,2,-1) = 0 and
           (1,2,-1) = 0 and
           (1,-1,-1) = 0 and
           (2,-1,-1) = 0 and
           (2,-2,-1) = 0 and
           (2,-3,-1) = 0 and
           ((2,-4,-1) = 0  or (2,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
         3 and (0,0,1) < 91 and (0,-5,-1) = 2 and (0,-3,-1) = 2 and (-
         1,-3,-1) = 2 and (1,-3,0) = 5 and cellPos(1) != 2 and
           (1,-6,-1) != 10 and (1,-6,-1) != 11 and
           (2,1,-1) = 0 and
           (1,1,-1) = 0 and
           (1,-2,-1) = 0 and
           (2,-2,-1) = 0 and
           (2,-3,-1) = 0 and
           (2,-4,-1) = 0 and
           ((2,-5,-1) = 0 or (2,-5,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
         0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-6,-1) = 2 and
         (0,-4,-1) = 2 and (-1,-4,-1) = 2 and (1,-4,0) = 5 and
         cellPos(1) != 3 and
           (1,-7,-1) != 10 and (1,-7,-1) != 11 and
           (2,0,-1) = 0 and
           (1,0,-1) = 0 and
           (1,-3,-1) = 0 and
           (2,-3,-1) = 0 and
           (2,-4,-1) = 0 and
           (2,-5,-1) = 0 and
           ((2,-6,-1) = 0 or (2,-6,0) = 4 ) }

#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_21)

rule : 6 1 { cellPos(0) = 2 and (0,0,-1) = 0 and (0,1,0) = 4 and
         remainder((0,1,1),4) = 1 and (0,1,1) < 91 and (0,-1,0) = 5
```

```
           and (1,-1,-1) = 2 and cellPos(1) != 0 and ((0,-4,-1) = 10 or
           (0,-4,-1) = 11  or
              ((-1,4,-1) != 0 and (-1,4,0) = 4 ) or
              ((-1,0,-1) != 0 and (-1,0,1) < 91) or
              ((-1,-1,-1) != 0 and (-1,-1,1) < 91 and (-1,-1,0) != 5 )
           or
              ((-1,-2,-1) != 0 and (-1,-2,1) < 91 and (-1,-2,0) != 5 )
           or
             ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and ( (-1,-3,0) != 5 or
           (-1,-3,0) != 4 ) ) ) }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           2 and (0,1,1) < 91 and (0,-2,0) = 5 and (1,-2,-1) = 2 and
           cellPos(1) != 1 and
              ( (0,-5,-1) = 10 or (0,-5,-1) = 11  or
              ((-1,3,-1) != 0 and (-1,3,0) = 4 ) or
              ((-1,-1,-1) != 0 and (-1,-1,1) < 91) or
              ((-1,-2,-1) != 0 and (-1,-2,1) < 91 and (-1,-2,0) != 5 )
           or
              ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and (-1,-3,0) != 5 )
           or
              ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and ( (-1,-4,0) != 5
           or (-1,-4,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           3 and (0,1,1) < 91 and (0,-3,0) = 5 and (1,-3,-1) = 2 and
           cellPos(1) != 2 and
              ( (0,-6,-1) = 10 or (0,-6,-1) = 11  or
              ((-1,2,-1) != 0 and (-1,2,0) = 4 ) or
              ((-1,-2,-1) != 0 and (-1,-2,1) < 91) or
              ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and (-1,-3,0) != 5 )
           or
              ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and (-1,-4,0) != 5 )
           or
              ((-1,-5,-1) != 0 and (-1,-5,1) < 91 and ( (-1,-5,0) != 5
           or (-1,-5,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
           0 and (0,1,1) < 91 and (0,1,1) != 0 and (0,-4,0) = 5 and (1,-
           4,-1) = 2 and cellPos(1) != 3 and
              ((0,-7,-1) = 10 or (0,-7,-1) = 11  or
              ((-1,1,-1) != 0 and (-1,1,0) = 4 ) or
              ((-1,-3,-1) != 0 and (-1,-3,1) < 91) or
              ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and (-1,-4,0) != 5 )
           or
              ((-1,-5,-1) != 0 and (-1,-5,1) < 91 and (-1,-5,0) != 5 )
           or
              ((-1,-6,-1) != 0 and (-1,-6,1) < 91 and ( (-1,-6,0) != 5
           or (-1,-6,0) != 4 ) ) )  }

rule : 6 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           1 and (0,0,1) < 91 and (0,-1,0) = 5 and (1,-1,-1) = 2 and
           cellPos(1) != 0 and
              ((0,-4,-1) = 10 or (0,-4,-1) = 11  or
              ((-1,3,-1) != 0 and (-1,3,0) = 4 ) or
              ((-1,0,-1) != 0 and (-1,0,1) < 91) or
             ((-1,-1,-1) != 0 and (-1,-1,1) < 91 and (-1,-1,0) != 5 )
          or
             ((-1,-2,-1) != 0 and (-1,-2,1) < 91 and (-1,-2,0) != 5 )
          or
             ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and ( (-1,-3,0) != 5
          or (-1,-3,0) != 4 ) ) ) }


rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          2 and (0,0,1) < 91 and (0,-2,0) = 5 and (1,-2,-1) = 2 and
          cellPos(1) != 1 and
             ((0,-5,-1) = 10 or (0,-5,-1) = 11  or
             ((-1,2,-1) != 0 and (-1,2,0) = 4 ) or
             ((-1,-1,-1) != 0 and (-1,-1,1) < 91) or
             ((-1,-2,-1) != 0 and (-1,-2,1) < 91 and (-1,-2,0) != 5 )
          or
             ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and (-1,-3,0) != 5 )
          or
             ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and ( (-1,-4,0) != 5
          or (-1,-4,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          3 and (0,0,1) < 91 and (0,-3,0) = 5 and (1,-3,-1) = 2 and
          cellPos(1) != 2 and
             ((0,-6,-1) = 10 or (0,-6,-1) = 11  or
             ((-1,1,-1) != 0 and (-1,1,0) = 4 ) or
             ((-1,-2,-1) != 0 and (-1,-2,1) < 91) or
             ((-1,-3,-1) != 0 and (-1,-3,1) < 91 and (-1,-3,0) != 5 )
          or
             ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and (-1,-4,0) != 5 )
          or
             ((-1,-5,-1) != 0 and (-1,-5,1) < 91 and ( (-1,-5,0) != 5
          or (-1,-5,0) != 4 ) ) )  }

