A Framework for Interoperability of Virtual Reality and Simulation Models in Support of Training

by

Ahmed Ahmed Abdalwhab Sayed Ahmed, M. Sc.

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering (OCIECE)

Department of Systems and Computer Engineering

Carleton University

Ottawa, Ontario, Canada, K1S 5B6

November 2011

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-87752-4

Our file Notre référence
ISBN: 978-0-494-87752-4

Abstract

Computer-based Modeling and Simulation (M&S) is a powerful tool for cost-effective analysis, design, control, and optimization of complex dynamic systems. One of the most advanced general purpose (M&S) frameworks is the Discrete Event System Specification (DEVS) formalism. Cell-DEVS combines DEVS with cellular models, allowing complex systems to be described using simple rules. 3D visualization methods (such as virtual reality (VR) or computer games) provide support to simulation studies; however, at present there is no way to integrate 3D visualization for DEVS and Cell-DEVS (using interactive, collaborative platforms). Likewise, developing and modifying scenarios in existing VR environments usually requires significant efforts in programming and validation.

In order to solve the aforementioned problems, this thesis focuses on the definition, design, and performance analysis of the visual Cell-DEVS (VCELL) framework, which allows different simulation models to receive real-time data to interact, collaborate, and adapt to simulation events (integrating 3D visualization, sensor data, DEVS, and Cell-DEVS modeling and simulation), which improves the models' design.

As a proof of concept, we applied VCELL to different applications, including building information modeling (BIM), emergency and disaster simulation, and land combat simulation.

BIM is used to generate and manage data for buildings during the project life cycle. Existing BIM applications include models for indoor climate, energy consumption,

and CO2 emissions. However, they do not take into consideration other problems. This research shows a more generic environment for Cell-DEVS and BIM integration, and a prototype implementation in the form of BIM for Cell-DEVS simulation and visualization.

In emergency and disaster simulations, it is usually important to consider the system evolution in time and space. Generally, such simulations are large-scale programs, which in turn raise the need for efficient simulation engines. However, some of these emergency simulations do not have real-time input data and are not adaptive. VCELL solves these problems, allowing the emergency simulation to be integrated with 3D visualization in real time.

In the area of land combat simulation, agent-based distillation (ABD) provides a method for studying different land combat behaviors, which helps with decision-making. ABD movement algorithms are used to simulate the target's movement in the battlefield by a large set of parameters. However, under some circumstances, these movement algorithms use random movements, which may result in an unpredicted simulation output. We propose to use a model that integrates a cellular agent model and a collaborative 3D visual agent model in real time. VCELL allows the randomization problem to be solved, providing more stable output and reducing the scenario development, modification, and validation time.

The main ideas, design, and implementation of VCELL are discussed, and the case studies are presented in detail.

Acknowledgments

First and foremost, I would like to express my deepest thanks to Prof. Gabriel A. Wainer for his guidance, support, and friendship. To me he is the role model of an exceptional scientist and teacher. He has always been supportive, not only when I made progress but also when I made mistakes. I have also learned from him how to be professional in an academic career. It has been my great privilege to study my PhD under his guidance.

I am also grateful to Prof. Samy A. Mahmoud for generously sharing his intelligent ideas and comments, which are constant sources of encouragement. I also greatly appreciate his friendship and empathy.

Lastly, I would like to thank my mother, my wife, and my kids for their love, support, patience, cooperation, and understanding throughout the course of my studies.

Contents

List of Tables

List of Figures

List of Acronyms

Acronyms	Definition
ABD	Agent Based Distillation
ABS	Agent Based Simulation
AEC	Architecture, Engineering, and Construction industry
API	Application Programming Interface
BIM	Building Information Modeling
CAS	Complex Adaptive System
COTS	Commercial off-the-shelf
CROCADILE	Conceptual Research Oriented Combat Agent Distillation Implemented in the Littoral Environment
DCD++	Distributed CD++
DEVS	Discrete Event Simulation
EINSTein	Enhanced ISAAC Neural Simulation Toolkit
IAI	International Alliance for Interoperability
IES	Interactive Environment System
IFCs	Industry Foundation Classes

IG	Image Generator
IL	Intermediate Language
ISAAC	Irreducible Semi-Autonomous Adaptive Combat
M&S	Modeling and Simulation
MANA	Map Aware Non-uniform Automata
MAS	Multi Agent System
RTV	3D Real Time Visualization
WISDOM	Warfare Intelligent System for Dynamic Optimization of Missions
VCELL	Visual CELL-DEVS
VR	Virtual Reality

Chapter 1

Introduction

Three-dimensional (3D) visualization methods (such as Virtual Reality (VR) or computer games) have changed the military training of soldiers for successful results in wartime missions [1]. VR can be defined as a computer-generated surrounding, an interactive 3D computer graphical interface, or an immersive interactive environment [2]. Training simulation systems can be treated as games; hence, significant effort has recently been devoted to trainee learning and improving their skills [3]. Digital Game-Based Learning [4], a learning style used currently and that will also be used in the future, is deployed as a method of learning and training, because it is motivating and effective when used correctly. Simulators have become a powerful tool that is changing military training. Simulators are used not only to teach troops how to use complex equipment, but also how to work efficiently in teams. Simulation also gives military decision-makers a strategic overview of options before engaging in real combat. In addition, they can estimate the performance of new weapons systems under consideration [1]. However, for military tactical training simulation, developing or modifying different scenarios in an existing virtual environment at the code level requires significant programming time and validation efforts.

These 3D visualization methods are also popular in various applications ranging from training to construction. For instance, if 3D visualization and simulation are

combined with building information modeling (BIM), this could result in increased productivity in building design and construction [5]. BIM is a technique used to generate and manage building data during the life cycle of a building's construction, which allows for better and more accurate construction projects with minimized financial cost [6]. Because BIM facilitates the coordination, cooperation, and maintenance of the building life cycle, many research studies have included simulation [7–11]. Nevertheless, most of these have focused only on thermal indoor climate, energy consumption, CO2 emissions, and other environmental aspects. In addition, no existing research has applied generic modeling and simulation (M&S) methods to BIM. Doing so would permit different studies to be carried out in the areas of building evacuation, fire spreading, and so on. Moreover, there are no reactive simulation systems that can interact in real time with changes to building parameters

Another popular use of 3D visualization and simulation is analysis of emergency situations. This enhances training preparation, allowing decision-makers to investigate different scenarios. In emergency simulation, M&S techniques play an important role when it is impractical to perform real-world studies due to the destructive impact of emergencies. Generally, such simulations are large-scale programs, which in turn raise the need for efficient simulation engines. Research studies [3, 12–18] state that some of these emergency simulations have 3D visualization, but they do not incorporate real input data from the field and are not adaptive in real time.

VR is usually combined with computer-based M&S as a powerful tool for cost-effective analysis, design, control, and optimization of complex dynamic systems. One of the most advanced general-purpose M&S frameworks is the Discrete Event System Specification formalism (DEVS) [19–22]. DEVS facilitates the reuse of tested models and improves the safety of the simulations. As a result, this reduces development time. The cellular DEVS (Cell-DEVS) [23] approach combines DEVS and cellular models, which allows complex systems to be described as a cell space using simple rules for

modeling. Existing DEVS and Cell-DEVS tools usually include 2D visualization to facilitate analyzing the output results on a 2D grid map, but 3D visualization and VR for DEVS and Cell-DEVS in real time are not available. Although surveys [3, 24, 25] show that 3D visualization can be applied to DEVS and Cell-DEVS, existing efforts are not interactive, collaborative, or adaptive.

Considering these issues, the main objective of this research is to develop a framework that allows different simulation models to receive real-time external parameters so that they can be interactive, collaborative, and adaptive to simulation events by integrating the DEVS, Cell-DEVS, and 3D VR with different modeling techniques.

1.1 Thesis Contributions

The central theme of this thesis is to develop a framework that allows different simulation models to receive real-time external parameters so that they can be interactive, collaborative, and adaptive to simulation events by integrating the DEVS, Cell-DEVS, and 3D visualizations with different modeling techniques, which improves simulation interoperability at the software level, which enhances the simulation results. Our objective is to develop a generic framework to be applied to simulation environments and applications. This thesis presents a simulation adaptation and interoperability methods using DEVS, Cell-DEVS, and 3D visualization. We use BIM simulation, emergency simulation, and land combat simulation as case studies. The key contributions are summarized below:

- The design and development of the framework, which is the first existing framework to be based on DEVS, Cell-DEVS, and 3D visualization simulation principles. The framework's key features are summarized as follows:
  - The framework serves as a container to hold different software components

without being specific to any implementation, which allows new components to be added without making changes at the code level or in the framework architecture.

  - Using the framework for adding various hardware devices allows hardware-in-the-loop, which enhances the simulation results by achieving more accurate results than when simulated hardware is used, and also improves training in different domains (such as in military training) by allowing the use of real equipment.
  - The framework can be used not only for real-time simulation, but also in virtual time, which allows for the enhancing of the simulation results by comparing the real-time and virtual-time output results, and hence modifying the simulation rules.
  - The framework can use predefined inputs, actual inputs from the simulated system, or a combination of both predefined and actual inputs, which gives flexibility for modifying the simulated system to get accurate results.

  The framework combines a Cell-DEVS cellular model, a DEVS-based controller, and VR visualization. This component-oriented approach provides model reusability and interoperability, allowing any of the components to be integrated or replaced.

- The RT cellular simulation is an on-demand data source of the scenario that is to be used by the DEVS model. A hardware-in-the-loop can be used to update the Cell-DEVS simulation data with a real situation and the information on the simulation space area.
- An interface is developed between DEVS, Cell-DEVS, and 3D visualization simulations so that various instances of each domain can cooperate with each other to simulate the same simulation session. In this case, the required simulation is performed within the framework. This means that the simulation can be

manipulated in real time.

As discussed earlier, the framework proposed in this research was applied to BIM simulation, emergency simulation, and land combat simulation. These applications are described below.

a) BIM Simulation

Most current BIM simulations do not cover our requirements, not only with regard to the prediction of the construction and building life cycle but also in the maintenance cycle. In this research, we focus on generalizing BIM simulations to overcome these limitations. We also focus on developing a simulation-driven architecture for integrating Cell-DEVS simulation with BIM on the simulated building in real time. A 3D visualization sub-system is then developed to present the output simulated results of Cell-DEVS on a BIM model.

b) Emergency Simulation

Since emergencies are processes that are distributed over both time and space, emergency and disaster simulations should take into consideration the system evolution in both time and space. In this research, we propose an integrated emergency management system based on DEVS and Cell-DEVS to develop new classes of models for emergency response applications. Cell-DEVS allows models to receive external information, and the simulation parameters can be updated at any time due to the continuous-time nature of the discrete-event specifications. The proposed emergency simulation integrated with emergency management is based on the collaboration of DEVS, Cell-DEVS real-time simulations with hardware-in-the-loop, and 3D real-time visual simulation. The emergency simulation is based on Cell-DEVS, and the emergency management uses a robotic agent controlled by a DEVS model to respond to the emergency in real time. We also use a 3D visualization engine that takes the results of the emergency simulation and the emergency management to be visualized in 3D in real time. Moreover, we propose a method to integrate Google Earth free

virtual reality web services with the visualization engine injecting the terrain data into the Cell-DEVS engine in real time.

c) Land Combat Simulation

Defense analysts have studied different behaviors of land combat battlefields based on agent-based distillations (ABD), which provide them with a useful tool for decision-making. ABDs are the most popular method and can be used to explore different aspects of land combat operations and help to quickly investigate different scenarios in real battle conditions. Although ABDs' movement algorithms are used to simulate the target's movement in the battlefield by a large set of parameters related to the battlefield, they move randomly, which may result in unpredicted simulation outputs. In addition, most existing ABDs include only 2D visualization facilities (with no 3D visualization scene). Although VR and computer games are used for military tactical training simulation, it requires significant programming time and validation efforts to develop or modify different scenarios in an existing virtual environment at the code level. In this research, we propose to deal with these problems using our framework, which integrates a cellular agent model and a collaborative 3D visual agent model in real time to solve the randomization problem of the ABDs' movement algorithms and also reduce the development time required for programming different scenarios. We also propose to integrate a hardware interface for our model, which facilitates the building of training simulators based on human-in-the-loop. The proposed human-in-the-loop training simulator will be designed, built, and used to train a tank crew in a virtual simulation environment. This will provide reality in training, by using real equipment, or a manufactured replica of it.

Parts of the thesis have appeared in the following publications:

- Ahmed Sayed Ahmed, Gabriel A. Wainer, and Samy Mahmoud. "Integrating Building Information Modeling & Cell-DEVS Simulation". Proceedings of 2010 Spring Simulation Conference (SpringSim10), Orlando, USA. April 2010.

- M. Moallemi, S. Jafer, A. Sayed Ahmed, and G. A. Wainer. "Interfacing DEVS and Visualization Models for Emergency Management". Proceedings of 2011 Spring Simulation Conference (SpringSim11), Boston, USA. April 2011.
- Ahmed Sayed Ahmed, M. Moallemi, Gabriel A. Wainer, and Samy Mahmoud. "VCELL: A 3D Real-Time Visual Simulation in Support of Combat". Proceedings of 2011 Summer Computer Simulation Conference (SCSC11), The Hague, Netherlands. June 2010

The last publication was awarded the second place Best Paper Award in recognition of its quality, originality and significance in modeling and simulation.

1.2 Thesis Organization

This thesis is organized as follows:

Chapter 2 provides detailed background on DEVS and Cell-DEVS formalism, BIM, complex adaptive systems (CAS) and ABD, and 3D real-time visualization.

Chapter 3 gives an overview of the VCELL framework and presents its architecture. It focuses on integrating DEVS, Cell-DEVS, and 3D real-time visualization simulation in virtual and real time.

Chapter 4 focuses on integrating Cell-DEVS simulation with the BIM model based on a collaboration of Cell-DEVS and BIM simulation in virtual time. We present the development of an Interactive Environment System (IES_Revit). It shows a Cell-DEVS/BIM integration system and describes a prototype implementation in the form of a BIM add-in tab for Cell-DEVS simulation, and then visualizes the output simulation of Cell-DEVS on the BIM model. The diffusion limited aggregation (DLA) model example is used to verify the feasibility of combining these two technologies using IES.

Chapter 5 focuses on integrating emergency simulation with emergency management based on the collaboration of DEVS, Cell-DEVS, and 3D real-time visualization simulation in real time. It discusses the modifications made to the CD++ simulation engine to enable real-time execution. We present the emergency management mechanism using a DEVS-based robotic agent and explain the visualization of the simulation. The message structure transferred between the three components is presented.

Chapter 6 describes a collaborative 3D real-time visual cellular agent model (VCELL) and a cellular agent simulation in real time. The architecture of the visual CELL-DEVS agent model is presented. This chapter also discusses the implementation of CellAgent and the 3D real-time visual simulation sub-models of land combat. The global message structure is illustrated.

Chapter 7 concludes this thesis by outlining major findings. Some future work is also suggested.

Chapter 2

Literature Review

This chapter presents a literature review of the proposed work. Section 2.1 gives background on VR and simulation. Section 2.2 describes the DEVS and Cell-DEVS M&S framework and its implementation in the CD++ environment. Section 2.3 introduces BIM. Emergency simulation is reviewed in Section 2.4. ABD for combat is presented in Section 2.5.

2.1 Virtual Reality and Simulation

In the past, simulators and other simulation applications were only available to industrial and military systems, due to the system requirements of high technology, which were expensive [26]. In recent years, the rapidly expanding technology in computer processing, storage, communications, and display capability has made it possible to run simulators and other simulation applications on personal computer hardware [27] because PCs now have advanced computational capabilities at very low cost [28]. This can be seen in the hardware of the current generation of video game consoles, such as Sony's PlayStation 3, Nintendo's GameBox, and Microsoft's Xbox [29–31].

In recent years, serious games, simulation games, training simulators, and VR have gained popularity in computer game technology. Serious games are the result of

applying simulation technology for training purposes [32], while simulation games are the result of applying simulation technology for entertainment purposes [33]. Training simulators, such as combat marksmanship simulators, flight simulators, and driving simulators [34–36], are designed to develop the skills or experience of the trainees who use them and to maximize their performance. VR technologies are used to enhance the immersiveness and interactivity of the training simulator [36].

Most military simulators are based on various types of military equipment, such as aircraft simulators, tank simulators, and Marine simulators. These simulators not only improve and enhance military training but also reduce training costs and save trainees' lives by giving them training in situations that would be too dangerous to execute in reality. Due to the cost-effectiveness of computer game technology, it is used in various armed-forces simulators. 3D games and serious games use these simulation games for military purposes, such as America's Army [37]. Also, several commercial off-the-shelf games, such as Delta Force 2 and Steel Beasts [38, 39], are used by armed forces to improve military training [40].