rule : 6 1  { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-4,0) = 5 and (1,-
          4,-1) = 2 and cellPos(1) != 3 and
             ((0,-7,-1) = 10 or (0,-7,-1) = 11  or
             ((-1,0,-1) != 0 and (-1,0,0) = 4 ) or
             ((-1,-3,-1) != 0 and (-1,-3,1) < 91) or
             ((-1,-4,-1) != 0 and (-1,-4,1) < 91 and (-1,-4,0) != 5 )
          or
             ((-1,-5,-1) != 0 and (-1,-5,1) < 91 and (-1,-5,0) != 5 )
          or
             ((-1,-6,-1) != 0 and (-1,-6,1) < 91 and ( (-1,-6,0) != 5
          or (-1,-6,0) != 4 ) ) )  }

rule : 1 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
          1 and (0,0,1) < 91 and (0,-1,0) = 5 and (1,-1,-1) = 2 and
          cellPos(1) != 0 and
             (0,-4,-1) != 10 and (0,-4,-1) != 11 and
             (-1,3,-1) = 0 and
             (-1,0,-1) = 0 and
             (-1,-1,-1) = 0 and
             (-1,-2,-1) = 0 and
             ((-1,-3,-1) = 0 or (-1,-3,0) = 4 ) }
```

```
rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           2 and (0,0,1) < 91 and (0,-2,0) = 5 and (1,-2,-1) = 2 and
           cellPos(1) != 1 and
              (0,-5,-1) != 10 and (0,-5,-1) != 11 and
              (-1,2,-1) = 0 and
              (-1,-1,-1) = 0 and
              (-1,-2,-1) = 0 and
              (-1,-3,-1) = 0 and
              ((-1,-4,-1) = 0 or (-1,-4,0) = 4 ) }

rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           3 and (0,0,1) < 91 and (0,-3,0) = 5 and (1,-3,-1) = 2 and
           cellPos(1) != 2 and
              (0,-6,-1) != 10 and (0,-6,-1) != 11 and
              (-1,1,-1) = 0 and
              (-1,-2,-1) = 0 and
              (-1,-3,-1) = 0 and
              (-1,-4,-1) = 0 and
              ((-1,-4,-1) = 0 or (-1,-4,0) = 4 ) }


rule : 4 1 { cellPos(0) = 2 and (0,0,0) = 6 and remainder((0,0,1),4) =
           0 and (0,0,1) < 91 and (0,0,1) != 0 and (0,-4,0) = 5 and (1,-
           4,-1) = 2 and cellPos(1) != 3 and (0,-7,-1) != 10 and (0,-7,-
           1) != 11 and
              (-1,0,-1) = 0 and
              (-1,-3,-1) = 0 and
              (-1,-4,-1) = 0 and
              (-1,-5,-1) = 0 and
              ((-1,-6,-1) = 0 or (-1,-6,0) = 4 ) }

#EndMacro


#BeginMacro(MoveCheckRule_WaitingDelay_end)

rule : {(0,1,0)} 1  {cellPos(1) = 0 and remainder((0,1,1),4) = 1 }
rule : {(0,1,0)} 1  {cellPos(1) < 1 and remainder((0,1,1),4) = 2}
rule : {(0,1,0)} 1  {cellPos(1) < 2 and remainder((0,1,1),4) = 3}
rule : {(0,1,0)} 1  {cellPos(1) < 3 and remainder((0,1,1),4) = 0}

#EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_SameLane1)
rule : 6 1 { (0,0,-1) = 0 and (0,-1,-1) = 0 and (0,1,0) = 4 and ((0,-
           2,0) = 6 or (0,-2,0) = 1 or (0,-2,0) = 2 or (0,-5,-1) = 10 or
           (0,-5,-1) = 11
             or ((0,-2,0) = 4 and remainder((0,-2,1),4) != 0 and (0,-
           2,1) != 0 ) or ((1,-3,0) = 1 or (2,-3,0) = 1 )
             or ((-1,-3,0) = 2 or (-2,-3,0) = 2 ) )
             and cellPos(1) != 0 and remainder((0,1,1),4) = 1 }

rule : 6 1 { ((0,-3,0) = 6 or (0,-3,0) = 1 or (0,-3,0) = 2 or (0,-6,-1)
           = 10 or (0,-6,-1) = 11 or ((0,-3,0) = 4 and remainder((0,-
       3,1),4) != 0 and (0,-3,1) != 0 ) or ((1,-4,0) = 1 or (2,-4,0)
       = 1 ) or ((-1,-4,0) = 2 or (-2,-4,0) = 2 ) ) and (0,1,0) = 4
       and cellPos(1) > 1 and remainder((0,1,1),4) = 2 and
       (trunc((0,1,1)/4 ) = trunc((0,0,1)/4 ))}


rule : 6 1 { ((0,-4,0) = 6 or (0,-4,0) = 1 or (0,-4,0) = 2 or (0,-7,-1)
       = 10 or (0,-7,-1) = 11 or ((0,-4,0) = 4 and remainder((0,-
       4,1),4) != 0 and (0,-4,1) != 0 ) or ((1,-5,0) = 1 or (2,-5,0)
       = 1 ) or ((-1,-5,0) = 2 or (-2,-5,0) = 2 ) ) and (0,1,0) = 4
       and cellPos(1) > 2 and remainder((0,1,1),4) = 3 and
       (trunc((0,1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { ((0,-5,0) = 6 or (0,-5,0) = 1 or (0,-5,0) = 2 or (0,-8,-1)
       = 10 or (0,-8,-1) = 11 or ((0,-5,0) = 4 and remainder((0,-
       5,1),4) != 0 and (0,-5,1) != 0 ) or ((1,-6,0) = 1 or (2,-6,0)
       = 1 ) or ((-1,-6,0) = 2 or (-2,-6,0) = 2 )  ) and (0,1,0) = 4
       and cellPos(1) > 3 and remainder((0,1,1),4) = 0 and
       (trunc((0,1,1)/4 - 1 ) = trunc((0,0,1)/4 )) }