Burdea and Coiffet define VR as 3D, immersive, and real time simulations of an environment that can be interactive with users via multiple sensorial channels [41]. A VR application is based on various components: *the scene and the objects*, *behaviors*, *interaction*, *communication*, and *sound* [42]. With regard to *the scene and the objects*: the scene presents the environment that contains 3D objects, which represent various 3D models such as terrain, vehicles, or trees. It also includes lighting, observers, collision detection, and environmental and special effects. *Behaviors* are the properties of the objects, such as moving, rotating, and scaling. *Interaction* can be defined as the action and reaction between the user and the virtual world, carried out through various hardware devices such as mouse, keyboard, 3D mouse, data gloves, or head-mounted devices. *Communication* means that the VR applications are collaborative environments in which remote users can interact with each other.

Sound represents various sound effects in the VR application of the scene and objects.

The use of immersive displays and desktop displays for various tasks in VR offer difficulties with regard to making a choice for specific tasks [43]. A great deal of research has been done to compare the usability of immersive and desktop displays [44–46] but immersive displays have been shown to be more advantageous.

Many VR software systems and VR toolkits have been proposed for the development of VR applications. These software tools are classified as follows:

- **Authoring tools** have a graphical user interface (GUI) which is used for creating the scene and objects using various scripting languages. The developers should have some knowledge of VR. The most frequently used authoring tools are Autodesk 3dMax [11], Autodesk Maya [47], MilkShape 3D [48], PlanIt 3D [49], AC3D [50], and Blender [51].
- **Software Programming Libraries**: in these, we can programme a VR application from scratch using a programming library for VR, such as Java3D [52], VRML [53], OpenGL [54], and X3D [55]. To use such a library, the developers should have a good knowledge of programming, VR, and computer graphics.

2.1.1 Virtual Reality Simulation in the Military

VR can be defined as a computer-generated surrounding, an interactive 3D computer graphical interface, or an immersive, interactive environment [2]. VR and computer games are modifying military preparation for war to an extreme degree [1]. The use of simulators has become a powerful tool that has changed military training. Simulators are used not only to teach troops how to use complex equipment, but also teach them how to work efficiently in teams. Simulation also gives military decision-makers a strategic overview of options before starting real combat. They can also evaluate the performance of new weapons systems under consideration [1].

The rapidly expanding technology in computer processing, storage, communications, and display capability has resulted in the rapid growth of software modeling and visual simulation [56]. Interactive 3D simulation is used in combat simulation, where it is necessary during peacetime to train soldiers for successful results in wartime missions [57]. Training simulation systems can be treated as games; hence, significant effort has recently been devoted to trainees' learning and improving their skills [58]. Digital Game-Based Learning [4], a learning style used currently and in the future, is deployed as a method of learning and training, because it is motivating and effective when used correctly.

The components used in a visual simulation are:

- A real-time application program, which controls the graphical scene, model dynamics, collision detection, and various special effects. A real-time virtual environment development uses a computer graphics library/application programming interface (API)/language such as OpenGL, Java3D, and VRML, which can be used with a common programming language, such as C++, Java, or Python.
- An image generator (IG), which is the graphical hardware responsible for drawing the scene on PCs and gaming consoles.
- The visual database is the data that describe what, when, and how to draw the scene.
- A modeling package, which creates visual databases that represent different 3D objects in the scene.

Many VR software systems and VR toolkits have been developed for VR applications, such as VPLs Body Electric [59], Alice [60], CAVElib [61], VRJuggler [62], DIVERSE [63], the Studierstube project [64], and MRToolkit [65]. Many Internet-based PC games were developed by the Moves Institute, under the auspices of the US Army [37]. The armed forces are looking for simulation solutions for training

that provide a high level of realism and interactivity in a visual representation. In a real-time visual simulation of a battlefield, tactical scenarios have to be designed for the enemy side. Implementation of these scenarios takes a long time, in terms of programming and verification. It may also take a long time to modify these scenarios for different situations.

2.2 DEVS and Cell-DEVS

2.2.1 The DEVS Modeling and Simulation Formalism

The Discrete Event Systems Specification (DEVS) is an M&S formalism that allows us to define hierarchical modular models [66]. DEVS M&S theory is based on systems theory concepts [19, 67, 68]. DEVS modular facilitates the reuse of tested models, improving the safety of the simulations. As a result, this can reduce development time. DEVS is a framework for constructing discrete-event hierarchical modular models, in which behavioral models (atomic) can be integrated, forming a hierarchical structural model (coupled). The atomic DEVS model is defined as in (2.1) and in Table 2.1.

$$M =< X,\ Y,\ S,\ \delta_{int},\ \delta_{ext},\ \lambda,\ ta > \tag{2.1}$$

Table 2.1: Explanation of Atomic DEVS Model Equation Variables

Variable	Definition
X	The set of input events
Y	The output events set
S	The set of sequential states
δ_{int}	The internal state transition function
δ_{ext}	The external state transition function
λ	The output function
ta	The time advance function

Fig. 2.1 shows the description of a DEVS atomic model. In the case of the absence of external events; if the DEVS model is in state $s \in S$ at a given time, an output value λ(s) is invoked at port y after ta(s) has finished and then the s state changes to δ_{int}(s). An internal transition occurs because of the consumption of time ta(s). When an external transition takes place, a state transition occurs. The new state transition is given by δ_{ext}(s, e, x) where s is the current state, e is the time elapsed since the last transition, and x is the external event that has been received.

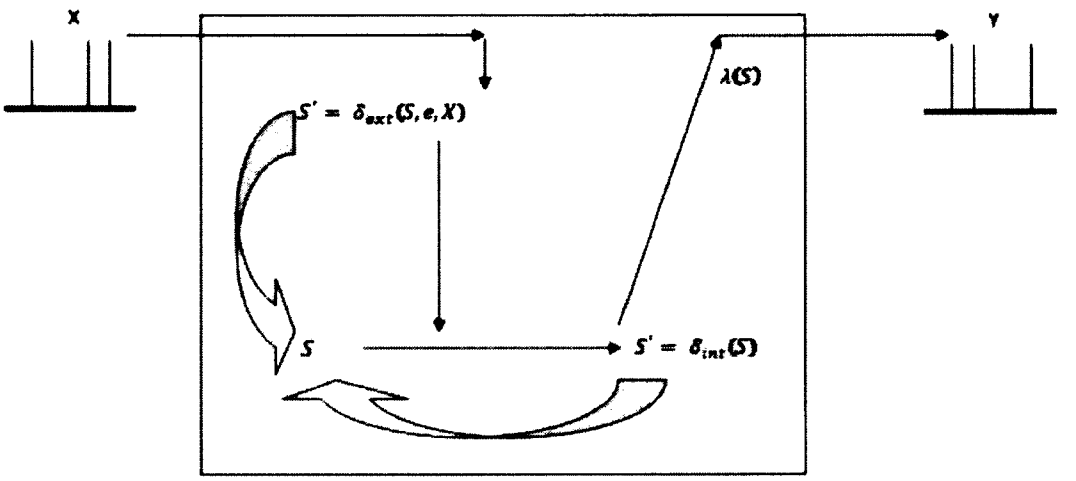


Figure 2.1: Discrete Event Systems Specification (DEVS) Atomic Model semantics [69]

The atomic model can be considered as the base element in which we define dynamics of any system, while the coupled structural model consists of one or more atomic and/or coupled models. Coupled models are defined as a set of basic components (atomic or coupled). Fig. 2.2 shows a description of the DEVS coupled model. The coupled model can be defined as in equation (2.2). The variables used are defined in Table 2.2

$$CM =< X,\ Y,\ D,\ \{M_d \mid d \in D\},\ EIC,\ EOC,\ IC,\ select > \qquad (2.2)$$

Table 2.2: Explanation of Coupled DEVS Model Equation Variables

Variable	Definition
X	The set of input events
Y	The set of output events
D	The set of component names and for each $d \in D$
M_d	A DEVS basic model(i.e., atomic or coupled)
EIC	the set of external input couplings
EOC	the set of external output couplings
IC	the set of internal coupling
select	the tie-breaker function

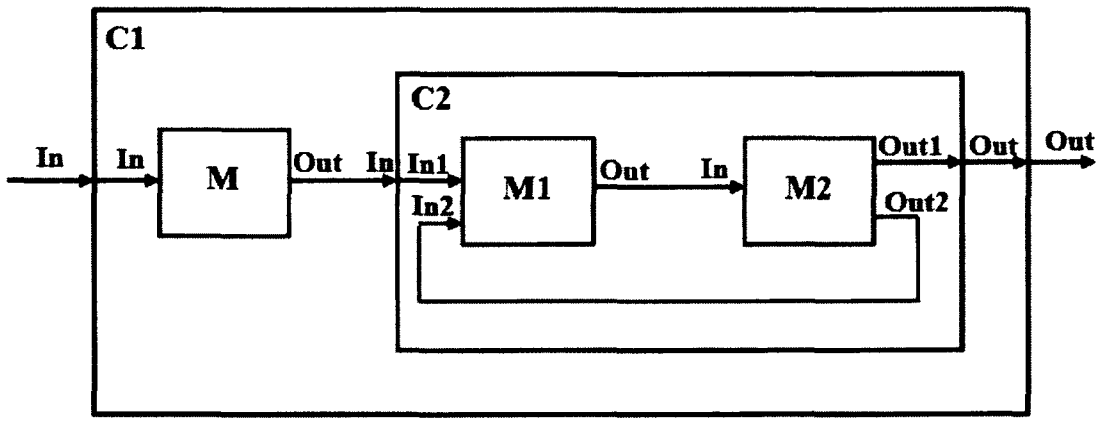


Figure 2.2: Discrete Event Systems Specification (DEVS) Coupled Model [69]

The coupled model explains how to convert the outputs of a model into inputs for the other models, and how to handle inputs/outputs to and from external models.

In recent years, the DEVS formalism [70] has gained popularity for modeling a variety of problems [71]. Various DEVS-based simulation tools have been implemented, such as DEVS-C++ [72], DEVS/HLA [73, 74], DEVSJAVA [75], and JDEVS [76]. DEVS was also implemented in the CD++ toolkit [77]. The CD++ toolkit is an open-source, object-oriented M&S environment that implements DEVS and Cell-DEVS formalisms.

2.2.2 Cell-DEVS Modeling and Simulation Formalism

Different formalisms have been used to capture the behavior of the systems that can be represented as cell spaces. These systems can be found in many fields, from chemistry to engineering and from physics to social sciences [78, 79]. Cellular automata are known formalisms that present these types of systems [80, 81]. Cellular automata were established by von Neumann [82] to study self-reproducing systems. A cellular automaton is a discrete model that is composed of a network of cells where each cell has a finite number of states [79]. The state of each of the cells in time t is a function of states of its predefined neighbor cells in time t-1. Cell-DEVS [23, 69] has extended the DEVS formalism, allowing us to implement cellular models with timing delays. Once the behavior of a cell is defined, a coupled Cell-DEVS can be created by interconnecting a number of cells with their neighbors.

A Cell-DEVS model is a lattice of cells, where each cell is a DEVS atomic component, holding state variables and a computing apparatus, which is in charge of updating the cell state according to a local rule-base. This is done using the current cell state and those of a finite set of nearby cells (called its neighborhood). Cell-DEVS improves execution performance of cellular models by using a discrete-event approach. It also enhances the cell's timing definition by making it more expressive. Each cell is defined as a DEVS atomic component, and it can later be integrated to a coupled component representing the cell space. Cell-DEVS atomic components are

informally defined as in Fig. 2.3.

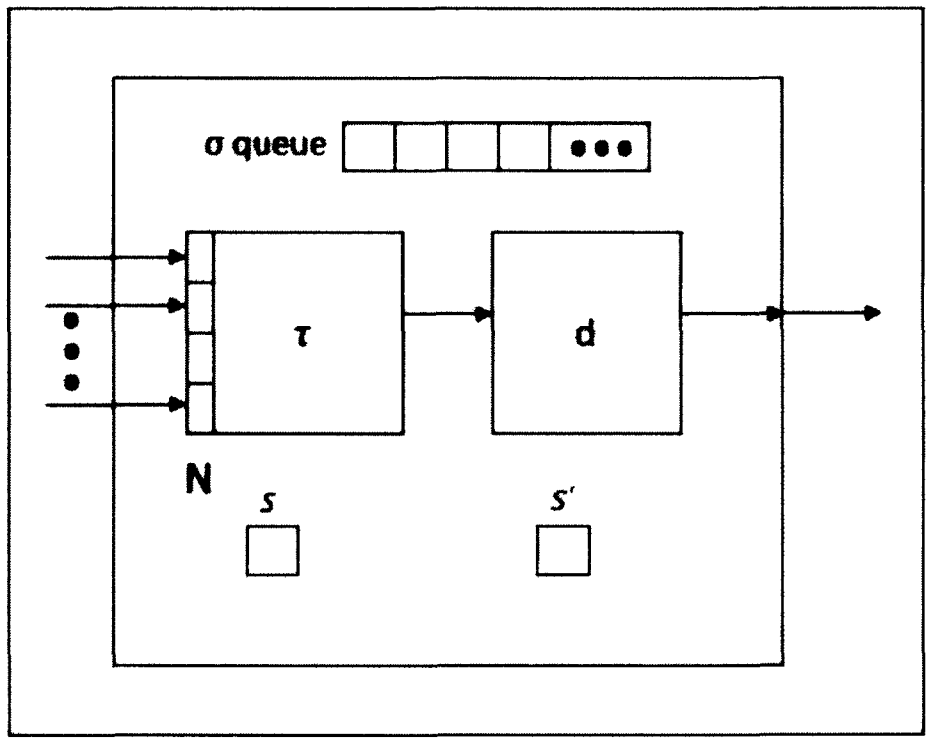


Figure 2.3: Description of a Cellular Discrete Event Systems Specification (Cell-DEVS) Atomic Component [23]

Each cell uses N inputs to compute its next state. These inputs, which are received through the model's interface, activate a local computing function (τ). A delay (d) can be associated with each cell. The state (s) changes can be transmitted to other models, but only after the consumption of this delay. This model can be formally described as in equation (2.3). The used variables are defined in Table 2.3

$$TDC =< X,\ Y,\ S,\ N,\ type,\ d,\ \tau,\ \delta_{int},\ \delta_{ext},\ \lambda,\ ta > \tag{2.3}$$

Table 2.3: Explanation of Atomic Cell-DEVS Model Equation Variables

Variable	Definition
X	The set of input external events
Y	The set of output external events
S	The set of states
N	The set of input values
d	The delay for the cell
type	The type of delay (transport/inertial/other)
τ	The local computing function
δ_{int}	The internal state transition function
δ_{ext}	The external state transition function
λ	The output function
ta	The state's lifetime function

Once the cell behavior is defined, a coupled Cell-DEVS can be created by putting together a number of cells interconnected by a neighborhood relationship. A Cell-DEVS coupled model is informally presented in Fig. 2.4. A coupled Cell-DEVS model is the resulting array of cells (atomic models) with given dimensions, borders, and zones (if applicable). Each cell is connected to its neighborhood through standard DEVS input/output ports.

The coupled Cell-DEVS model can be formally described as in equation (2.4). The variables used are defined in Table 2.4

$$TDC =< X,\ Y,\ Xlist,\ Ylist,\ \eta,\ N,\ m,\ n,\ C,\ B,\ Z,\ select > \qquad (2.4)$$

Table 2.4: Explanation of Coupled Cell-DEVS Model Equation Variables

Variable	Definition
X	The set of input external events
Y	The set of output external events
X*list*	The list of input coupling
Y*list*	The list of output coupling
η	The neighborhood size
N	The neighborhood set
m, n	The size of the cell space
C	The cell space set
B	The border cells set
Z	The transition function
select	The tie breaking selector function

Cell-DEVS were implemented using CD++. CD++ has solved a variety of complex problems [83–85]. The basic features of the CD++ toolkit can be shown in an example of an application. A maze-solving algorithm, defined in [86], is used as an example of the application of such models [87], and is shown below:

```
[top]
components : maze
[maze]
type : cell
dim : (15, 15)
delay : transport
defaultDelayTime  : 100
border : nowrapped
neighbors :              maze(-1,0)
neighbors : maze(0,-1)   maze(0,0)   maze(0,1)
neighbors :              maze(1,0)
initialvalue : 0
initialcellsvalue : maze.val
localtransition : maze-rule
```

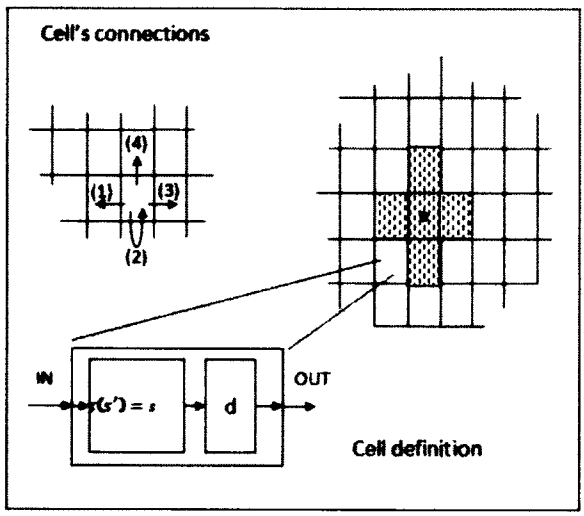


Figure 2.4: Description of a Cellular Discrete Event Systems Specification (Cell-DEVS) Atomic Component [23]

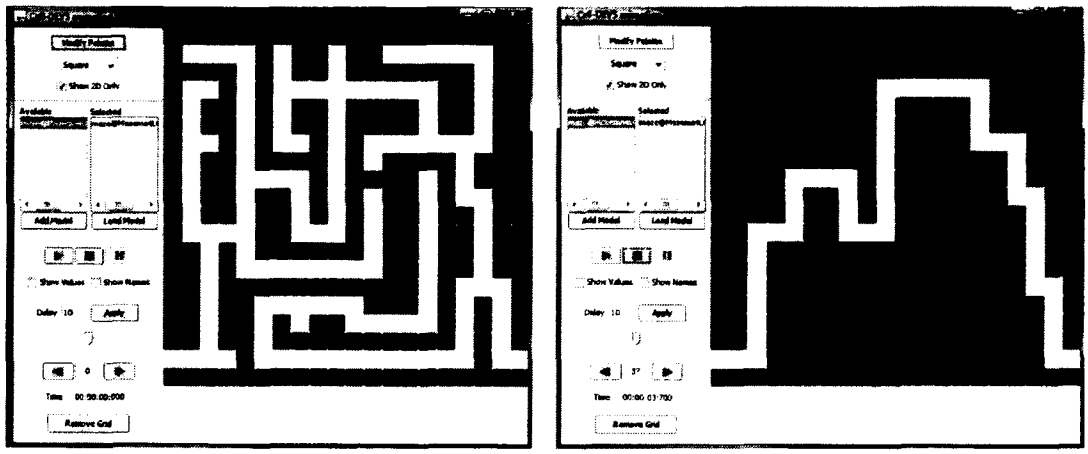
```
[maze-rule]
rule : 1 100 { (0,0) = 0 and (truecount = 3
        or truecount = 4) }
rule : 0 100 { (0,0) = 0 and truecount < 3 }
rule : 1 100 { t }
```

In the maze example, the rules are as follows:

- If the cell is a wall cell, the cell remains a wall cell.
- If the number of neighborhoods of a cell is three or more, the cell becomes a wall cell.

When the maze model is executed using these rules, all non-solution paths in the maze are closed successfully. One example of the initial cell state to the maze and the final steady state of the given initial maze cells is shown in Fig. 2.5, which is drawn using CD++ Modeler (a GUI included with the tool) [88]. Cell-DEVS was used to solve different problems in construction and architecture projects [89, 90].

The construction and architecture model shows a space representation and conflict analysis during construction, but there is no visualization used in this model. 3D visualization simulation [91, 92] was used only to visualize different problems that were implemented in Cell-DEVS.



(a) (b)

Figure 2.5: (a) Initial State of Maze (b) Final State of Maze

Cell-DEVS was also used to solve complex problems in a battlefield on land. A land battlefield model was introduced by Madhoun and Wainer [93, 94]. The model describes two armies engaging on a battlefield. Each army consists of a number of soldiers defending their flag or attacking the enemy's flag. The soldiers move on the battlefield according to a simple rule to obtain the enemy's flag. The movement is made by comparing the current cell position of the soldier and the enemy's flag position only, and no consideration is given to neighboring friendly and enemy soldiers. Therefore, the output results cannot be guaranteed.

2.3 Building Information Modeling (BIM)

For successful construction projects, an enormous amount of data should be collected and analyzed in the pre-design phase of the project. Until recently, the success of this phase was dependent on the experience of experts. However, the information required to complete the project and the amount of data that must be analyzed to do so is now greater and more complex. Therefore, there is a need for tools that can directly support this pre-design phase. BIM has been considered as a tool that can support this part of the construction project [95]. BIM has resulted in improvements to the way architects-contractors and fabricators work [96].

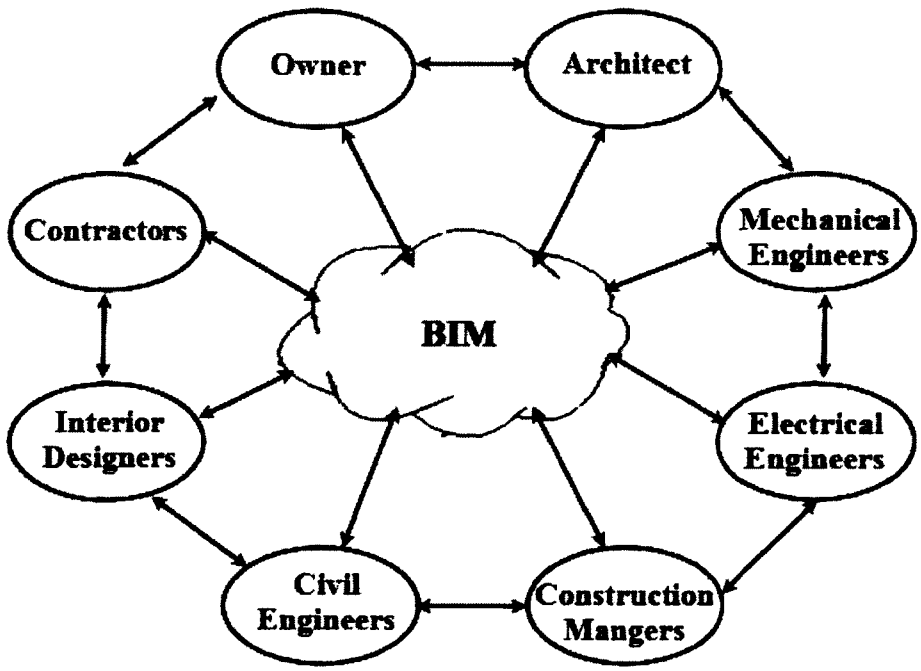


Figure 2.6: Building Information Modeling Life Cycle [97]

BIM is the process of generating and managing building data during the life cycle of the building project [5], as shown in Fig. 2.6. BIM uses 3D, real-time,

dynamic building modeling software to increase productivity in building design and construction. BIM allows us to achieve better and more accurate construction projects with minimized financial costs [6]. BIM also facilitates coordination and cooperation and supports the easy maintenance of a building life cycle.

BIM software creates parametric 3D models instead of 2D perspective drawings and operates on a digital database where any change made to this database will be reflected in the whole drawing that is produced. BIM software is often associated with industry foundation classes (IFCs), which are data structures used to represent information used in BIM. IFCs were developed by the International Alliance for Interoperability [98]. The IFCs are based on the Standard for the Exchange of Product (STEP) [99]. This model data (STEP) is a way to exchange product data information. IFCs are supported in most of the current architecture software and it is recognized as a valuable means of providing interoperability and better integration of the life cycle process of buildings. The concept of interoperability in BIM is that all the data associated to a building can be handled in one repository, which facilitates exchange between different domains in the architecture, engineering, and construction industries, by having a centralized data model accessible to different domain applications [100]. The interoperability encloses the integration of both the tools for design and analysis, and the multiple domains of expertise in the building life cycle process [101]. BIM is considered to be an important improvement to the way architects-contractors and fabricators work [96], in that it allows for conflicts between them to be minimized, and presents a 3D visualization of the building during design and fabrication. Therefore, errors made by the design team can be minimized, resulting in reduced costs.

There are different simulation applications for BIM, such as IDA Indoor Climate and Energy (IDA ICE) and DesignBuilder Software [102, 103]. IDA Indoor Climate

and Energy (IDA ICE) is a dynamic simulation application that allows us to calculate the thermal indoor climate of individual zones and the energy consumption of the entire building. DesignBuilder Software is a simulation software used to check building energy consumption, CO2 emissions, and other building environmental aspects. In the research of [104], the integration of energy simulation as a mode of design assessment shows that information can be described in detail in a BIM to support design evaluation and decision-making in concept design. The BIM-based parametric design method models the building modeling and its configuration to be connected to real-world climatic parameters, and facilitates the study of building sustainability related to energy efficiency [105]. Most BIM simulation focuses on energy efficiency, climate (heating, ventilating, and air-conditioning), and different aspects concerning the building. We have surveyed the available systems, such as Bentley Architecture, Graphisoft ArchiCAD, VectorWorks Architect, Autodesk Revit Architecture, and Autodesk 3ds Max [7–11], to see whether they support our proposal. These systems allow us to use BIM applications and present a 3D visualization that improves productivity in building design and construction.

Although some of these BIM simulations are very advanced and focus on particular applications, they are not applicable for some aspects, such as building evacuation and emergency planning. Khan and Wainer in [24,25] presented a design and visualization of DEVS and Cell-DEVS simulation using Maya. Maya is used to visualize output simulation results in virtual time. The system is not interactive and cannot run in real time.

2.4 Emergency Simulation

Emergency simulation has received increasing attention in recent years and research has been developed and proposed for this purpose. Specifically, disaster management and evacuation strategies are the two most important subjects within this field. Due to the devastating occurrence and destructive impact of emergencies, it is impractical to perform real-world studies of this nature. M&S is an alternative to field-based experiments [12]. Generally, such simulations are large-scale programs, which in turn raise the need for efficient simulation engines. Modeling, simulation, and visualization techniques can help address many of the challenges in emergency response planning. A number of modeling and simulation applications for analyzing various disasters are surveyed in [12].

In [16], a mathematical model was implemented in an emergent algorithm that was presented to be used for movement in robotic collectives. A tree-in-motion mapping (TIMM) technique was presented in [17]; this allows for efficient environment mapping by a set of intelligent mobile-robot sensor agents that dominate limited communication bandwidth and computational power. Abielmona and Petriu in [18] provided a design and development for a multi-agent system that included agents that dominate a limited amount of communication bandwidth and computational power for a territorial security application. Castonguay and Wainer in [3] provided a design and development for an aircraft evacuation using DEVS simulation and visualization, which allowed the aircraft evacuation results to be visualized after the simulation.

A number of M&S applications exist for studying individual aspects of emergency response scenarios. However, a number of simulation tools have to be integrated to address multiple aspects of a single disaster event. Jain and McLean in [106] presented a framework for M&S in emergency response applications, which systematically integrates M&S tools to address the overall response. In [107], Jain and McLean

integrated gaming and simulation systems to train decision-makers and responders to work together as a team. A discrete-event environment introduced by Gonzalez in [108] presents the processes of analysis and design of a multi-agent system for a crisis response organization with the purpose of building a simulation testbed to experiment with different coordination mechanisms. To achieve efficient emergency management, we use virtual simulation with real-time emergency response, due to its capabilities to provide real-time system observations [13–15].

2.5 Agent-Based Distillation for Combat

Lanchester's Equations, presented by Lanchester in 1916 [109], were considered by defense analysts to model and hypothesize combat attrition (which is the act of weakening the enemy side by attack) [110, 111]. Lanchester's Equations are a set of linear dynamic equations that address attrition as a continuous function over time. Combat is modeled as a deterministic process that needs an attrition-rate coefficient.

Although Lanchester's Equations are easy to apply, such models, based on mathematical equations and physical description of combat, can explain an ideal model of military operations that is too abstract and does not satisfy reality. The drawbacks of Lanchester's Equations have been described and analyzed in [110, 112–114]. Research results in [110, 111, 114] show that warfare can be considered to be nonlinear behavior. Combat can be considered as a complex adaptive system (CAS) [111], and multi-agent system (MAS) platforms have been applied successfully for studying CAS. The combatants are modeled as agents, usually with a set of predefined characteristics. These agents adapt, evolve, and co-evolve with their environment [114, 115]. This view of combat allows researchers to use agent-based simulations on military operations. The field is usually known as agent-based distillation (ABD) or agent-based simulation (ABS). ABD emphasizes the concept of incorporating agents into

the environment [116]. Defense analysts have studied different behaviors of warfare based on ABD. Simulation is used to study and analyze the dynamics and behaviors of the system, which provide defense analysts with a useful tool for helping them to make decisions.

The agent's movement depends on five different weights: agent healthy friend, agent injured friend, agent healthy opponent, agent injured opponent, and the flag position [117]. The first four categories describe the agent's health state, which affect its movement in the battlefield space. The flag position is also used to calculate the nest movement, according to the distance between the agent and its opposite flag. The movement is calculated at each simulation time step for each agent. The agent can move to another cell or decide to stay in the same cell. Each cell in the space cannot be occupied by more than one agent at a time. The decision-making used by each agent to decide on the direction in which to move depends on the agent's personality in the movement algorithm. The movement algorithm of, for example, the Enhanced ISAAC Neural Simulation Toolkit (EINSTein) uses equation (2.5) to compute the penalty for the next location [110, 113]:

$$Z_{new} = \left(\frac{W_E}{E * R_S \sqrt{2}} \sum_{i=1}^{E} D_{i,new} \right) + W_F \left(\frac{D_{F,new}}{D_{F,old}} \right) \tag{2.5}$$

where

R_S Sensor range of agent about to move;

E Number of enemy entities within sensor range;

W_E Weighting toward enemy agents;

$D_{i,new}$ Distance to the ith enemy from the new location;

W_F Weighting toward the flag;

$D_{F,new}$ Distance to the flag from the new location;

$D_{F,old}$ Distance to the flag from the current location.

The movement algorithm of Map Aware Non-uniform Automata (MANA) (uses equation (2.6) to compute the penalty for the next location [114, 118]:

$$Z_{new} = \left(\frac{W_E}{100 * E}\right)\left(\sum_{i=1}^{E} \frac{D_{i,new} + (100 - D_{i,old})}{100}\right) + \left(\frac{W_F}{100}\right)\left(\frac{D_{F,new} + (100 - D_{F,old})}{100}\right) \tag{2.6}$$

where

E Number of enemy entities within sensor range;

W_E Weighting toward enemy agents;

$D_{i,new}$ Distance to the ith enemy from the new location;

$D_{i,old}$ Distance to the ith enemy from the current location

W_F Weighting toward the flag;

$D_{F,new}$Distance to the flag from the new location;

$D_{F,old}$ Distance to the flag from the current location.

Various development efforts for combat have been designed based on agent-based simulation. One of these is Irreducible Semi-Autonomous Adaptive Combat (ISAAC) [110, 111] and its extension, EINSTein [110, 113], designed by the US Marine Corps Combat Development Command. Although ISAAC and EINSTein have good analysis capabilities and support 2D visualization, they are not open source, meaning that the source code cannot be read or modified, and they also do not support 3D visualization. BactoWars was developed by Land Operations [119]. Although BactoWars is open source, so that the source code can be read and modified and is flexible with regard to adding any new, desired features to a model, it does not have good analysis capabilities and also does not support 3D visualization. MANA [114, 118] was provided by New Zealandś Defence Technology Agency. Although MANA has good analysis

capabilities and supports 2D visualization, it is not open source and also does not support 3D visualization. The Conceptual Research Oriented Combat Agent Distillation Implemented in the Littoral Environment (CROCADILE) [112] and the Warfare Intelligent System for Dynamic Optimization of Missions (WISDOM) [120–122] were developed at the University of New South Wales at the Australian Defence Force Academy. Although WISDOM has good analysis capabilities and supports 2D visualization, it is not open source and also does not support 3D visualization.

All of these development tools use a function for the agent's movement in space. Research introduced in [123] examined MANA's movement algorithms. The movement algorithm of agents within the EINSTein and MANA ABDs was modified by Grieger [123]. In such combat simulations, an agent always moves to the cell with the maximum weight. If a tie happens, the agent selects randomly between the cells in the tie. Due to this randomization, the stability of the solution may be affected and the outputs of this combat simulation are not guaranteed [117].

Chapter 3

The VCELL Framework

3.1 Introduction

3D visualization has become important in simulation, as it presents a 3D graphical interface that is effective when used for training simulations. 3D visualization also gives decision-makers an overview of the problem at hand before they begin working with the real system. However, for military tactical training simulation, developing or modifying different scenarios in an existing virtual environment at the code level requires significant programming time and validation efforts. As discussed in Chapter 1, 3D visualization simulation could be combined with different applications, for instance, BIM software, in order to increase productivity in building design and construction. 3D visualization could also be combined with emergency simulations and battlefield simulation to enhance training preparation, allowing decision-makers to investigate different scenarios. As discussed in Chapter 1, the central theme of this thesis is to develop a framework that allows different simulation models to receive real-time external parameters to be interactive, collaborative, and adaptive to the simulation events by integrating DEVS, Cell-DEVS, and 3D visualization with different modeling techniques, which improve simulation interoperability at the software level, which enhances simulation results. Our objective is to develop a generic framework to be

applied to simulation environments and applications. We use BIM simulation, emergency simulation, and land combat simulation as case studies. This thesis presents a simulation adaptation and interoperability methods using DEVS, Cell-DEVS, and 3D visualization.