rule : 6 1 { (0,0,0) = 6 and ((0,-2,0) = 6 or (0,-2,0) = 1 or (0,-2,0)
       = 2
        or (0,-5,-1) = 10 or (0,-5,-1) = 11 or ((0,-2,0) = 4 and
       remainder((0,-2,1),4) != 0 and (0,-3,1) != 0 ) or ((1,-3,0) =
       1 or (2,-2,0) = 1 )
        or ((-1,-3,0) = 2 or (-2,-3,0) = 2 ) ) and cellPos(1) != 0
       and remainder((0,0,1),4) = 1 }

rule : 6 1 { ((0,-3,0) = 6 or (0,-3,0) = 1 or (0,-3,0) = 2 or (0,-6,-1)
       = 10 or (0,-6,-1) = 11 or ((0,-3,0) = 4 and remainder((0,-
       3,1),4) != 0 and (0,-3,1) != 0 ) or ((1,-4,0) = 1 or (2,-4,0)
       = 1 ) or ((-1,-4,0) = 2 or (-2,-4,0) = 2 ) ) and (0,0,0) = 6
       and cellPos(1) > 1 and remainder((0,0,1),4) = 2 and
       (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { ((0,-4,0) = 6 or (0,-4,0) = 1 or (0,-4,0) = 2 or (0,-7,-1)
       = 10 or (0,-7,-1) = 11 or ((0,-4,0) = 4 and remainder((0,-
       4,1),4) != 0 and (0,-4,1) != 0 ) or ((1,-5,0) = 1 or (2,-5,0)
       = 1 ) or ((-1,-5,0) = 2 or (-2,-5,0) = 2 ) ) and (0,0,0) = 6
       and cellPos(1) > 2 and remainder((0,0,1),4) = 3 and
       (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { ((0,-5,0) = 6 or (0,-5,0) = 1 or (0,-5,0) = 2 or (0,-8,-1)
       = 10 or (0,-8,-1) = 11 or ((0,-5,0) = 4 and remainder((0,-
       5,1),4) != 0 and (0,-5,1) != 0 ) or ((1,-6,0) = 1 or (2,-6,0)
       = 1 ) or ((-1,-6,0) = 2 or (-2,-6,0) = 2 ) ) and (0,0,0) = 6
       and cellPos(1) > 3 and remainder((0,0,1),4) = 0 and
       (trunc((0,-1,1)/4  ) = trunc((0,0,1)/4 - 1 )) }

rule : 4 2 { (0,0,0) = 6 and (0,-1,0) = 0 and ((0,-2,0) = 0 and (0,-5,-
       1) != 10 and (0,-5,-1) != 11 or ((0,-2,0) = 4 and
       remainder((0,-2,1),4) = 0 and (0,-3,1) != 0 )) and ((1,-
       3,0) != 1 and (2,-2,0) != 1 )
        and ((-1,-3,0) != 2 and  (-2,-3,0) != 2 ) and cellPos(1) !=
       0 and remainder((0,0,1),4) = 1 }
```

```
rule : 4 2 { ((0,-3,0) = 0 or ((0,-3,0) = 4 and remainder((0,-3,1),4) =
          0 and (0,-3,1) != 0 )) and ((1,-4,0) != 1 and (2,-4,0) != 1)
          and ((-1,-4,0) != 2 and  (-2,-4,0) != 2 )  and (0,0,0) = 6
            and cellPos(1) > 1 and remainder((0,0,1),4) = 2
            and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))
            and (0,-6,-1) != 10 and (0,-6,-1) != 11 }

rule : 4 2 { ((0,-4,0) = 0 or ((0,-4,0) = 4 and remainder((0,-4,1),4) =
          0 and (0,-4,1) != 0 )) and ((1,-5,0) != 1 and (2,-5,0) != 1)
          and ((-1,-5,0) != 2 and  (-2,-5,0) != 2 ) and (0,0,0) = 6
            and cellPos(1) > 2 and remainder((0,0,1),4) = 3
            and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))
            and (0,-7,-1) != 10 and (0,-7,-1) != 11 }

rule : 4 2 { ((0,-5,0) = 0 or ((0,-5,0) = 4 and remainder((0,-5,1),4) =
          0 and (0,-5,1) != 0 )) and ((1,-6,0) != 1 and (2,-6,0) != 1)
          and ((-1,-6,0) != 2 and  (-2,-6,0) != 2 ) and (0,0,0) = 6
            and cellPos(1) > 3 and remainder((0,0,1),4) = 0
            and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 - 1 ))
            and (0,-8,-1) != 10 and (0,-8,-1) != 11 }
  #EndMacro


#BeginMacro(MoveCheckRule1_WaitingDelay_SameLane2)

rule : 6 1 { (0,0,-1) = 0 and (0,1,0) = 4 and ((0,-1,0) = 6 or (0,-1,0)
          = 1 or (0,-1,0) = 2 or (0,-4,-1) = 10 or (0,-4,-1) = 11
            or (0,-1,0) = 4  or ((1,-2,0) = 1 or (2,-2,0) = 1 ) or ((-
          1,-2,0) = 2 or (-2,-2,0) = 2 )) and cellPos(1) != 0 and
          remainder((0,1,1),4) = 1 }
rule : 6 1 { (0,1,0) = 4 and ((0,-2,0) = 6 or (0,-2,0) = 1 or (0,-2,0)
          = 2 or (0,-2,0) = 4 or ((1,-3,0) = 1 or (2,-3,0) = 1 or (0,-
          5,-1) = 10 or (0,-5,-1) = 11 )
            or ((-1,-3,0) = 2 or (-2,-3,0) = 2 ))
            and cellPos(1) > 1 and remainder((0,1,1),4) = 2
            and (trunc((0,1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { (0,1,0) = 4 and ((0,-3,0) = 6 or (0,-3,0) = 1 or (0,-3,0)
          = 2 or (0,-3,0) = 4 or ((1,-4,0) = 1 or (2,-4,0) = 1 or (0,-
          6,-1) = 10 or (0,-6,-1) = 11 )
            or ((-1,-4,0) = 2 or (-2,-4,0) = 2 ))
            and cellPos(1) > 2 and remainder((0,1,1),4) = 3
            and (trunc((0,1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { (0,1,0) = 4 and ((0,-4,0) = 6 or (0,-4,0) = 1 or (0,-4,0)
          = 2 or (0,-4,0) = 4 or ((1,-5,0) = 1 or (2,-5,0) = 1 or (0,-
          7,-1) = 10 or (0,-7,-1) = 11 )
            or ((-1,-5,0) = 2 or (-2,-5,0) = 2 ))
            and cellPos(1) > 3 and remainder((0,1,1),4) = 0
            and (trunc((0,1,1)/4 - 1 ) = trunc((0,0,1)/4 )) }