As discussed in Section 2.2, DEVS is one of the most advanced general-purpose M&S frameworks. DEVS facilitates the reuse of tested models and improves the safety of simulations. As a result, this reduces development time. The Cell-DEVS approach combines DEVS and cellular models, which allow complex systems to be described as a cell space by using simple rules for modeling. Existing DEVS and Cell-DEVS tools usually include 2D visualization tools to facilitate the analysis of the output results on a 2D grid map, but 3D visualization for DEVS and Cell-DEVS in real time is still limited. Although surveys [3] show that 3D visualization can be applied to DEVS and Cell-DEVS, most existing methods are not interactive, collaborative, or adaptive. Many of them work with the outputs after the simulation ends (and not in real time). They are not collaborative, as they cannot exchange simulation data at runtime, and so they are not adaptive to the change or modification of simulation data during the simulation.

We also discussed the fact that 3D visualization methods (such as VR and computer games) have changed military training preparation of soldiers for successful results in wartime missions (as discussed in Section 2.1). However, for military tactical training simulation, developing or modifying different scenarios in an existing virtual environment requires significant programming time and validation efforts at the code level. Military tactical simulations are usually performed using land combat simulation, which was discussed in Section 2.5. The most popular method, agent-based distillations (ABDs), can be used to explore different aspects of land combat operations and help decision-makers to quickly investigate various scenarios in real battle conditions. The ABDs' movement algorithms are used to simulate the target's

movement in the battlefield by a large set of parameters related to the battlefield. However, under some circumstances, these algorithms use random movements, which may result in unpredicted simulation outputs. In addition, most existing ABDs include only 2D visualization facilities.

The analysis of the state-of-the-art also showed how simulation could be combined with 3D visualization for BIM in order to increase productivity in building design and construction (as discussed in Section 2.3). As BIM facilitates the coordination, cooperation, and maintenance of the construction life cycle, many research studies have included simulation. However, most of them focus only on thermal indoor climate, energy consumption, CO2 emissions, and other environmental building aspects. We discussed the fact that no current research has applied generic M&S to BIM, which permits different studies, including building evacuation and fire spreading. There are also no adaptive simulation systems that can interact in real time with building parameters changes.

Finally, we discussed the fact that current efforts in 3D visualization could be combined with emergency simulations to enhance training preparation, allowing decision-makers to investigate different scenarios (as discussed in Section 2.4). We also showed that M&S techniques have played an important role in emergency simulation where it is impractical to perform real-world studies. Generally, such simulations are large-scale programs, which in return raise the need for efficient simulation engines. Research studies described in Section 2.4 state that some of these emergency simulations have 3D visualization, but they do not have real input data from the field and they are not adaptive in real time.

Based on the above, this research aims to design and develop a framework that enhances and improves the existing simulation environments discussed previously in this thesis. The framework, called VCELL (visualization of DEVS and CELL-DEVS), can be used to solve various complex M&S problems with interactive and

collaborative 3D visualization for enhanced and improved results. The proposed framework should be adaptive to external data parameters, such as various hardware components, to achieve more accurate results. The framework should run in virtual and real time to allow for comparison between both output results, which facilitates the modification of the simulation rules and methods. The proposed framework should use predefined inputs, actual inputs, or a combination of both predefined and actual inputs of the simulated system for flexibility in different simulation scenarios. The framework must reduce the time spent in development, modification, and validation, to allow for the creation and modification of different scenarios for various simulation environments within a short period of time. These simulation environments should communicate with each other, and be interactive, collaborative, and adaptive. The VCELL framework will be explained in detail in the following sections.

3.2 VCELL Framework Definition and Features

The VCELL framework must allow different simulation models to receive real-time external parameters, to be interactive and adaptive to the simulation events by integrating DEVS, Cell-DEVS, and 3D visualization with different modeling techniques. VCELL is a simulation-driven architecture and a collaborative system for integrating Cell-DEVS simulation with DEVS simulation dealing with the event locations, which are spread out on the field space of the simulation and then visualized on a 3D visual scene.

We use CD++ (which was presented in Section 2.2) as a development tool for DEVS and Cell-DEVS simulations. As CD++ is open source, modifications can be made and new modules or facilities can be added. This also allows us to extend and integrate it with other toolkits. To achieve our goals, we modified CD++, adding two modules for sending and receiving the required data to and from other toolkits

(such as 3D visualization software) through a network. We also used C++, OpenGL, and other tools for 3D visualization simulation. We implemented and developed two modules for sending and receiving the required data to and from other toolkits (such as CD++ toolkit) through a network. This will be discussed in detail in Chapter 4, Chapter 5, and Chapter 6.

Using DEVS in VCELL helped us reuse tested models, improving the safety of the simulations, which resulted in reduced development time. In addition, using Cell-DEVS, which is based on DEVS, allows the user to solve problems by using simple rules for modeling the phenomena on the cell space. Moreover, using 3D visualization simulation provides us with a 3D graphical scene, which enhances and improves training, and helps decision-makers investigate different scenarios to obtain more accurate results. VCELL facilitates the integration of different hardware devices via the DEVS component. The hardware devices are updated on a grid space corresponding to the simulated space area and accordingly reach every location in the space, dealing with all the events in the Cell-DEVS component. A 3D model of different hardware devices is then included in the 3D scene environment to be visualized in the visual simulation component. We applied VCELL to the different simulation applications discussed earlier, BIM, emergency planning, and land combat, as case studies.

VCELL combines the Cell-DEVS definition for cellular models, DEVS-based real-time controllers, and VR. This component-oriented approach provides model reusability and interoperability, allowing any of the components to be integrated or replaced. The RT cellular simulation is an on-demand data source of the scenario, which is used by the DEVS model. Hardware-in-the-loop can be used to update the Cell-DEVS simulation data with real information on the simulation space area.

The key features of VCELL are summarized as follows:

- VCELL serves as a container to hold different software components without being specific to any implementation, which allows new components to be added

without having to make changes at the code level or the framework architecture.

- Using VCELL for adding various hardware devices allows hardware-in-the-loop, which enhances the simulation results by achieving more accurate results than when simulated hardware is used, and also improves training in different domains (such as in military training) by allowing the use of real equipment.
- VCELL can be used not only for real-time simulation, but also in virtual time, which allows enhancement of the simulation results by comparing both real-time and virtual-time outputs and hence modifying the simulation rules.
- VCELL can use predefined inputs, actual inputs of the simulated system, or a combination of both predefined and actual inputs, which provide flexibility for modifying the simulated system to get accurate results.
- VCELL reduces the time taken in scenario development, modification, and validation by using the simple rules in Cell-DEVS simulation.

We first extended CD++ with a new component to hide CD++ internal implementation. This component consists of functions to support synchronization and adaptation and communicates using network messages. An interface was developed between DEVS, Cell-DEVS simulations, and 3D visualization, so that various instances of each domain can cooperate with each other during the same simulation session. This means that simulation can be manipulated in real time.

VCELL is generic and can be used in different application domains. As a proof of concept, the framework was applied successfully to BIM, emergency simulation, and land combat simulation, as discussed previously. First, we applied VCELL and integrated it with BIM models (this means that different simulations can be performed on BIM models to improve and enhance the BIM simulation results). VCELL is adaptive and collaborative, as we can obtain the actual building parameters and perform simulations on them, which results in improvement and enhancement of the simulation of BIM. We can also visualize the output simulation on the 3D visualization

sub-system. This will be explained in detail in Chapter 4.

Table 3.1 compares VCELL with existing BIM toolkits (note that there are many existing BIM toolkits, which makes it difficult to list all of them; this table summarizes the most advanced BIM systems developed recently). The results shown are on a scale of P (poor), G (good), and E (excellent). *Open source* means that the source code can be read and modified. *Flexibility* is in terms of the ability to add any new features desired in a model. *Adaptation* considers a structure modified to fit a changed environment. *Analysis* means that statistical packages are included. *Visualization* means that there is visualization support.

When VCELL was applied to an emergency simulation, we implemented an integrated emergency management system based on the DEVS sub-system to develop new classes of cellular models for emergency response applications. The Cell-DEVS sub-system allows models to receive external information, and the simulation parameters can be updated at any time, due to the continuous-time nature of the discrete-event specifications. The emergency simulation is based on the Cell-DEVS sub-system, and the emergency management is based on the DEVS sub-system, which uses a robotic agent as an example to respond to the emergency simulation in real time. We also use the 3D visualization sub-system, which takes the results of the emergency simulation and the emergency management to be visualized in 3D real-time visualization. This will be explained in detail in Chapter 5.

Table 3.2 shows a comparison between VCELL and selected emergency simulation models (here it is also difficult to list all the emergency simulation models that are published; however, the list on the table focuses on some of the most recent emergency simulation work [3, 13, 15, 17, 18].

Finally, when we applied VCELL to land combat simulation, we incorporated two sub-systems: a Cell-DEVS agent simulation, and a 3D visualization agent, which is

Table 3.1: Comparison of BIM Simulation Toolkits with VCELL

Model	Open Source	Flexibility		Adaptation		Analysis		Visualization	
Bentley Architecture	No	G	Specific algorithms	G	Evolution of BIM attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization
Graphisoft ArchiCAD	No	G	Specific algorithms	G	Evolution of BIM attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization
VectorWorks Architect	No	G	Specific algorithms	G	Evolution of BIM attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization
Revit Architecture	No	G	Specific algorithms	G	Evolution of BIM attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization
VCELL	Yes	E	User specified algorithms	G	Evolution of BIM attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization

Table 3.2: Comparison of Emergency Simulation Models with VCELL

Model	**Open Source**	**Flexibility**		**Adaptation**		**Analysis**		**Visualization**	
[13]	No	P	Few algorithms	P	Merely reactive	P	N/A	G	3D visualization
[15]	No	P	Few algorithms	P	Merely reactive	P	N/A	G	3D visualization
[17, 18]	No	P	Few algorithms	G	Evolution of agents attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization
[3]	Yes	P	Few algorithms	P	Merely reactive	P	N/A	G	3D visualization
VCELL	Yes	E	User specified algorithms	G	Evolution of agent attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	2D&3D visualization

based on a 3D visualization sub-system. The Visual Cell-DEVS agent for land combat is a new way of solving the randomization movement problem of movement algorithms in agent-based simulation by using the 3D visualization agent, which obtains the real position of agents in combat. The Cell-DEVS agent simulation reduces the time taken to create or modify tactical scenarios in the 3D visualization simulation by using the Cell-DEVS formalism and CD++, which includes an interpreter to write simple rules. The results can be viewed in 3D visualization by using 3D scenario generation. These components will be explained in detail in Chapter 6.

We compared VCELL when applied to land combat simulation with selected existing ABDs, as shown in Table 3.3 (in this case, there are also plenty of ABDs toolkits available, which makes it difficult to list all of them. However, the list includes the most recent ABDs). It is evident that in this case, *speed* of execution is important. When analyzing the tables, we see that VCELL is open source and more flexible than the other ABDs. Its performance onscreen is good. In addition, VCELL is more adaptive than other ABDs and includes good analysis capabilities. It also supports 3D visualization, which is not applicable in other ABDs.

3.3 The Framework Architecture

The following chapters in this thesis will discuss the design, implementation, and performance analysis of the VCELL framework in detail. In this section, we present the overall software architecture of the framework.

Fig. 3.1 shows the overall architecture of VCELL, which integrates DEVS simulation, Cell-DEVS simulation, and 3D visualization outputs. VCELL is composed of three collaborative sub-systems: the cellular model implemented in Cell-DEVS, the hardware device interfaces implemented in DEVS, and the visualization component

Table 3.3: Comparison of ABDs with VCELL

Model	Open Source	Flexibility		Adaptation		Analysis		Speed		Visual	
BactoWars	Yes	E	User specified algorithms	P	Merely reactive	P	Computes averages of some variables	G	Runs with occasional pauses	G	2D
EINSTein	No	P	Few algorithms	P	Merely reactive	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	Runs continuously	G	2D
MANA	No	P	Few algorithms	P	Merely reactive	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	G	Runs with occasional pauses	G	2D
WISDOM	No	P	Few algorithms	G	Evolution of agent attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	P	Noticeable time for screen refreshes	G	2D
VCELL	Yes	E	User specified algorithms	G	Evolution of agent attributes	G	Supports basic statistical methods (e.g. mean, standard deviation, etc.)	E	Runs continuously	E	2D & 3D

that renders the 3D scenes. Each of these sub-systems runs on a different computer, communicating through messages sent over a network. The framework allows different users in different interfaces to collaborate and interact with each other in real-time. Each sub-system is independent in its engine and function and update each other through messaging via a network infrastructure. Each sub-system is composed of two main components: a *Sender* that transmits the information and simulation updates from one sub-system to the other two sub-systems, and a *Receiver* that collects information and the simulation updates of the other two sub-systems in real time. The Cell-DEVS sub-system communicates with the DEVS sub-system, informing it about the dimensions of the cell space area, and sends updates about the location of the simulated space on the grid. At the same time, it also sends this information to the visualization sub-system, providing it with the required real-time data about the scene to generate the 3D scene environments. The DEVS model uses the information received from the Cell-DEVS engine to perform the requested simulation. Based on these commands, the hardware devices respond on the simulated grid area and update the grid locations within the cell space area one after another when required. The DEVS sub-system dynamically updates the cell-DEVS sub-system and the visualization sub-systems about the grid locations that have been simulated. This process continues until all grid locations have been simulated.

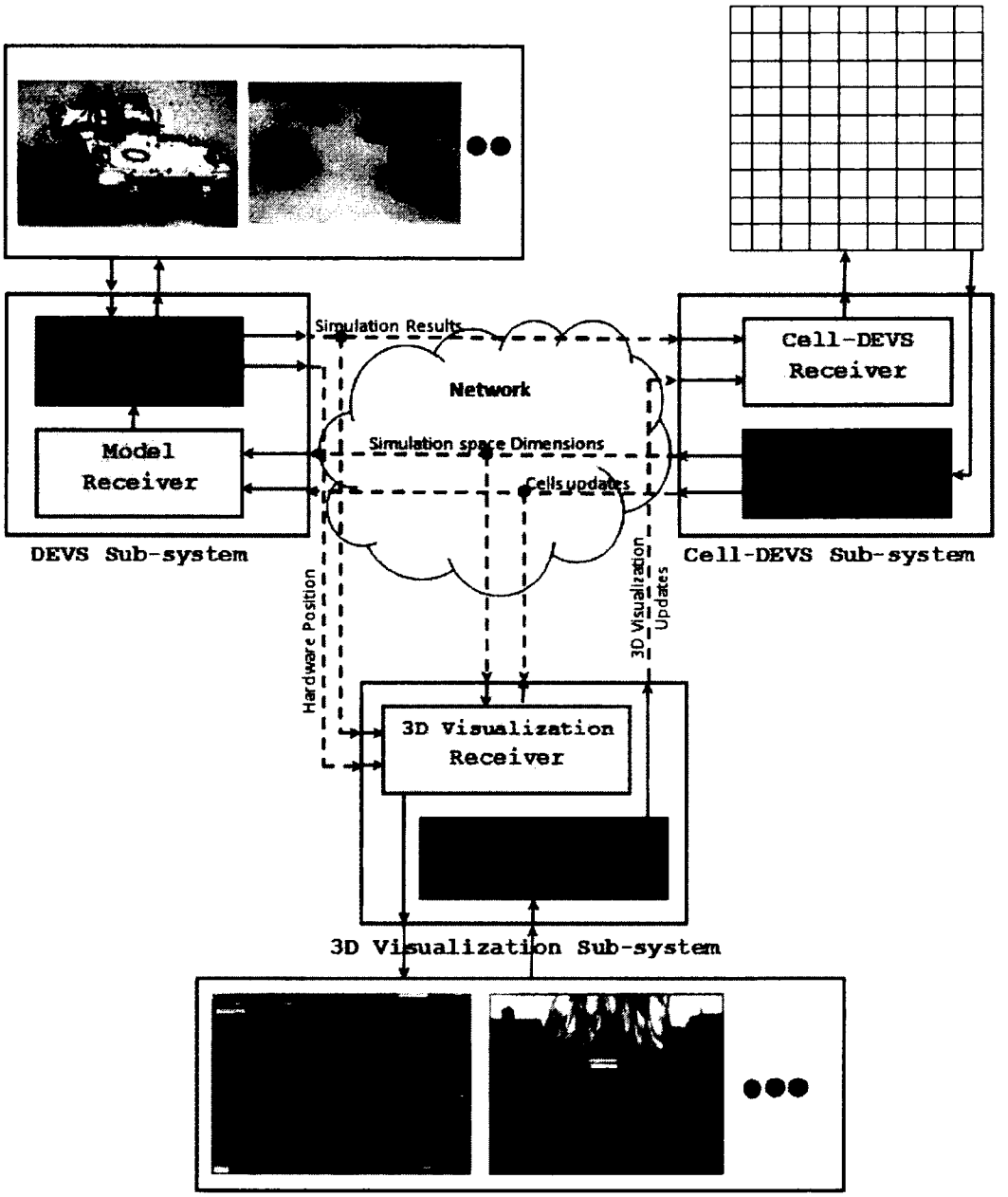


Figure 3.1: The Visual Cellular Discrete Event System Specification (VCELL) Framework Architecture

The 3D visualization sub-system shown in Fig. 3.1 consists of two main components: the *Receiver*, which takes the data from the DEVS and the Cell-DEVS sub-systems, and the *Sender*, which is responsible for the calibration and casting of data to be transmitted to the DEVS and the Cell-DEVS sub-systems. On the Receiver component, there is a separate thread that is spawned for receiving the required simulation data. The visualization sub-system produces 3D scenes from the updates received dynamically from both the DEVS simulation and the Cell-DEVS sub-systems. The visualization sub-system also sends 3D models dimension updates and the required simulation information to the Cell-DEVS sub-system. It is possible to create reusable library of the visualization components. It is also possible to add or extend the visualization components without changing in the core of the framework. The visualization sub-system can be a 3D visualization engine, VR software, serious game engine, or BIM software, such as Autodesk Revit Architecture. A more detailed overview of the system will be discussed later in Chapter 4, Chapter 5, and Chapter 6.