rule : 6 1 { (0,0,0) = 6 and ((0,-1,0) = 6 or (0,-1,0) = 1 or (0,-1,0)
          = 2 or (0,-4,-1) = 10 or (0,-4,-1) = 11
          or (0,-1,0) = 4  or ((1,-2,0) = 1 or (2,-2,0) = 1 ) or ((-
        1,-2,0) = 2 or (-2,-2,0) = 2 ) ) and cellPos(1) != 0 and
        remainder((0,0,0),4) = 1 }


rule : 6 1 { (0,0,0) = 6 and ((0,-2,0) = 6 or (0,-2,0) = 1 or (0,-2,0)
        = 2 or (0,-2,0) = 4 or (0,-5,-1) = 10 or (0,-5,-1) = 11
         or ((1,-3,0) = 1 or (2,-3,0) = 1 ) or ((-1,-3,0) = 2 or (-
        2,-3,0) = 2 ) )
         and cellPos(1) > 1 and remainder((0,0,1),4) = 2
         and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { (0,0,0) = 6 and ((0,-3,0) = 6 or (0,-3,0) = 1 or (0,-3,0)
        = 2 or (0,-3,0) = 4 or (0,-6,-1) = 10 or (0,-6,-1) = 11
         or ((1,-4,0) = 1 or (2,-4,0) = 1 ) or ((-1,-4,0) = 2 or (-
        2,-4,0) = 2 ) )
         and cellPos(1) > 2 and remainder((0,0,1),4) = 3
         and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 6 1 { (0,0,0) = 6 and ((0,-4,0) = 6 or (0,-4,0) = 1 or (0,-4,0)
        = 2 or (0,-4,0) = 4 or (0,-7,-1) = 10 or (0,-7,-1) = 11
         or ((1,-5,0) = 1 or (2,-5,0) = 1 ) or ((-1,-5,0) = 2 or (-
        2,-5,0) = 2 ) )
         and cellPos(1) > 3 and remainder((0,0,1),4) = 0
         and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 - 1 )) }

rule : 4 3 { (0,0,0) = 6 and (0,-1,0) = 0 and (1,-2,0) != 1 and (2,-
        2,0) != 1 and (0,-4,-1) != 10 and (0,-4,-1) != 11
         and (-1,-2,0) != 2 and (-2,-2,0) != 2
         and cellPos(1) != 0  and remainder((0,0,1),4) = 1 }

rule : 4 3 { (0,0,0) = 6 and (0,-2,0) = 0 and (1,-3,0) != 1 and (2,-
        3,0) != 1 and (0,-5,-1) != 10 and (0,-5,-1) != 11
         and (-1,-3,0) != 2 and (-2,-3,0) != 2
         and cellPos(1) > 1 and remainder((0,0,1),4) = 2 and
        (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 4 3 { (0,0,0) = 6 and (0,-3,0) = 0 and (1,-4,0) != 1 and (2,-
        4,0) != 1 and (0,-6,-1) != 10 and (0,-6,-1) != 11
         and (-1,-4,0) != 2 and (-2,-4,0) != 2
         and cellPos(1) > 2 and remainder((0,0,1),4) = 3 and
        (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 ))}

rule : 4 3 { (0,0,0) = 6 and (0,-4,0) = 0 and (1,-5,0) != 1 and (2,-
        5,0) != 1 and (0,-7,-1) != 10 and (0,-7,-1) != 11
         and (-1,-5,0) != 2 and (-2,-5,0) != 2
         and cellPos(1) > 3 and remainder((0,0,1),4) = 0
         and (trunc((0,-1,1)/4 ) = trunc((0,0,1)/4 - 1 )) }

#EndMacro


#BeginMacro(MoveCheckRule_AtLoading)

rule : 2 1 {(0,1,0) = 4 and (0,0,0) = 0 and (1,1,-1) = 0 and
        remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (2,-1,-1) = 3
        and ((2,-1,1) - trunc((0,1,1)/4 ) = 91 ) }
```

```
rule : 2 1 {(0,1,0) = 4 and remainder((0,1,1),4) = 0 and (0,1,-1) = 1
                   and (2,-1,-1) = 3 and ((2,-1,1) - trunc((0,1,1)/4 -
           1 ) = 91 ) }

rule : 2 1 {(0,1,0) = 4 and remainder((0,1,1),4) != 0 and
           remainder((0,1,1),4) != 1 and (0,1,-1) = 1 and (2,-1,-1) = 3
           and ((2,-1,1) - trunc((0,1,1)/4 ) = 91 ) }

rule : 2 1 { (0,1,0) = 4 and (0,0,0) = 0 and (1,1,-1) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (2,-1,-1) = 5
           and ((2,-1,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 2 1 {(0,1,0) = 4 and remainder((0,1,1),4) = 0 and (0,1,-1) = 8
                   and (2,-1,-1) = 5 and ((2,-1,1) - trunc((0,1,1)/4 -
           1 ) = 81 ) }

rule : 2 1 { (0,1,0) = 4 and remainder((0,1,1),4) != 0 and
           remainder((0,1,1),4) != 1 and (0,1,-1) = 8 and (2,-1,-1) = 5
           and ((2,-1,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 1 1 {(0,1,0) = 4 and (0,0,0) = 0 and (-1,1,-1) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-2,-1,-1) = 3
           and ((-2,-1,1) - trunc((0,1,1)/4 ) = 91 ) }

rule : 1 1 {(0,1,0) = 4 and remainder((0,1,1),4) = 0 and (0,1,-1) = 1
                   and (-2,-1,-1) = 3 and ((-2,-1,1) - trunc((0,1,1)/4
           - 1 ) = 91 ) }