The collaboration of the sub-systems is based on a global message structure transferred over a network infrastructure. Each sub-system has two separate threads: one for sending simulation data, and the other for receiving requested simulation data. The message structure contains an integer data type to decode the type of the message according to the sending/receiving sub-system, the dimension message, the cell-space update message, the visualization update message, the next movement message, and the extinguish message, which will be explained in Chapter 5 and Chapter 6.

3.4 VCELL Framework Implementation

The VCELL framework has been implemented using various software components, libraries, and packages, details of which will be discussed in the following chapters.

For the DEVS and Cell-DEVS sub-systems, we used the CD++ toolkit for modeling and simulation. We developed and implemented the two main components of DEVS and Cell-DEVS sub-systems, and added the two main components to CD++. For the 3D visualization sub-system, we used various software tools for each case study of the simulation. As discussed earlier, we applied VCELL to BIM, emergency and disaster, and land combat simulations. For the BIM models, we used the Autodesk Revit Architecture and Autodesk 3d Max toolkits. These tools are provided with a scripting language and APIs that were used to develop and implement the two main components in the 3D visualization sub-system. The implementation of this component will be explained in more detail in Chapter 4. For the emergency and land combat simulation, the visualization sub-system is developed and implemented in Visual C++, OpenGL, and Vega Prime toolkits. The development and implementation of these will be explained in more detail in Chapter 5 and Chapter 6.

The remaining chapters of the thesis are organized as follows:

- In ***Chapter 4***, we show the use of VCELL (Cell-DEVS and 3D visualization) in virtual time for BIM. VCELL is applied and integrated with BIM models. This means that different simulations can be performed on BIM models to improve and enhance the BIM simulation results. We propose designing a Cell-DEVS/BIM integration and describe a prototype implementation in the form of a BIM add-in for Cell-DEVS simulation, and then visualize the output simulation of Cell-DEVS on the BIM model.
- In ***Chapter 5***, we use the VCELL framework in real time for emergency simulation. We introduce a simulation-driven architecture for integrating emergency simulation with robotic first responders moving towards emergency locations, which are spread out on the field. The robot is placed on a grid corresponding to the simulated emergency area, and reaches every location, dealing with the emergency.

- In ***Chapter 6***, we use the VCELL framework (the Cell-DEVS and 3D visualization components only) in real time for land combat warfare. We show how VCELL can provide a solution to the randomization problem caused by ABD toolkits. We present collaboration between an agent based on Cell-DEVS formalism and a visual agent simulation based on a 3D real-time visualization simulation in real time. We also show how to reduce programming time to develop or modify the scenario tactics for combat in real-time visual simulation using Cell-DEVS.

Chapter 4

VCELL for Building Information Modeling

4.1 Introduction

As discussed in Chapter 3, we used the VCELL framework in the context of BIM simulations. In this chapter, we focus on showing how VCELL provides a general framework that can be used for BIM applications.

In order to solve different design and simulation problems in the field of building information modeling, it is important to be able to easily incorporate new models and simulations. To do so, VCELL provides a reconfigurable Interactive Environment System (IES) to support the simulation of BIM. Cell-DEVS models receive information from BIM, and the output results of Cell-DEVS simulation can be used for visualization on the BIM model. We show a Cell-DEVS/BIM integration and describe a prototype implementation in the form of a BIM add-in for the Cell-DEVS simulation, and then we will visualize the output simulation of Cell-DEVS on the BIM model.

As discussed in Chapter 3, the central theme of this thesis is to develop a framework that allows different simulation models to receive real-time external parameters,

in order to be interactive, collaborative, and adaptive to simulation events, by integrating DEVS, Cell-DEVS, and 3D visualization with different modeling techniques, which improve simulation interoperability at the software level, which in turn enhances simulation results. As previously mentioned, we first selected the Cell-DEVS, DEVS framework to design a new VCELL framework, which mainly aims to interoperate independently developed and adaptive simulation systems. The design methodologies presented in this chapter show how these were adapted for BIM, which mainly demonstrate building information as parameters where the simulation information flows from those models.

We used CD++ [77] to simulate Cell-DEVS models, and the Autodesk Revit architecture and Autodesk 3ds Max toolkits for BIM [10,11]. CD++ obtains the initial simulation values from an external value file, and runs the simulation according to the rules defined by the model. On the other hand, this interface exposes the internal CD++ implementation. This means that the BIM input/output parameters are tied to the CD++ model file and value file. Thus, using the CD++ specific interface to interoperate with other systems is not practical, as it would require implementation changes in those systems. To bring those interfaces together and ease interoperability at the software level, this research targeted the two major syntactic and structural elements of the BIM and Cell-DEVS integration. We extended the Cell-DEVS architecture with real BIM parameters, and described the synchronization messages to be transmitted through a network. This interface simplified the synchronization between different systems and improved their performance. In this VCELL framework, CD++ still uses its original component to interoperate various CD++ instances, while using the new designed component to interoperate with BIM.

This system is applied and integrated with BIM models (this means that different simulations can be performed on BIM models to improve and enhance the BIM simulation results). We design a Cell-DEVS/BIM integration and describe a prototype

implementation in the form of a BIM add-in for Cell-DEVS simulation, and then visualize the output simulation of Cell-DEVS on the BIM model.

4.2 BIM and Cell-DEVS System Architecture

The system is composed of two collaborative sub-systems: the cellular model implemented in Cell-DEVS, and the BIM model. Each of these sub-systems runs on a different computer, communicating through messages sent over a network, updating each other through messaging via a network infrastructure. The Cell-DEVS model communicates with the BIM sub-system, informing it about the dimensions of the cell space area. The Cell-DEVS model uses the BIM information received from the BIM model to perform the required simulation. Fig. 4.1 and Fig. 4.2 present a more detailed overview of the VCELL framework of the IES for BIM models. The Interactive Environment System for Revit (*IES_Revit*), shown in Fig. 4.1, integrates the Cell-DEVS simulation and the BIM model to simulate the data received from the BIM model on Cell-DEVS. Then the *IES_Max*, shown in Fig. 4.2, integrates the Cell-DEVS simulation and the BIM model to visualize the output simulation results sent from Cell-DEVS to BIM on its interface. Based on the above, we see that VCELL can be adapted and used in a collaborative manner when utilized in BIM modeling: collaborative because we can exchange data between different systems, and adaptive because we can obtain the actual building parameters and then run the simulation on the actual requested parameters to get the simulation output results, which can be changed according to the change in the actual building parameters. This results in improvement and enhancement of the BIM simulation.

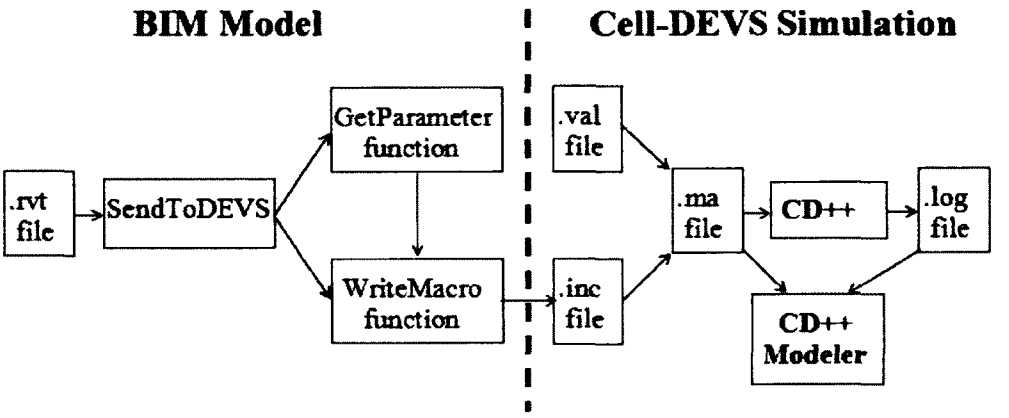


Figure 4.1: Interactive Environment System for Revit (*IES_Revit*) Architecture

Fig. 4.1 illustrates the architecture of the *IES_Revit*. The *IES_Revit* consists of two main phases:

- Receiving the required data to be simulated from the BIM model
- Simulating the data received from the BIM model using Cell-DEVS.

In the first phase, we developed an *IES_Revit* model to get different data parameters that will be simulated. The *IES_Revit* model then transfers these data as values to be sent to Cell-DEVS. The *IES_Revit* model is developed using Autodesk Revit Architecture [10] as implementation software. This part of the *IES_Revit* was written in Visual C#, which provides a graphical user interface invoked from Revit.

In the second phase, we focused on simulating the received BIM model data using Cell-DEVS. To do this, we defined a Cell-DEVS model that contained the cell space definition: dimensions, initial values, data received from the BIM model, and the rules that will be applied to the BIM model. CD++ allows for these models to be implemented, and provides 2D and 3D visualization using VRML and Java. 2D and 3D visualization enables visualization of Cell-DEVS models so that the output of our simulation model will be shown as a grid

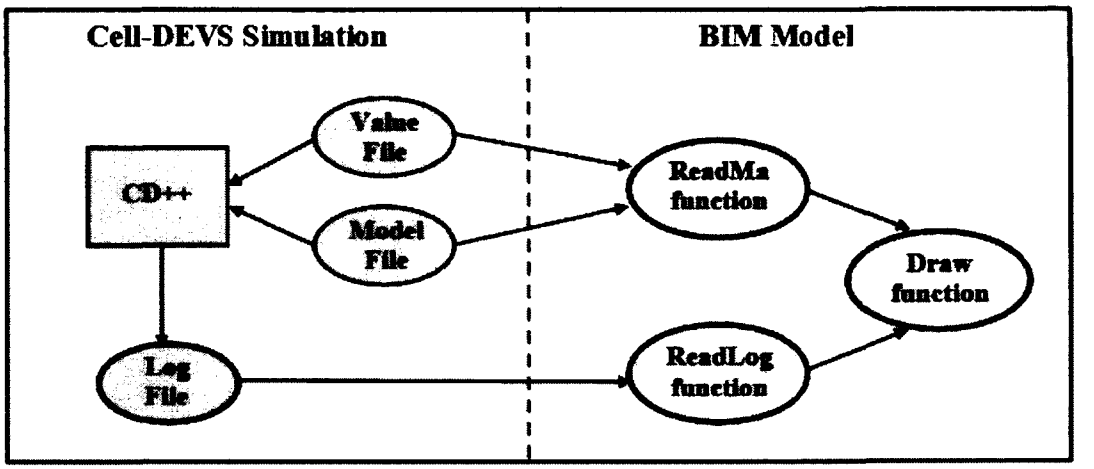


Figure 4.2: Interactive Environment System Max (*IES_Max*) Architecture

Fig. 4.2 illustrates the architecture of the *IES_Max*, which consists of two main phases:

- Receiving the simulated data results from Cell-DEVS to be visualized on the BIM model
- Visualizing the data received from Cell-DEVS on the BIM model.

The first phase of *IES_Max* obtains the simulation data results to be visualized. These data are then transferred as values to the BIM model. The *IES_Max* model is developed using Autodesk 3d Max as the implementation software, and Max script. We first read the Cell-DEVS model file (.MA), which contains the definition of the behavior of the Cell-DEVS models and the initial data, which can also be contained in an external value file (.val). We then read the initial status of the Cell-DEVS and the simulation log (.log file), which contains all the steps of the output simulation results and their virtual time.

In the second phase, we visualize the data received from Cell-DEVS on the BIM model using Autodesk 3d Max. To do this, we define a model that reads the cell space definition: dimensions, initial values, and data received from the Cell-DEVS

model. Autodesk 3d Max allows these models to be implemented, and provides a 3D visualization scene that enables the visualization of Cell-DEVS simulation results on the BIM model. Based on the simulation status during the simulation lifetime and the final simulation results, we draw the collected simulation data in a 3d Max visualization scene.

In the following sub-sections, we show a detailed version of each of the simulation steps based on these ideas.

4.2.1 Integrating Cell-DEVS and BIM

As discussed in the previous section, Cell-DEVS simulation is applied to BIM (using Autodesk Revit Architecture) to improve the output results, enhance the BIM models, and improve the Cell-DEVS simulation results by applying the simulation on actual parameters that are requested from the BIM model. Revit Architecture is a Parametric BIM tool, in which 3D models and 2D drawings can be built. We can develop different tasks using the Revit API. The Revit API allows us to create and delete different model elements, such as floors and walls. We also use the Revit API to obtain different model parameter data and model graphical data. The Revit Platform API applications can be developed using Visual C# or VB.NET. Both Visual C# and VB.NET allow for the writing of equivalent code and compile to the same intermediate language (IL) code, which implies that one has no performance advantage over the other. We decided to use Visual C# for practical reasons.

The first phase of *IES_Revit* is to obtain different parameters to be simulated from the BIM model, and to transfer these data to the Cell-DEVS models. We developed a prototype implementation in the form of a BIM add-in for Cell-DEVS simulation. This can now be invoked as the add-in tab for the AutoDesk Revit Architecture, as shown in Fig. 4.3. The add-on is responsible for executing the program to obtain the

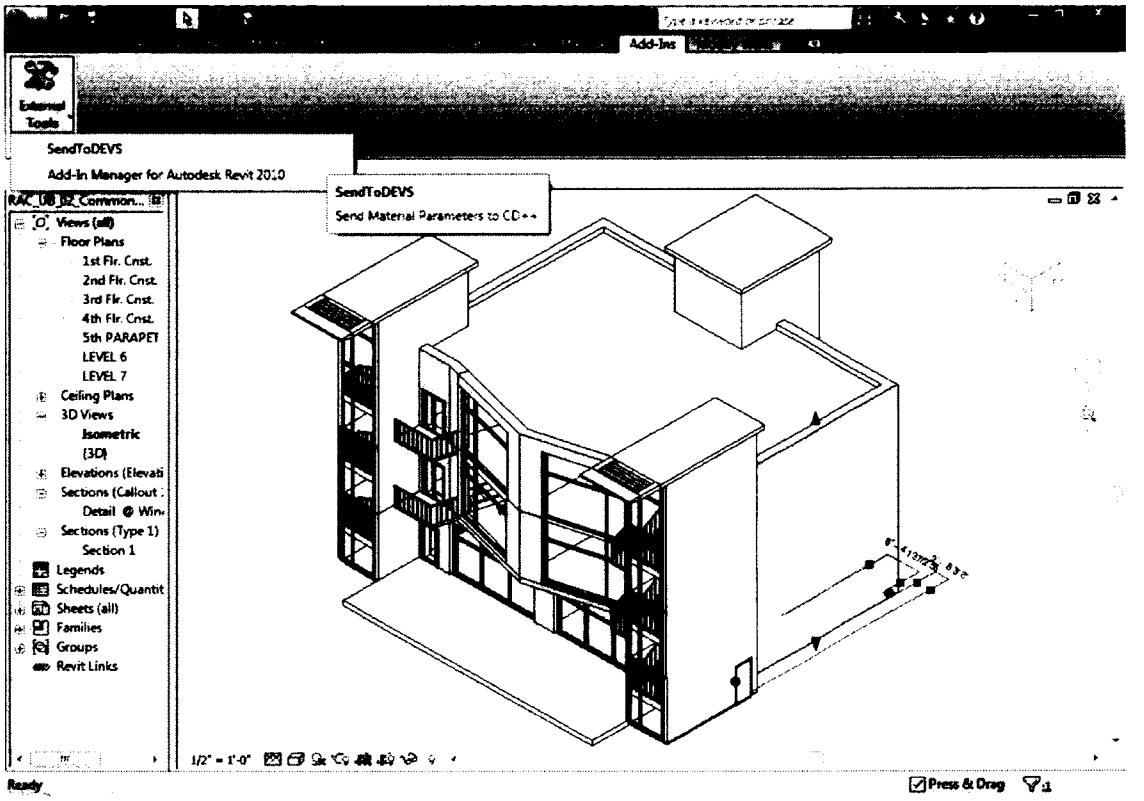


Figure 4.3: Interactive Environment System for Revit (*IES_Revit*) Interface

required parameters and send them to Cell-DEVS to be simulated.

The *IES_Revit* interface receives the required parameters of the chosen item (e.g., a wall) in the active Revit document. This function receives the parameters of the selected element in the active document of the Autodesk Revit architecture. We then use this information to define a CD++ macro containing the parametric information. The new macro now contains the new value, which will be simulated in Cell-DEVS. Different elements in the same active Revit document will transmit different parameters in the newly selected element.

4.2.2 Simulation with BIM Data

The second phase of *IES_Revit* is responsible for simulating the received BIM model data based on the rules that are written in the model file. The model file contains the cell space dimensions, the initial values, the definition of the macro file that contains the values received from the BIM model, and the rules that will be applied to the BIM model.

In this section, we show a sample Cell-DEVS model representing a diffusion limited aggregation (DLA) phenomenon [124]. DLA occurs when diffusing particles stick to and progressively enlarge an initial seed, represented by a fixed object. The seed typically grows in an irregular shape resembling frost on a window [81], or humidity and mold on a wall. The DLA model was defined using CD++, which includes an interpreter for a specification language that describes Cell-DEVS models. A set of rules is used to define the model; each rule indicates the output value for the cell's state after satisfying the precondition in this rule. These rules are performed sequentially until one rule produces the solution. We used CD++ macro definition facilities to read the parameter values received from the BIM model, and defined a Revit macro for the Cell-DEVS model. The output simulation can be seen using 2D visualization facilities provided by the CD++ modeler tool.

The DLA model uses two types of particles: fixed particles (seeds) and mobile particles. There can be one or more seeds in each DLA Cell-DEVS model. A cell with a seed is fixed, and it has a value equal to 5. There is a mobile particle percentage of the cells in each DLA Cell-DEVS model. A mobile particle can move according to its value, in one of four directions: up (1), right (2), down (3), and left (4). We set an initial value from 1 to 4 randomly to occupy the cells in a certain concentration. This concentration is calculated and obtained in the Revit macro from the BIM model. Below is a description of some of the rules used:

```
% initialize the cells with mobile particles
% in the range with value of concentration
rule : {round(uniform(1,4))} 100 {(0,0) = -1
        and random < \textbf{#macro(Revit)}}
```

The following rule presents that fixed particles remains fixed:

```
% fixed particles remains to be fixed
rule : 5 100 { (0,0)=5 }
```

The following rules present the moving of mobile particles:

- A cell has a mobile particle with value equal one can move to the above empty cell if there is no other mobile particle trying to move in to this empty cell.

```
% direction = 1 (up)
% stay and change direction when nowhere to move
rule : {round(uniform(1,4))} 100 {(0,0)=1
    and (-1,0)!=0}
rule : {round(uniform(1,4))} 100 {(0,0)=1
    and (-1,0)=0 and (((-2,0)=3 and(-2,-1)!=5
    and(-3,0) !=5 and(-2,1)!=5) or ((-1,-1)=2
    and(-1,-2)!=5 and(-2,-1)!=5 and(0,-1)!=5)
    or ((-1,1)=4 and(-2,1) !=5 and(-1,2) !=5
    and(0,1)!=5))}
% move otherwise
rule : 0 100 {(0,0)=1 and (-1,0)=0 and t}

% direction = 2 (right)
% stay and change direction when nowhere to move
rule : {round(uniform(1,4))} 100 {(0,0)=2
    and (0,1)!=0}
rule : {round(uniform(1,4))} 100 {(0,0)=2
    and (0,1)=0 and (((0,2)=4 and (-1,2)!=5
    and (0,3)!=5 and (1,2)!=5) or((-1,1)=3
    and (-1,0)!=5 and(-2,1)!=5 and (-1,2)!=5))}
```

```
% move otherwise
rule : 0 100 {(0,0)=2 and (0,1)=0 and t}
```

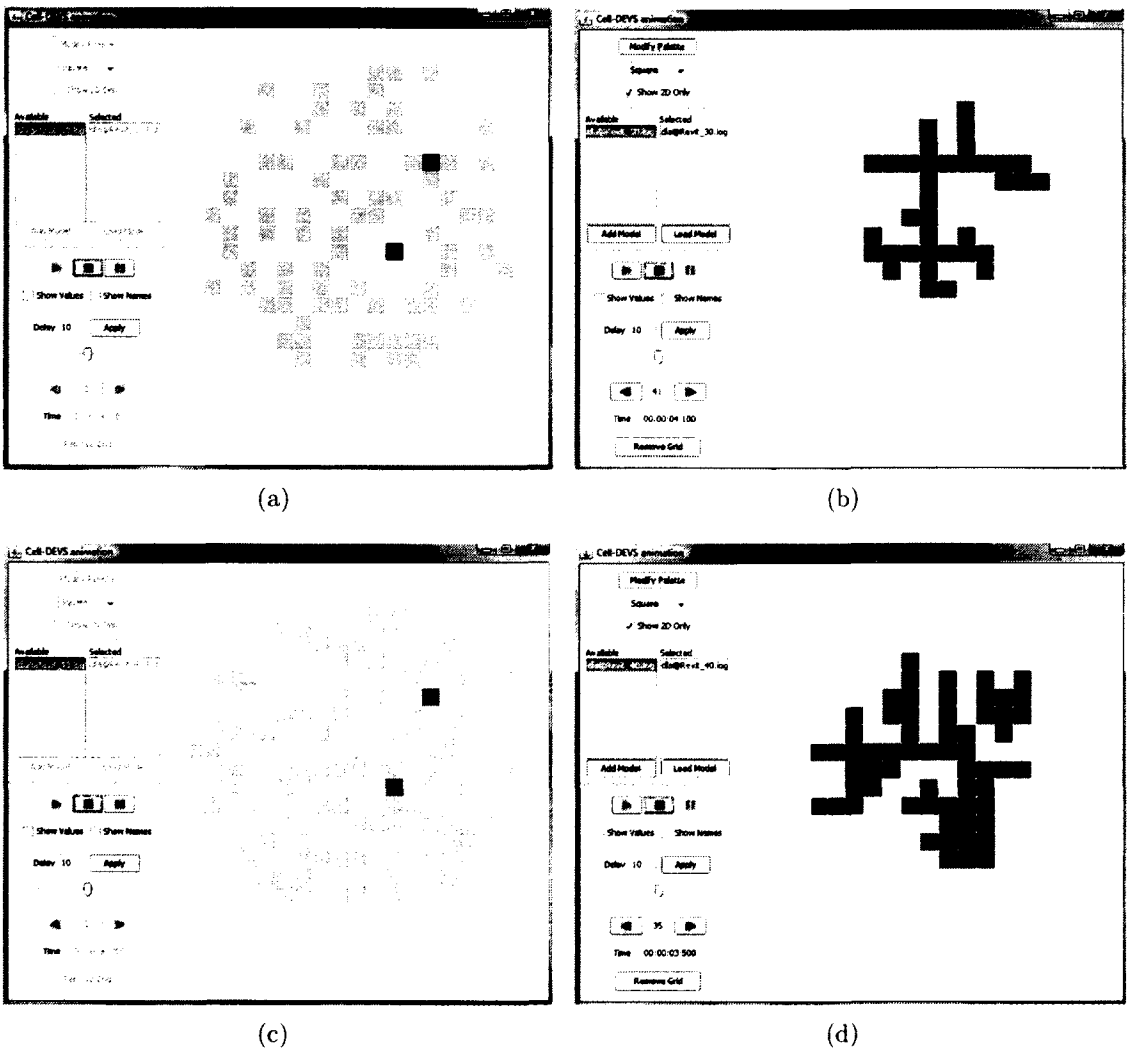
- A cell has a mobile particle with value equal three can move to the down empty cell if there is no other mobile particle trying to move in to this empty cell.
- A cell has a mobile particle with value equal four can move to the left empty cell if there is no other mobile particle trying to move in to this empty cell.
- A cell has a mobile particle becomes fixed if there is an adjacent fixed particle cell.

```
% the particle becomes fixed if an adjacent cell
% contains fixed particle
rule : 5 100 { (0,0) >0 and (0,0)<5 and
    ((-1,0)=5 or (0,-1)=5 or (0,1)=5 or (1,0)=5)}
```

Based on the above rules, a cell that has a mobile particle with a value equal to 1 can move to the empty cell above; with a value equal to 2, it can move to the empty cell to the right; with a value equal to 3, it can move to the empty cell below; or with a value equal to 4, it can move to the left empty cell if there is no other mobile particle trying to move into this empty cell. Finally, a cell with a mobile particle becomes fixed if there is an adjacent fixed particle cell.

We assume a DLA Cell-DEVS model with two initial seeds. The concentration percentage of mobile particles will vary due to the material parameter type value received from BIM. We ran the simulation for two different materials for the specified two seeds in the DLA Cell-DEVS model: one for concrete and the other for brick. We assume that the concentration percentage will be 30% for the concrete material and 40% for the brick material. The simulation output of each run for the concrete and the brick is shown in Fig. 4.4. We see that both initial figures have two initial seeds (dark cells). As the simulation starts, some of the mobile particles that are adjacent to the fixed initial seeds and satisfy the rules stated above will become fixed particles.

During the simulation, the rules are checked for each step until the simulation ends. We observe that the deformation of the DLA on the brick surface is bigger than the deformation of the DLA on the concrete surface, as the deformation of the DLA is proportional to the concentration, and the concentration of the brick surface is greater than that of the concrete surface.



(a) (b)

(c) (d)

Figure 4.4: (a)Initial State for Concrete; (b)Final State for Concrete; (c)Initial State for Brick; (d)Final State for Brick

Based on the above example, we see that VCELL is more flexible than the existing BIM toolkits, as we can simulate any model (like DLA) based on the BIM model, while existing BIM toolkits do not support this feature. VCELL allows for new features to be added to the BIM model to easily simulate various problems, as discussed in Chapter 3.

4.2.3 Cell-DEVS & BIM 3D Visualization

This sub-section discusses the details of the design and implementation of the visualization sub-system for the output results of the Cell-DEVS simulation on the BIM model. The visualization sub-system integrates the Cell-DEVS simulation and the BIM model to visualize the output result of the cell-DEVS simulation on the BIM model. This will facilitate the improvement of the BIM model, as we can visualize the output results directly on it, which is more effective for the 2D visualization of Cell-DEVS. In this sub-section, we show that VCELL is not specific to any implementation, as we need the output results. This implies that new components can be added without changes having to be made at the code level or in the framework architecture, as discussed in Chapter 3.

The methodology of the visualization sub-model is based on integration between the Cell-DEVS simulation and the BIM model. In the Cell-DEVS simulation, we retrieve the output simulation result of the Cell-DEVS simulation that was applied to the BIM model and then visualize these output data on the BIM model visual environment. The Cell-DEVS simulation uses its model file, which contains the definition of the behavior of the Cell-DEVS models and the initial data or the path of the value file, and the parameters received from the BIM model to run the simulation. These output simulation data will be saved in a log file. On the other hand, the BIM model has its visual environment, which is used to visualize the output simulation of the Cell-DEVS. We read and obtained the output simulation data and presented it

by developing two functions on the BIM model: one to obtain the initial data and the cell dimension of the model file, and the other to retrieve the output result data from the Cell-DEVS log file. We also developed a GUI that facilitates loading the value file, the model file, and the log file and then displaying the Cell-DEVS simulation output on BIM model. This GUI will be described as a prototype implementation in the form of a BIM add-in for Cell-DEVS simulation. We will build the graphical display output using 3D visualization tools.

This model allows us to obtain a 3D visualization environment of the simulated output results of Cell-DEVS on the same BIM model. This will improve the decisions taken by all members who are involved in the construction project cycle. Therefore, the visualization sub-model will improve and enhance the simulations applied to the BIM model.

The *IES_Max* sub-system integrates the Cell-DEVS simulation and the BIM model to visualize the output result of the Cell-DEVS simulation on the BIM model. This will facilitate the improvement of the BIM model, as we can visualize the output results directly on it, which is more effective for the 2D visualization of Cell-DEVS. *IES_Max* is implemented using Autodesk 3ds Max, used because it supports BIM and has a great 3D environment scene.

We built a graphical display output using 3D visualization tools. We decided to expand our visual environment using Autodesk 3ds Max. Autodesk 3ds Max is a powerful application for 3D modeling and animation, using special effects and rendering. 3ds Max allows users to create 3D animation and visual effects. More functions can be added to Autodesk 3ds Max using MAXScript, which is a built-in script language that facilitates the creation of functions and tools to efficiently enhance 3ds Max. We used the 3ds Max modeling and animation toolkit to create 3D visual environments for the Cell-DEVS simulation of DLA as an example. *IES_Max* is an application written in MAXScript that provides a GUI allowing CD++ files (*.ma

and *.log files) to interact with 3ds Max, and allows the corresponding Cell-DEVS simulation to be visualized in a 3D visual environment of the BIM model. This BIM model, which is exported as an FBX file (type of Autodesk file formats) from the Autodesk Revit Architecture, is imported into 3ds Max. *IES_Max* then animates the 3D visual scene file in accordance with the CD++ files. *IES_Max* allows us to create a 3D visualization from the CD++ files created by the CD++ toolkit. 3ds Max has implicit support for hardware accelerated rendering. The 3ds Max visualization tool provides basic services that enable simple visualizations, including design and implementation of a GUI based on the MAXScript within the 3ds Max toolkit.

We used the MAXScript language-i.e., the 3ds Max Toolkit script-to write the program to initialize the GUI interface window for the 3D visualization. We read the output simulation data of the CD++ file, and then displayed the 3D visual outputs. We obtained the Cell-DEVS model to read the *.ma file, which contains the dimensions of the simulation model and the value file name, then reformatted it to be used in the required argument to obtain the initial values of each object from the simulation model and reformatted them to be used in the required argument to draw the visual outputs. We read the log file, which contains the time and position of each object from the simulation model, and reformatted it for display. We created objects and displayed the 3D visualization of the CD++ simulation model in the display window of 3ds Max, based on the dimension of the simulation model, which controls the size of the drawing area and the position of each cell to be drawn in the specified location.

IES_Max consists of a GUI; as shown in Fig. 4.5, this is the graphical interface that requests the user to select a particular file. In the display window in this figure, we can see the DLA model (inside the circle) on the brick surface. As we can see, the results of the DLA are visualized in the building model, which provides an interactive visualization of the output simulation results of the Cell-DEVS simulation. VCELL

allows this to be done, and it is not specific to any implementation, as we simply read the output simulation results from the Cell-DEVS. This means that we can add new components without having to make changes at the code level or in the framework architecture, as discussed in Chapter 3.

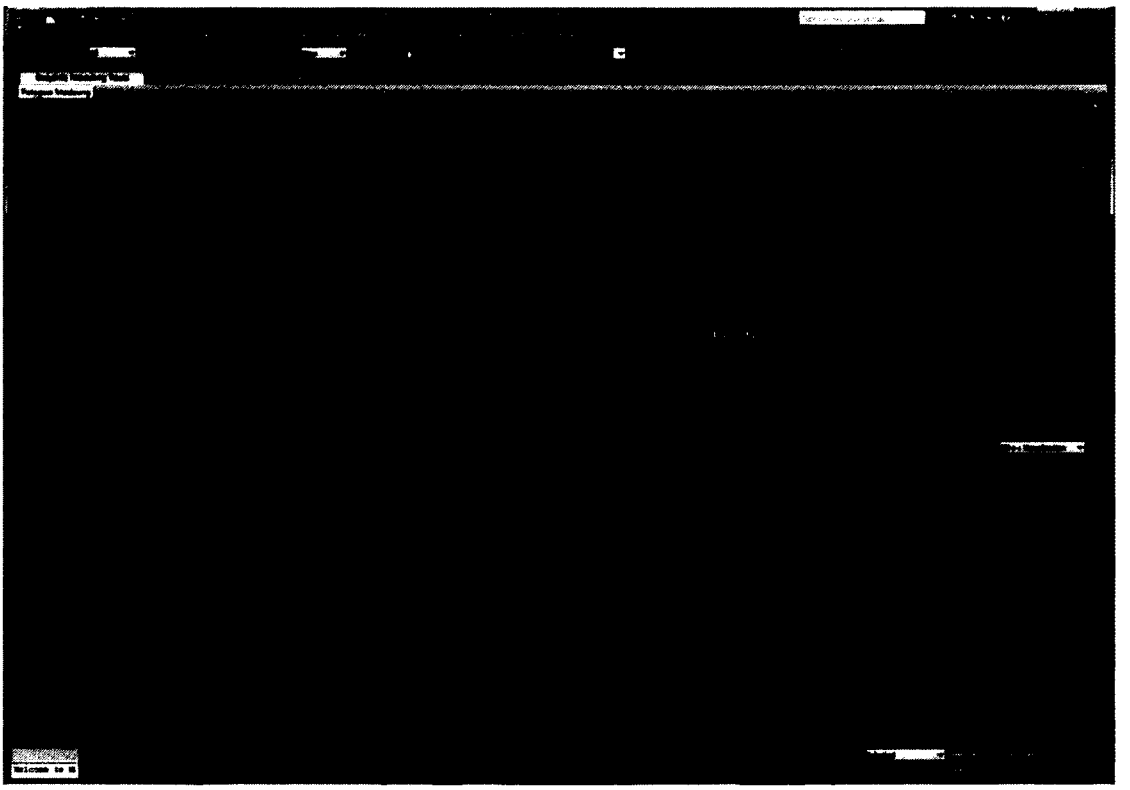


Figure 4.5: Interactive Environment System 3ds Max Interface

4.3 Summary and Discussion

In this chapter, we showed the structural and syntactic rules of the Cell-DEVS/BIM for designing a new VCELL framework that mainly aims to interoperate independently developed simulation systems. This chapter illustrated the implementation of the VCELL framework on Cell-DEVS/BIM and described the implementation of the

interactive environment system (*IES_Revit*) as an integration of Cell-DEVS formalism into BIM. The Cell-DEVS approach can be applied to improve and enhance the development of BIM. We discussed the details of the design and implementation of the visualization sub-system for output results of the Cell-DEVs simulation on the BIM model. The visualization sub-system (*IES_Max*) integrates the Cell-DEVS simulation and the BIM model to visualize the output result of the Cell-DEVS simulation on the BIM model. This will facilitate the improvement and enhancement of the BIM model, as we can visualize the output results directly on the BIM model, which is more effective for a 2D visualization of Cell-DEVS. It also helps decision-makers make decisions and modifications. CD++ is used as a toolkit for the Cell-DEVS models. We used the Autodesk Revit Architecture as a toolkit for BIM. The VCELL framework was applied and integrated with BIM models (this means that different simulations can be performed on the BIM models to improve and enhance the BIM simulation results). The VCELL framework is adaptive and collaborative, as we can obtain the actual building parameters and perform a simulation using the requested parameters, which results in the improvement and enhancement of the BIM simulation. The feasibility of the VCELL framework was verified using the DLA model. VCELL is open source, while other existing BIM toolkits are not. In addition, VCELL is more flexible than other existing BIM toolkits in terms of the ability to add new, desired features in a model, which allows user-specified algorithms to be used. Moreover, VCELL is more adaptive than other existing BIM toolkits in that a structure can be modified to fit a changed environment.