rule : 1 1 {(0,1,0) = 4 and remainder((0,1,1),4) != 0 and
           remainder((0,1,1),4) != 1 and (0,1,-1) = 1 and (-2,-1,-1) = 3
           and ((-2,-1,1) - trunc((0,1,1)/4 ) = 91 ) }

rule : 1 1 { (0,1,0) = 4 and (0,0,0) = 0 and (-1,1,-1) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (-2,-1,-1) = 5
           and ((-2,-1,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 1 1 {(0,1,0) = 4 and remainder((0,1,1),4) = 0 and (0,1,-1) = 8
                   and (-2,-1,-1) = 5 and ((-2,-1,1) - trunc((0,1,1)/4
           - 1 ) = 81 ) }

rule : 1 1 { (0,1,0) = 4 and remainder((0,1,1),4) != 0 and
           remainder((0,1,1),4) != 1 and (0,1,-1) = 8 and (-2,-1,-1) = 5
           and ((-2,-1,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 4 1 {(0,0,0) = 0 and (0,-1,0) = 0 and (-1,0,0) = 2 and (1,0,0) =
           5 and remainder((-1,0,1),4) = 1 }

rule : 4 1 {(-1,0,0) = 2 and (1,0,0) = 5 and (0,-1,0) != 5 and
           remainder((-1,0,1),4) != 1 and remainder((-1,0,1),4) != 0}

rule : 4 1 {(0,0,0) = 0 and (0,-1,0) = 0 and (1,0,0) = 1 and (-1,0,0) =
           5 and remainder((1,0,1),4) = 1 }

rule : 4 1 {(1,0,0) = 1 and (-1,0,0) = 5 and (0,-1,0) != 5 and
           remainder((1,0,1),4) != 1 and remainder((1,0,1),4) != 0}

rule : 5 1 {(-1,0,0) = 2 and (1,0,0) = 5 and remainder((0,1,1),4) = 0
```

```
                   and ((1,-1,1) - trunc((-1,0,1)/4 - 1 ) = 91 ))}
rule : 5 1 {(-1,0,0) = 2 and (1,0,0) = 5 and remainder((0,1,1),4) = 0
             and ((1,-1,1) - trunc((-1,0,1)/4 - 1 ) = 81 ))}

rule : 5 1 {(1,0,0) = 1 and (-1,0,0) = 5 and remainder((0,1,1),4) = 0
                   and ((-1,-1,1) - trunc((1,0,1)/4 - 1 ) = 91 ))}

rule : 5 1 {(1,0,0) = 1 and (-1,0,0) = 5 and remainder((0,1,1),4) = 0
             and ((-1,-1,1) - trunc((1,0,1)/4 - 1 ) = 81 ))}

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and (0,0,0) = 0 and
          remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (1,2,-1) = 3
          and ((1,2,1) - trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,-1) = 1 and (1,1,-1) = 3 and ((1,1,1) -
          trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,-1) = 1 and (1,0,-1) = 3 and ((1,0,1) -
          trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,-1) = 1 and (1,-1,-1) = 3 and ((1,-1,1) -
          trunc((0,1,1)/4 - 1 ) = 91 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and (0,0,0) = 0 and
          remainder((0,1,1),4) = 1 and (0,1,-1) = 8   and (1,2,-1) = 5
          and ((1,2,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,-1) = 8 and (1,1,-1) = 5 and ((1,1,1) -
          trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,-1) = 8 and (1,0,-1) = 5 and ((1,0,1) -
          trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 2 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,-1) = 8 and (1,-1,-1) = 5 and ((1,-1,1) -
          trunc((0,1,1)/4 - 1 ) = 81 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and (0,0,0) = 0 and
          remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-1,2,-1) = 3
          and ((-1,2,1) - trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,-1) = 1 and (-1,1,-1) = 3 and ((-1,1,1) -
          trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,-1) = 1 and (-1,0,-1) = 3 and ((-1,0,1) -
          trunc((0,1,1)/4 ) = 91 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,-1) = 1 and (-1,-1,-1) = 3 and ((-1,-1,1) -
          trunc((0,1,1)/4 - 1 ) = 91 ) }
```

```
rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and (0,0,0) = 0 and
          remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (-1,2,-1) = 5
          and ((-1,2,1) - trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          2 and (0,1,-1) = 8 and (-1,1,-1) = 5 and ((-1,1,1) -
          trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          3 and (0,1,-1) = 8 and (-1,0,-1) = 5 and ((-1,0,1) -
          trunc((0,1,1)/4 ) = 81 ) }

rule : 5 1 { cellPos(0) = 3 and (0,1,0) = 4 and remainder((0,1,1),4) =
          0 and (0,1,-1) = 8 and (-1,-1,-1) = 5 and ((-1,-1,1) -
          trunc((0,1,1)/4 - 1 ) = 81 ) }

 #EndMacro


#BeginMacro(MoveCheckRule1_GoAroundTeam)

rule : 1 1 { cellPos(0) = 2 and (0,1,0) = 4 and (0,0,0) = 0 and (-1,0,0)
          = 0 and (0,-1,0) = 5 and (1,-1,0) = 5 and (1,-1,-1) = 2 and
          remainder((0,1,1),4) = 1 and (1,-4,-1) != 10 and (1,-4,-1) !=
          11  and
             (0,-4,-1) != 10 and (0,-4,-1) != 11 and
             (2,-4,-1) != 10 and (2,-4,-1) != 11
             and ((0,1,-1) != 12 and (0,1,-1) != 13 and (0,1,-1) != 14
          and (0,1,-1) != 15 )}

 rule : 1 1 { (1,0,0) = 1 and (0,0,0) = 0 and (-1,0,0) = 0 and (0,-1,0)
          = 5 and (0,-1,-1)!= 2 and (1,-1,0) = 5 and (1,-1,-1) = 2 and
          remainder((1,0,1),4) = 1
              and ((1,0,-1) != 12 and (1,0,-1) != 13 and (1,0,-1) != 14
          and (1,0,-1) != 15) }

rule : 2 1 { cellPos(0) < 3 and (0,1,0) = 4 and (0,0,0) = 0 and (1,0,0)
          = 0 and (0,-1,0) = 5 and remainder((0,1,1),4) = 1 and
             (2,-4,-1) != 10 and (2,-4,-1) != 11
             and ((0,1,-1) != 12 and (0,1,-1) != 13 and (0,1,-1) != 14
          and (0,1,-1) != 15) }