Chapter 5

VCELL for Emergency Management

5.1 Introduction

In this chapter, we show the use of VCELL for simulating an emergency. As emergencies are processes that are distributed over both time and space, emergency and disaster simulations should take into consideration the system evolution in both time and space. In this chapter we present an integrated emergency management system based on Cell-DEVS to develop new classes of cellular models for emergency response applications.

We focus on integrating emergency simulation with emergency management based on the collaboration of DEVS and Cell-DEVS formalisms. The emergency simulation is based on Cell-DEVS, and emergency management is performed by a robotic agent controlled by a DEVS model to respond to the emergency in real time. We also use a visualization engine that takes the results of the emergency simulation and the emergency management as input and produces 3D visualizations of the simulation scenarios.

As discussed in Chapter 3, we first selected the Cell-DEVS, DEVS, and visualization engine to design new a VCELL framework that mainly aims to facilitate the

interoperation of independently developed and adaptive simulation systems. The design methodologies presented in this chapter show how these were adapted for an emergency simulation, which mainly exposes emergency information as values where the simulation information flows from those models. It also shows that the system is flexible with regard to adding any new, desired features to a model, which allows user-specified algorithms to be utilized.

This chapter is organized as follows. Section 5.2 discusses the system archeticture. Section 5.3 shows the Cell-DEVS emergency model that we have designed using a real-time version of the CD++ toolkit. In Section 5.4, we present the emergency management mechanism using a DEVS-based robotic agent. Section 5.5 explains the visualization of the simulation. Section 5.6 describes the message structure transferred between the three components, followed by a summary and discussion of the work of this chapter in Section 5.7.

We introduce a simulation-driven architecture for integrating emergency simulation with robotic first responders moving towards the locations of the emergency, which are spread out on the field. The robot is placed on a grid corresponding to the simulated emergency area and reaches every location, dealing with the emergency. Our work differs from previous research in three ways:

- The RT cellular emergency simulation is an on-demand data source of the scenario, which is to be used by the robot. A supervisory control station can be used to update the emergency simulation data with that of a real emergency situation and information about the area.
- This multi-model combines a Cell-DEVS cellular model, a DEVS-based robotic controller, and virtual reality visualization. This component-oriented approach provides model reusability and interoperability, allowing for any of the components to be integrated or replaced.
- Using a simulation-driven approach for controlling the robot allows the robot

controller to be tested in a fully simulated environment, then the same model to be used to control a real robot. Model-continuity from early simulation stages to its final embedding on the hardware speeds up the development process while increasing the reliability of the product and reducing risk and cost.

5.2 The System Architecture

Fig. 5.1 shows the system architecture that integrates Cell-DEVS for emergency simulation, DEVS for emergency response, and 3D visualization to output the results visually. The system is composed of three collaborative sub-systems: the cellular emergency model implemented in Cell-DEVS, the emergency response by a robotic agent implemented in DEVS, and the visualization component that renders the 3D scenes. Each of these sub-systems runs on a different computer, communicating through messages sent over a network. All three sub-systems run in real time and update each other through messaging via a network infrastructure.

Fig. 5.1 also presents a more detailed overview of the system. The emergency simulation sub-system is in charge of the Cell-DEVS emergency model. It communicates with the DEVS emergency response sub-system, informing it about the dimensions of the emergency area, and sends updates regarding the location of a fire on the grid. At the same time, it also sends this information to the visualization sub-system, providing it with real-time data on the scene. The DEVS-based control model uses the emergency information received from the Cell-DEVS engine to carry out an emergency response. Based on these commands, the robot moves on the simulated grid area and extinguishes the fires in the emergency area, one after another. The emergency response sub-system dynamically updates the emergency simulation and the visualization sub-systems regarding the fires that have been extinguished. This process continues until all fires have been extinguished. The visualization sub-system

produces 3D scenes from the updates received dynamically from both the emergency simulation and the emergency response sub-systems.

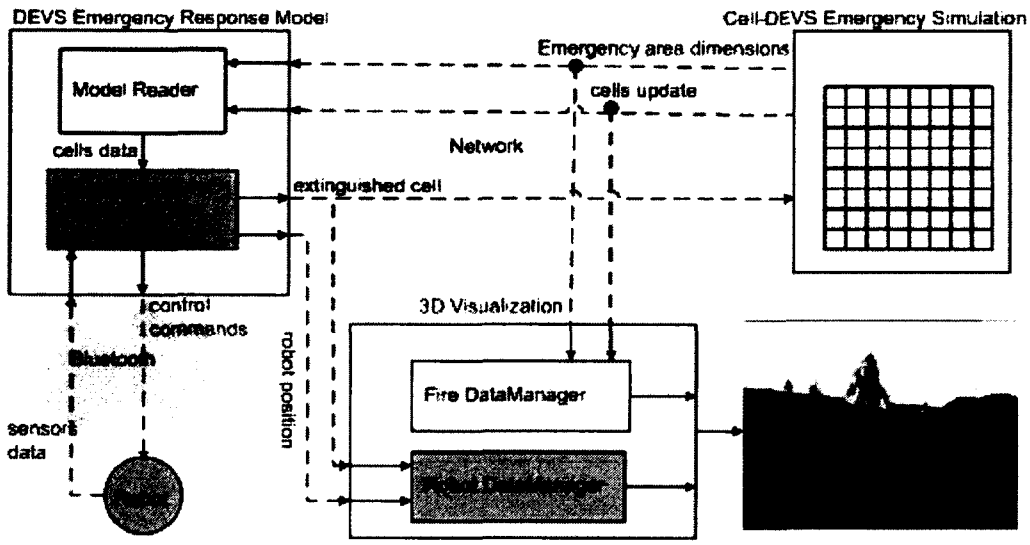


Figure 5.1: Detailed System Architecture [125]

5.3 Cell-DEVS Emergency Simulation

In this section, we present the use of VCELL for adding various hardware devices that allow hardware-in-the-loop. This achieves more accurate simulation results than when simulated hardware is used. It also improves training in different domains, such as in emergency training, by allowing real equipment to be used, as discussed in Chapter 3. VCELL also allows models to receive real-time external information, and the simulation parameters can be updated at any time due to the continuous-time nature of the discrete-event specifications. By using Cell-DEVS to model emergency situations, the actual area is modeled as a cell-space and is divided into cells. The emergency model represents an area that has a number of locations on fire (e.g., roadside bombs or explosions), which are ignited randomly during the simulation.

The model is defined according to the conventions of Cell-DEVS using the CD++ toolkit, as seen in Fig. 5.2. In order to run the model in real time, we modified the CD++ simulation engine. The virtual time-advance was replaced with a real-time version, allowing the emergency simulation to interact with the emergency response and the visualization sub-systems in real time. We also added a generic interface to the simulation engine, which enables it to interact with the external environment (e.g., a network). The simulator sends the dimensions of the cell-space at the beginning of the simulation and submits any cell updates, and at the same time receives input to the cellular model using the message structure that will be discussed in Section 5.6.

Figure 5.2: Emergency Cellular Space with Random Locations of Fires [125]

5.4 DEVS-based Robotic First Responder Agent

In this section, we note that Mohamed Moallemi worked on the details of the design and implementation of the DEVS-based controller for the autonomous first responder robot. The controller model can also be used for a robotic swarm [98], in which a large number of homogeneous autonomous robots are engaged in an activity. We used the e-puck robotic kit [99] as a prototype of a first responder robot to deploy the simulation-driven controller developed for the emergency response system. The e-puck is a small, mobile robot capable of moving and spinning and is equipped with sensors and motors. It uses 8 infrared distance proximity sensors to detect obstacles around it. There are 8 LEDs mounted on top in the shape of a ring. The robot can interact with a PC via Bluetooth connection. We executed the prototype model on a PC while interacting with the robot using a Bluetooth connection; the robot was programmed to listen to the commands received from the PC. However, the final goal is to develop a simulation-driven, embedded controller for the robot.

5.4.1 The Logical Controller

The two main operations of the robot to help human first responders are as follows:

- Deploying robots to gather information about the locations of emergency incidents
- To perform first-responder operations in the emergency locations and cooperate with human first responders.

Our focus in this work is on the second usage, in which we try to develop a DEVS-based, model-driven controller for an autonomous robot, which collaborates with the cellular emergency simulation engine. The robot tries to reach the locations of the fires and extinguish them one at a time, using the cellular space as a map of the region in which it is operating. Initially, the robot model receives the size of the cell-space

and builds a copy of the cellular space for itself. As the cellular model develops in real time, the robot also receives updates of cell values from the cell-DEVS model and marks the changes in its own copy. The robot controller model consists of two levels of controls, a higher-level and a lower-level controller. The higher-level controller is responsible for planning the path toward the closest emergency location (fire) using the data provided by the cell-space, while the lower-level controller is responsible for avoiding the obstacles in the path.

5.4.2 DEVS Controller Model Specifications

The robot model interacts with the cellular emergency model and the visualization engine. The model responsibilities are divided into two parts, constructing two main components in the model. The Model Reader is responsible for creating the local cell-space, updating the cell-space by receiving the updates from the Cell-DEVS engine, and signaling the Controller component periodically to make a decision on path-planning. The Controller component is responsible for implementing the HLC and LLC algorithms, sending control commands to the robot and informing the visualization engine about the robot movements.

Fig. 5.3 depicts an abstract representation of the behavior of the two components in DEVS graph format. The DEVS graph state diagram [126] summarizes the behavior of a DEVS atomic component by rendering the states, transitions, inputs, outputs, and state durations of the atomic component graphically. The continuous edges between the states represent external transitions, with the input port, the input value, and any condition on the input. The discrete lines represent internal transitions with the associated outputs.

The Model Reader starts in the wait for dimension state, where it waits to receive the dimensions of the cell-space from the Cell-DEVS engine. As soon as it receives the

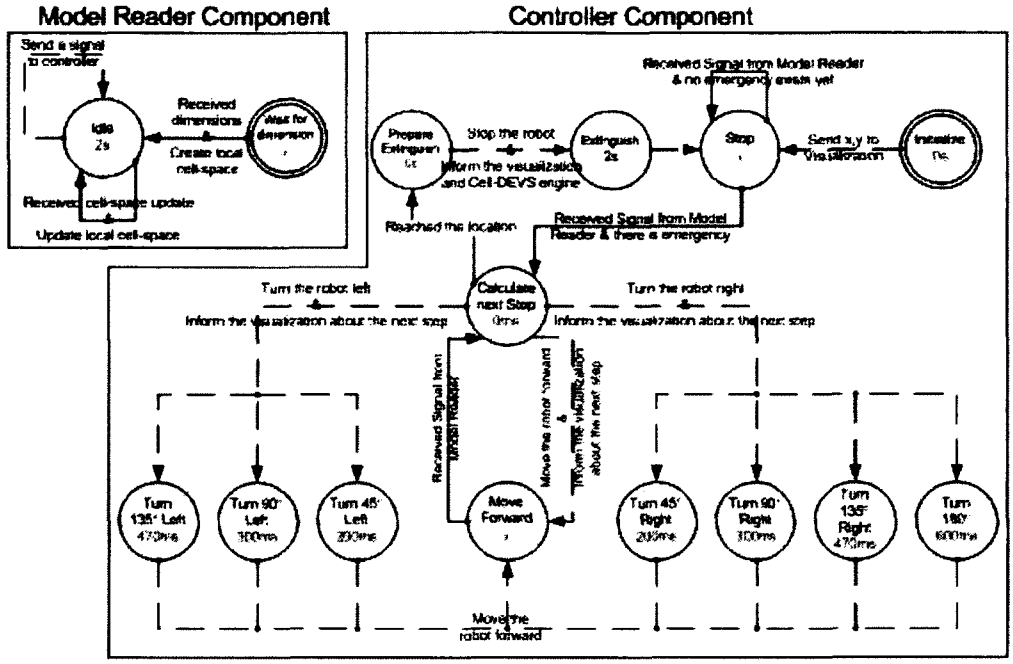


Figure 5.3: DEVS Graph State Diagram of the Robot Controller [125]

dimensions, it creates the local cell-space matrix and then transfers to the idle state. The idle state has a limited period, which corresponds to the movement period of the robot. During the idle state, the Model Reader also receives cell-space updates from the Cell-DEVS engine and marks them on the local cell-space, and if an emergency update is received, it adds it to the emergency list. At the end of this state, the Controller is signaled to carry out the next movement. The Controller starts by sending the initial position of the robot to the visualization engine and transitions to the stop state, where it receives periodic signals from the Model Reader. If there is an emergency location in the emergency list, the Controller is transferred to the calculate next step state, and the following tasks are executed in the corresponding external transition function: sort the emergency list, find the closest one, apply the HLC and LLC algorithms, and calculate the next step. Based on the result of the two-level control algorithms, the Controller is transitioned to one of the movement states and

in the output function, the movement commands for the robot and the next step information for the visualization engine are outputted. The Controller continues this sequence until it reaches the emergency location, when it is transferred to the prepare extinguish state. At the end of this state, it outputs the stop command to the robot, informs the Cell-DEVS and visualization engines about the emergency restraint, and transitions to the stop state, where it waits for the next emergency location.

5.4.3 Controller Model Implementation on E-CD++

E-CD++ is an open-source, embedded, real-time DEVS-based modeling, simulation, and application development environment [127], built as an extension to the CD++ simulator. The models are developed incrementally in an Eclipse-based environment in C++ language and then embedded in the target hardware. E-CD++ deploys real-time services offered by the underlying Xenomai real-time Linux kernel to execute the model, providing a reliable formal platform for real-time application development. The model structure is declared in a model file and an optional event file supplies the virtual inputs to the DEVS model. E-CD++ allows the model to be run as a simulation in virtual time and real time, and also as a real system interacting with the actual hardware counterparts. ECD++ allows for the definition of the driver interface functions for each input and output port of a DEVS model, in which the integer I/O values of a DEVS system are translated to signals to the external environment.

5.5 3D Visualization Engine

Visualization of emergency behavior can provide a number of benefits. First, it provides scientists with an interactive environment to verify the accuracy of these models by comparing the results of an actual emergency with the output of a simulated version. Once the model is validated, it can then be used to predict not only the behavior

of an existing emergency, but also the consequences of preventative measures, such as vegetation thinning and prescribed burns. Displaying these predictions in a visually informative manner allows emergency departments to better educate the first responders on existing emergency hazards. Furthermore, enabling interactive manipulation of the simulation along with the visualization allows emergency leaders to be trained with respect to resource allocation and emergency behavior. While it would be risky and costly to experiment in a real-life situation, these risks can be mitigated by simulating untested approaches first. 3D user interfaces provide a more intuitive form of interaction. Additionally, high-fidelity graphics enable an observer to better compare a simulated emergency with a historic emergency. In the following sub-sections, we show that VCELL is not specific to any implementation, as we need the output results. This implies that new components can be added without changes having to be made at the code level or in the framework architecture, as discussed in Chapter 3.