rule : 2 1 { cellPos(0) = 2 and (0,1,0) = 4 and (0,0,0) = 0 and (1,0,0)
          = 0 and (0,-1,0) = 5 and (-1,-1,0) = 5 and (-1,-1,-1) = 2 and
          remainder((0,1,1),4) = 1 and (-1,-4,-1) != 10 and (-1,-4,-
          1) != 11  and (0,-4,-1) != 10 and (0,-4,-1) != 11 and ((0,1,-
          1) != 12 and (0,1,-1) != 13 and (0,1,-1) != 14 and (0,1,-
          1) != 15 ) }

rule : 2 1 { (-1,0,0) = 2 and (0,0,0) = 0 and (1,0,0) = 0 and (0,-1,0)
          = 5
             and remainder((-1,0,1),4) = 1 and ((-1,0,-1) != 12 and (-
          1,0,-1) != 13 and (-1,0,-1) != 14 and (-1,0,-1) != 15 ) }

rule : 4 1 { (0,0,0) = 0 and (1,0,0) = 1 and (0,-1,0) = 0 and
          remainder((1,0,1),4) = 1
```

```
            and ((1,0,-1) != 12 and (1,0,-1) != 13 and (1,0,-1) != 14
          and (1,0,-1) != 15 ) }

rule : 4 1 { (0,0,0) = 0 and (-1,0,0) = 2 and (0,-1,0) = 0 and
          remainder((-1,0,1),4) = 1 and ((-1,0,-1) != 12 and (-1,0,-
          1) != 13 and (-1,0,-1) != 14 and (-1,0,-1) != 15 ) }

rule : {(0,0,0)} 1 { ((1,0,-1) = 1 or (1,0,-1) = 8 or (1,0,-1) = 4 or
          (1,0,-1) = 9 )

             and (1,0,0) = 1 and remainder((1,0,1),4) != 1 }
rule : 4 1 { (0,0,0) = 0 and (-1,0,0) = 2 and (0,-1,0) = 0 and
          remainder((-1,0,1),4) = 1 and ((-1,0,-1) != 12 and (-1,0,-
          1) != 13 and (-1,0,-1) != 14 and (-1,0,-1) != 15 ) }

rule : 4 1 { (0,0,0) = 0 and (1,0,0) = 1 and (0,-1,0) = 0 and
          remainder((1,0,1),4) = 1
            and ((1,0,-1) != 12 and (1,0,-1) != 13 and (1,0,-1) != 14
          and (1,0,-1) != 15 )  }

rule : {(0,0,0)} 1 { ((-1,0,-1) = 1 or (-1,0,-1) = 8 or (-1,0,-1) = 4
          or (-1,0,-1) = 9 ) and (-1,0,0) = 2 and remainder((-
          1,0,1),4) != 1 }

 #EndMacro


#BeginMacro(MoveCheckRule_AfterLoading)

rule : 4 1 { (0,-2,0) = 0 and (0,-1,0) = 0 and (0,0,0) = 5 and (0,0,-1)
          = 10 and (0,1,-1) = 4 and remainder((0,1,1),4) = 2
            and remainder((0,0,1),4) = 1 and ((1,2,1) - trunc((0,0,1)/
          4 ) = 91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-1,-1) = 10 and
          remainder((0,0,1),4) = 2 and ((1,1,1) - trunc((0,0,1)/ 4 ) =
          91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-2,-1) = 10 and (0,-
          1,-1) = 4 and remainder((0,0,1),4) = 3 and ((1,0,1) -
          trunc((0,0,1)/ 4 ) = 91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-3,-1) = 10 and (0,-
          2,-1) = 4 and remainder((0,0,1),4) = 0 and ((1,-1,1) -
          trunc((0,0,1)/ 4 - 1 ) = 91 )}

rule : 4 1 { (0,-2,0) = 0 and (0,-1,0) = 0 and (0,0,0) = 5 and (0,0,-1)
          = 11 and (0,1,-1) = 9 and remainder((0,1,1),4) = 2
            and remainder((0,0,1),4) = 1 and ((1,2,1) - trunc((0,0,1)/
          4 ) = 81 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-1,-1) = 11 and
          remainder((0,0,1),4) = 2 and ((1,1,1) - trunc((0,0,1)/ 4 ) =
          81 )}
```

```
rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-2,-1) = 11 and (0,-
        1,-1) = 9 and remainder((0,0,1),4) = 3 and ((1,0,1) -
        trunc((0,0,1)/ 4 ) = 81 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-3,-1) = 11 and (0,-
        2,-1) = 9 and remainder((0,0,1),4) = 0 and ((1,-1,1) -
        trunc((0,0,1)/ 4 - 1 ) = 81 )}

rule : 4 1 { (0,-2,0) = 0 and (0,-1,0) = 0 and (0,0,0) = 5 and (0,0,-1)
        = 10 and (0,1,-1) = 4 and remainder((0,1,1),4) = 2
         and remainder((0,0,1),4) = 1 and ((-1,2,1) - trunc((0,0,1)/
        4 ) = 91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-1,-1) = 10 and
        remainder((0,0,1),4) = 2 and ((-1,1,1) - trunc((0,0,1)/ 4 ) =
        91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-2,-1) = 10 and (0,-
        1,-1) = 4 and remainder((0,0,1),4) = 3 and ((-1,0,1) -
        trunc((0,0,1)/ 4 ) = 91 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 4 and (0,-3,-1) = 10 and (0,-
        2,-1) = 4 and remainder((0,0,1),4) = 0 and ((-1,-1,1) -
        trunc((0,0,1)/ 4 - 1 ) = 91 )}

rule : 4 1 { (0,-2,0) = 0 and (0,-1,0) = 0 and (0,0,0) = 5 and (0,0,-1)
        = 11 and (0,1,-1) = 9 and remainder((0,1,1),4) = 2
         and remainder((0,0,1),4) = 1 and ((-1,2,1) - trunc((0,0,1)/
        4 ) = 81 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-1,-1) = 11 and
        remainder((0,0,1),4) = 2 and ((-1,1,1) - trunc((0,0,1)/ 4 ) =
        81 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-2,-1) = 11 and (0,-
        1,-1) = 9 and remainder((0,0,1),4) = 3 and ((-1,0,1) -
        trunc((0,0,1)/ 4 ) = 81 )}

rule : 4 1 { (0,0,0) = 5 and (0,0,-1) = 9 and (0,-3,-1) = 11 and (0,-
        2,-1) = 9 and remainder((0,0,1),4) = 0 and ((-1,-1,1) -
        trunc((0,0,1)/ 4 - 1 ) = 81 )}

rule : 5 1 { ((0,0,-1) = 10 or (0,0,-1) = 11 ) and (0,0,0) = 5 }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
        remainder((0,1,1),4) = 1 and (0,1,-1) = 4 }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
        remainder((0,1,1),4) = 1 and (0,1,-1) = 9 }

#EndMacro

#BeginMacro(MoveCheckRule_BeforeLoading)

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
        remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (1,2,-1) != 3
        and (2,-1,-1) != 3  }
```

```
rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (1,2,-1) = 3
           and ((1,2,1) - trunc((0,1,1)/4 ) != 91)}

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (1,2,-1) != 3
           and (2,-1,-1) = 3 and ((2,-1,1) - trunc((0,1,1)/4 ) != 91 ) }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (1,2,-1) != 5
           and (2,-1,-1) != 5  }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (1,2,-1) = 5
           and ((1,2,1) - trunc((0,1,1)/4 ) != 81)}

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (1,2,-1) != 5
           and (2,-1,-1) = 5 and ((2,-1,1) - trunc((0,1,1)/4 ) != 81 ) }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-1,2,-1) != 3
           and (-2,-1,-1) != 3  }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-1,2,-1) = 3
           and ((-1,2,1) - trunc((0,1,1)/4 ) != 91)}

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 1 and (-1,2,-1) != 3
           and (-2,-1,-1) = 3 and ((-2,-1,1) - trunc((0,1,1)/4 ) !=
           91 ) }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (-1,2,-1) != 5
           and (-2,-1,-1) != 5  }

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (-1,2,-1) = 5
           and ((-1,2,1) - trunc((0,1,1)/4 ) != 81)}

rule : 4 1 { (0,1,0) = 4 and (0,0,0) = 0 and (0,-1,0) = 0 and
           remainder((0,1,1),4) = 1 and (0,1,-1) = 8 and (-1,2,-1) != 5
           and (-2,-1,-1) = 5 and ((-2,-1,1) - trunc((0,1,1)/4 ) !=
           81 ) }
rule : {(0,0,0)} 1 { ((0,1,-1) = 1 or (0,1,-1) = 8 or (0,1,-1) = 4 or
           (0,1,-1) = 9 )
             and (0,1,0) = 4 and remainder((0,1,1),4) != 1 }

rule : 5 1 { (0,0,0) = 5 and (0,0,-1) != 0 }

#EndMacro
```

APPENDIX C: Example of Sources Code of DEVS Model (Plantser.cpp)

```
/** include files **/
#include "plantser.h"       // class Plantser
#include "message.h"     // class ExternalMessage, InternalMessage
#include "mainsimu.h"    // MainSimulator::Instance().getParameter( ... )
#include "distri.h"      // class Distribution
/** public functions **/

/*******************************************************************
* Function Name: Plantser
* Description:
*******************************************************************/
Plantser::Plantser(const string &name)
: Atomic (name)
, plantserin (addInputPort ("plantserin"))
, done (addOutputPort ("done"))
, plantout (addOutputPort ("plantout"))
 {
        dist = Distribution::create( MainSimulator::Instance().getParameter( description(),
        "distribution" ) );
        MASSERT( dist ) ;
        for ( register int i = 0; i < dist->varCount(); i++ )
        {
string parameter( MainSimulator::Instance().getParameter( description(), dist->getVar( i ) ) ) ;
                        dist->setVar( i, str2Value( parameter ) ) ;
        }

}

/*******************************************************************
* Function Name: initFunction
* Description: Resete Id new queue
* Precondition:
*******************************************************************/
Model &Plantser::initFunction () {
        curid = 11;
        holdIn(active, Time::Zero);
        return *this;
}

/*******************************************************************
* Function Name: externalFunction
* Description:
*******************************************************************/
Model &Plantser::externalFunction (const ExternalMessage &msg) {
        if (msg.port () == plantserin && msg.value ())  {
                std::cout << "Here is the plantserin port of plant server" << msg.time();
                if (msg.value () > 10){
                  std::cout << "(****Plantser External)the value received in plant server is
(id)" << msg.value()<<"at time: "<<msg.time()<< std::endl;
                  curid = msg.value();
                  holdIn(active, Time( static_cast< float >( fabs( distribution().get() ) ) ));
                }
        }
        return *this;
}

/***********************************************************************
* Function Name: internalFunction
* Description:
***********************************************************************/
Model &Plantser::internalFunction (const InternalMessage &msg) {
        passivate();
        return *this ;
}

/***********************************************************************
* Function Name: outputFunction
* Description:
***********************************************************************/
Model &Plantser::outputFunction (const InternalMessage &msg) {
        sendOutput (msg.time(), plantout, curid);
        std::cout << "(Plantser Output)Send out " << curid <<" to plantque at "<< msg.time()<<
std::endl;
        sendOutput (msg.time(), done, 1);
        std::cout << "(Plantser Output)Send out done to plantque at "<< msg.time()<< std::endl;
        return *this ;
}
Plantser::~Plantser()
{
        delete dist;
}
```

APPENDIX D: Using Patterns for Developing Rules

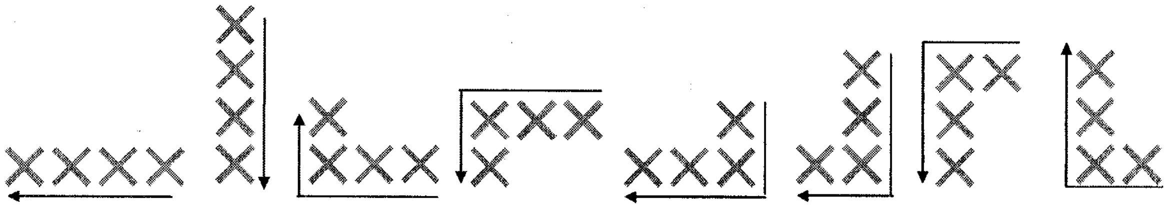
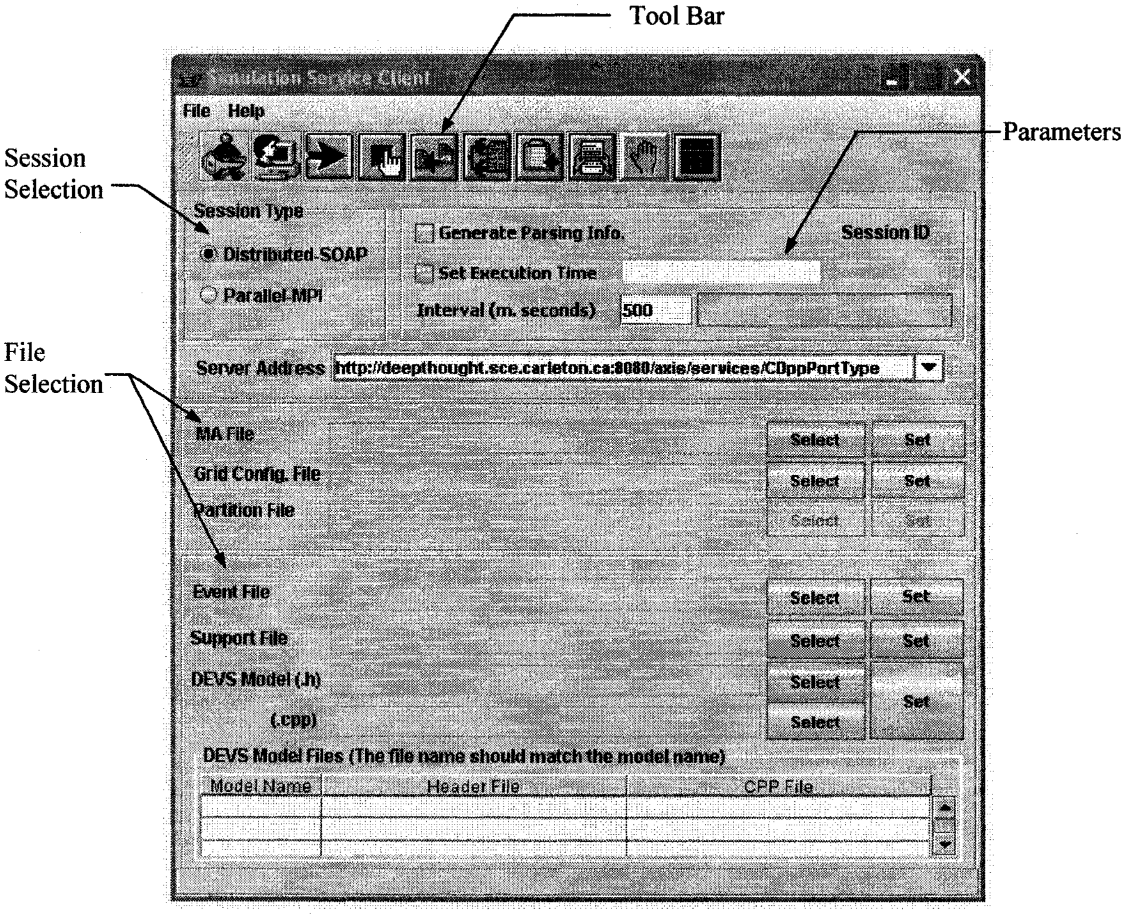


Figure E.1 Example of Patterns of 4-Cell Truck Used in Developing Reduced Turning Speed Rules

APPENDIX E: Introduction of DCD++ (distributed simulation)

The distributed simulation engine follows the conservative approach for synchronization among the nodes, and takes advantage of web service technologies in order to execute complex models using the resources available in a grid environment. In addition, it allows for the integration with other systems using standard web service tools. The performance of the engine depends on the network connectivity among the nodes; which can be commodity Internet connections, or dedicated point-to-point links created using User Controlled Light Path (UCLP). UCLP is a web service-based network management tool used by grid applications to allocate bandwidth on demand. Figure D-1 shows the Graphical User Interface of simulation service client (Madhoun and Wainer 2007).



Session Type

● Distributed-SOAP

○ Parallel-MPI

□ Generate Parsing Info.

Session ID

□ Set Execution Time

Interval (m. seconds) 500

Server Address http://deepthought.sce.carleton.ca:8080/axie/services/CDppPortType

MA File

Grid Config. File

Partition File

Event File

Support File

DEVS Model (.h)

(.cpp)

DEVS Model Files (The file name should match the model name)

Figure D.1 Graphical User Interface of Simulation Service Client

Steps:

(1) Select files such as the (.ma) files;

(2) Run the simulation;

(3) Save the output file ((.log) file).

System Requirements:

(4) Fedora Core with Linux kernel 2.4.21.

(5) Version 2.95.3 GCC package

(6) "Yacc", "ar", and "make" commands on Linux.

(7) A MPICH package (version 1.2.6, available at ftp://ftp.mcs.anl.gov/pub/mpi/old/)

(8) The Java runtime environment on the cluster at Carleton University is 1.5.0_04.

APPENDIX F: List of Publications

a. Articles in refereed journals

(1) Zhang, C., Hammad, A., Zayed, T.M., Wainer, G., and **Pang, H.** (2007). Cell-based Representation and Analysis of Spatial Resources in Construction Simulation, Automation in Construction, Vol. 16, No. 4, pp. 436-448.

b. Articles in refereed conference proceedings

(2) **Pang, H.**, Zhang, C., and Hammad, A. (2006). Sensitivity Analysis of Construction Simulation Using Cell-DEVS and MicroCYCLONE, Proceedings of 2006 Winter Simulation Conference, IEEE, Monterey, California, pp. 2021-2028.

(3) **Pang, H.**, Zhang, C., and Hammad, A. (2007). Sensitivity Analysis of Construction Simulation Considering Site Layout Patterns, International Workshop on Computing in Civil Engineering, ASCE, Pittsburgh, Pennsylvania, July 2007 (Accepted).