5.5.1 Three-dimensional Visualization Engine Description

The 3D visualization engine is used to visualize the simulation output results of both the emergency simulation model and the robotic first responder agent. The visualization engine is implemented using Vega Prime [128] and OpenGL . Vega Prime is a high-performance software environment and toolkit for real-time simulation and virtual reality applications. It serves as an API consisting of a GUI called LynX Prime and Vega Prime libraries and header files of C++-callable functions.

The 3D scenes are rendered using 3D openflight models. The terrain model consists of trees, different buildings, roads, etc. The DEVS-based robotic agent is represented by a 3D emergency truck model. We can control the environmental effects and time of day in the 3D scene visualization.

A 3D scene, as shown in Fig. 5.4 and Fig. 5.5, is displayed in a window that is

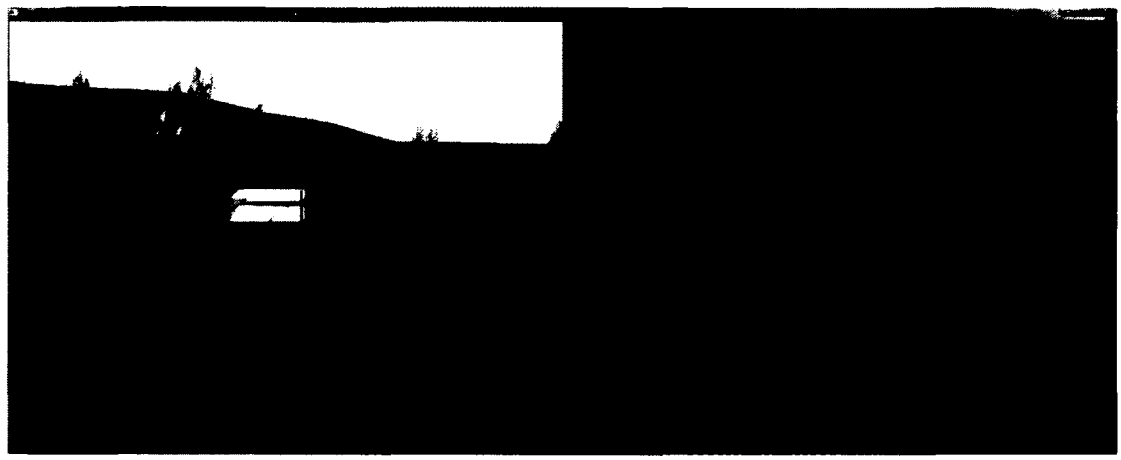


Figure 5.4: Three-dimensional Visualization Engine Cellular Map [125]

divided into two channels: one for a perspective view of a 3D scene (on the left), and the other for an orthographical view of the 3D scene, which acts as a 2D map of the area (on the right).

- In the perspective view in the first channel, the movement of the emergency responder truck is displayed, which is the 3D model representing the robot, and is observed using a fixed camera. The observer view can be changed to five positions: back, front, left side, right side, or rotate around the emergency responder truck.
- In the orthographical view in the second channel, a red grid is created that represents the cellular grid of the simulated emergency area (see Fig. 5.4).

The locations of the fires received from the Cell-DEVS engine are rendered by flashing yellow circles, and the emergency responder truck is represented by a white circle (see Fig. 5.5). The white circle changes to green when the robot extinguishes a fire in the scene, after which the fire special effects and the flashing yellow circle are removed from the 3D scene. The orthographical view is capable of zooming in and

Figure 5.5: Three-dimensional Visualization Engine Zoomed Map [125]

out and the cellular grid can be removed for a better view (see Fig. 5.5).

5.5.2 3D Visualization Engine Implementation

The visualization program is implemented in Visual C++. The 3D visualization subsystem consists of two main components:

- The *Receiver*, which receives the data from the DEVS-based robot model and the Cell-DEVS emergency model
- The *Visualizer*, which is responsible for the display of the visualization scene.

In the *Receiver* component is a separate thread spawned to receive the emergency and suppression data. The data received (such as the positions of the fire truck, the robotic first responder agent, and the fire) are transformed from the 2D grid position to the 3D visualization scene position.

In the *Visualizer* component, there are five sub-components:

- DrawGrid, which receives the cellular space dimensions from the Receiver and draws them on the 2D map channel.

- DrawCircle, which receives the fire positions from the Receiver during the execution and renders flashing yellow circles (the fire) at the corresponding coordinates. It also obtains the position of the robot, draws a white circle at the corresponding position, and changes the circle to green when the robot has extinguished a fire.
- CreateEmergency, which receives the positions of the fires and applies them to the corresponding positions by rendering 3D special effects of the fire.
- MotionModel, which receives the initial coordinates of the robot, the period of each step of the robot, and the next movement. It then creates a motion model for the robot in the 3D visualization scene.
- RemoveEmergency, which receives the coordinates of the fire that is extinguished by the robotic first responder agent and removes the special effect of the fire from this location.

The 3D visualization engine is capable of deploying different 3D terrain openflight models and different cellular areas (dimensions and initial values) without changing the code of the visualization. As a result, we see that VCELL is not specific to any implementation, as we read the output simulation results from the Cell-DEVS. This means that we can add new components without having to make changes at the code level or in the framework architecture, as discussed in Chapter 3. We can also visualize the output simulation on the 3D visualization scene, which improves and enhances decision-making.

5.6 Global Message Structure

The collaboration of the three components in this project is based on a global message structure transferred over a network infrastructure. The network_struct contains the following five data fields:

1. *msg_id*: an integer data type used to decode the type of the message and the value of the next fields in the message. There are generally five types of message:
   - The dimension message carries the size of the cell-space from the Cell-DEVS engine to the DEVS and visualization at the start of the execution.
   - The robot initial location message carries the initial coordinates of the robot from the DEVS engine to the visualization.
   - The cell-space update message carries the cell value changes during the execution from the Cell-DEVS engine to the DEVS and visualization.
   - The next movement message carries the direction of the next movement at the start of each step from DEVS to the visualization engine.
   - The extinguish message carries the location of the fire that has been extinguished by the robot, from the DEVS sub-system to the Cell-DEVS and visualization sub-systems.
2. x: used to carry the horizontal axis value (the horizontal dimension or the horizontal coordinate).
3. y: used to carry the vertical axis value (the vertical dimension or the vertical coordinate).
4. *dir*: carries the next direction.
5. *value*: carries the value of the cell and is used in the cell update message.

These messages are embedded in a UDP packet and transferred during the execution of the model through the network. However, for the next stages of the project we are planning to use a TCP protocol, which is more reliable and prevents packet loss. The design of the system is such that the number of messages transferred through the network is as low as possible, thus preventing delay in the message transfer.

5.7 Summary and Discussion

We have a DEVS-based emergency management simulation and visualization system. The system offers a robust software framework to make a real-time emergency response system more flexible and more scalable. The Cell-DEVS sub-system allows models to receive external information, and the simulation parameters can be updated at any time, due to the continuous-time nature of the discrete-event specifications. A robotic agent acting as a first responder is placed in a virtual environment generated from a Cell-DEVS emergency simulation. The controller of the robot is a DEVS-based emergency response model that interacts with the emergency simulation through messaging and is informed about the map of the area and the location of the incident (e.g., roadside bombs, fire, explosions, etc). Both the emergency simulation and the emergency response sub-systems run in real time and communicate with each other and with a 3D visualization engine. The purpose of the visualization system is to generate 3D scenes and to visually monitor the activities of the robotic first responder. Although the emergency model is a simulation, it is simple to replace it with more complex emergency simulation models or a real emergency database fed from real-world data. The generic interface and message structure that enable the emergency simulation, the emergency response, and visualization sub-systems to interchange data also allow our system to simulate emergency management in real time under various conditions. Moreover, it can be integrated with stochastic optimization models that use the scenario results from the simulation to determine an optimal mix of emergency planning resources to dispatch to an emergency situation. Our system is intended not only to train emergency response personnel, but can also be used as a core real-time strategy and response system. VCELL is open source, while most emergency simulation models are not. Likewise, VCELL is more flexible than other selected emergency simulation models in terms of the ability to add any

new, desired features to a model, which allows user-specified algorithms to be used, while other selected emergency simulation models are restricted to a few algorithms. Moreover, VCELL is more adaptive than other existing emergency simulation models and includes good analysis capabilities. It also supports 3D visualization, which is not applicable in other selected emergency simulation models.

Chapter 6

A Real-time Visual Simulation in Support of Combat

6.1 Introduction

In this chapter, we show the use of VCELL for a land combat simulation. Since land combat movement algorithms can be distributed over both time and space, land combat simulations should take into consideration the system evolution in both time and space. In this chapter, we present a collaborative land combat model based on the Cell-DEVS formalism and 3D real-time visualization to develop new classes of land combat movement algorithms.

We focus on collaboration between an agent based on Cell-DEVS formalism and a visual agent simulation based on a 3D real-time visualization simulation. The visual agent simulation allows us to visualize the land combat simulation scenarios in a 3D scene.

As discussed in Chapter 3, we first selected the Cell-DEVS, DEVS, and visualization engine to design the new VCELL framework, which mainly aims to facilitate the interoperation of independently developed and adaptive simulation systems. The design methodologies presented in this chapter show how these were adapted for a

land combat simulation, which mainly show land combat information as values where the simulation information flows from those models. It also shows that the system is flexible with regard to adding any new, desired features to a model, which allows for user-specified algorithms to be utilized. VCELL supports 3D visualization, which is not applicable in other ABDs.

This chapter is organized as follows. Section 6.2 discusses the system architecture and its components. Section 6.3 describes the Cell-DEVS agent model that we have designed using a real-time version of the CD++ toolkit. The modifications made to the simulation engine to enable real-time execution are also pointed out and the message structure transferred between the components. In Section 6.4, we present a description of the visualization sub-model and its implementation. Section 6.5 explains the scalability of the system, followed by a summary of the work in this chapter in Section 6.6.

We present a collaborative 3D real-time visual cellular agent model (VCELL) and a cellular agent simulation in real time. The agents are divided into two teams: the blue team in the 3D visualization agent sub-model, and the red team in the Cell-DEVS agent sub-model. These sub-models collaborate via a network connection. Our work differs from previous research as follows:

- This work incorporates two components: a Cell-DEVS agent simulation and a 3D visualization agent simulation. This component-oriented approach provides model reusability and interoperability, allowing for integration or replacement of any of the two components.
- It uses a VCELL model to remove the random movement problem in the blue team, providing a 3D visualization agent that retrieves the real position of agents in combat. This guarantees the combat simulation output for the blue team.
- The 3D visualization agent simulation is an on-demand data source for the

combat scenario. Real fighters can be invoked in the 3D visualization agent simulation to update the agent simulation data with real situation data.

6.2 The Visual CELL-DEVS Agent

In this section, we present the collaboration of the VCELL components and how they are connected to each other. VCELL also allows models to receive real-time external information, and the simulation parameters can be updated at any time due to the continuous-time nature of the discrete-event specifications, as discussed in Chapter 3. The Visual CELL-DEVS Agent (VCELL), as shown in Fig. 6.1, is composed of two main subsystems:

1. The CellAgent sub-model, which is implemented using a Cell-DEVS model running in real time
2. The 3D real-time visual simulation (RTV) sub-model, which is implemented using Vega Prime and OpenGL.

The CellAgent sub-model and the 3D real-time visualization sub-model each run on a different machine in real time and communicate via messages transferred through a network infrastructure. VCELL is a multi-agent simulation combat system that facilitates the analysis and understanding of land combat, and 3D real-time visualization simulation for tactics in land combat. By using VCELL, not only can the analysts understand the overall shape and dynamics of a battle and know the output of an operation, but combatants can also be trained in the 3D visual real-time simulation system. An agent in VCELL is characterized by certain properties, such as capabilities, movements, communications, and health. Agents can communicate by exchanging messages. Health can be defined as the level of energy of an agent, which is defined by users. When an agent is attacked by the opponent agent type, its

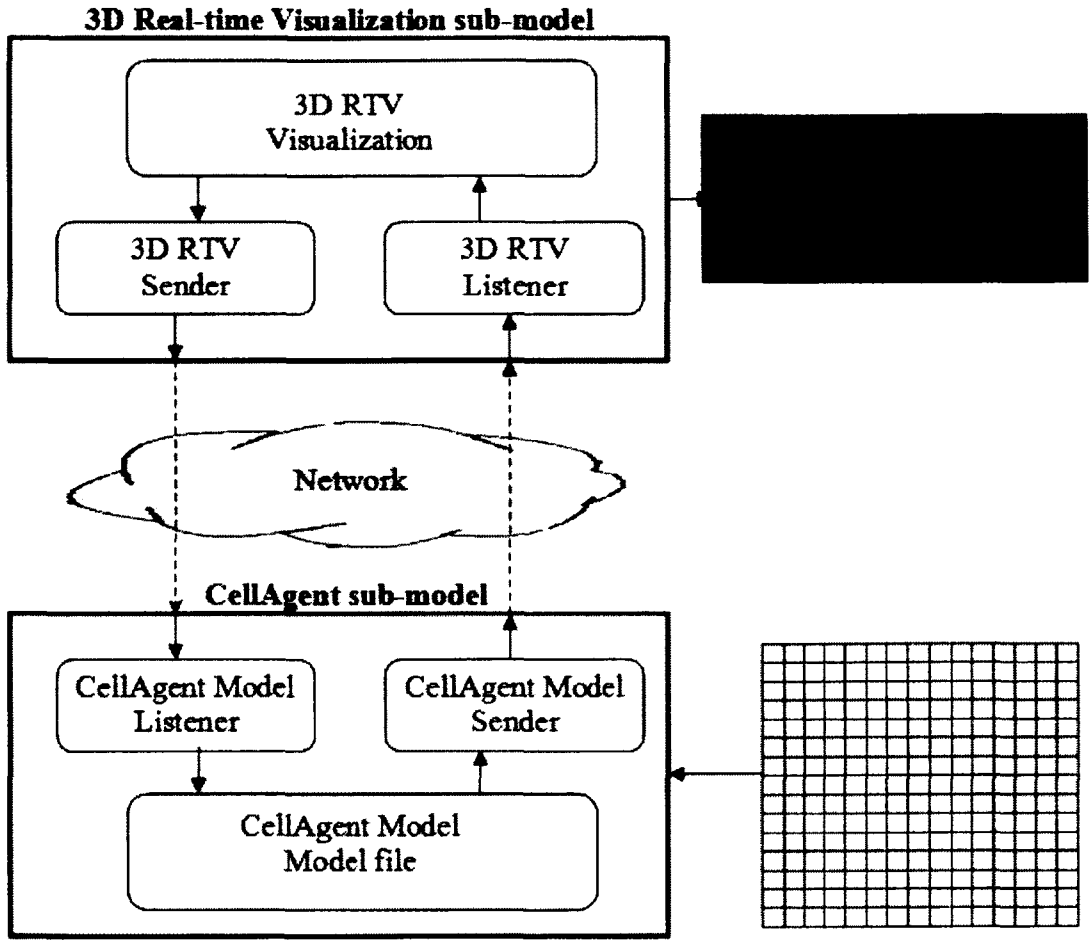


Figure 6.1: The Visual CELL-DEVS Agent (VCELL) Architecture

health depends on the number of the neighborhood agents and their health strength. Users can import different 3D terrain types in the real-time visual model. The type of terrain affects the agent's movements.

The agent's movement depends on five different weights: agent healthy friend, agent injured friend, agent healthy opponent, agent injured opponent, and the flag. The movement is calculated at each simulation time step for each agent. The agent can move to another cell or decide to stay in the same cell. No cell in the space can be occupied by more than one agent at a time. The decision-making used by each

agent to decide on the direction in which to move depends on the agent's personality in the movement algorithm. The movement algorithms used in VCELL are the same as in EINSTein and MANA, but VCELL is not restricted to these algorithms only, as we can apply different movement algorithms. The Movement Algorithm of EINSTein uses equation (6.1) to compute the penalty for the next location [110, 113]:

$$Z_{new} = \left(\frac{W_E}{E * R_S\sqrt{2}} \sum_{i=1}^{E} D_{i,new} \right) + W_F \left(\frac{D_{F,new}}{D_{F,old}} \right) \tag{6.1}$$

where

R_S Sensor range of agent about to move;

E Number of enemy entities within sensor range;

W_E Weighting towards enemy agents;

$D_{i,new}$ Distance to the ith enemy from the new location;

W_F Weighting towards the flag;

$D_{F,new}$ Distance to the flag from the new location;

$D_{F,old}$ Distance to the flag from the current location.

The Movement Algorithm of MANA uses equation (6.2) to compute the penalty for the next location [114, 118]:

$$\begin{aligned} Z_{new} = & \left(\frac{W_E}{100 * E} \right) \left(\sum_{i=1}^{E} \frac{D_{i,new} + (100 - D_{i,old})}{100} \right) \\ & + \left(\frac{W_F}{100} \right) \left(\frac{D_{F,new} + (100 - D_{F,old})}{100} \right) \end{aligned} \tag{6.2}$$

where

E Number of enemy entities within sensor range;

W_E Weighting towards enemy agents;

$D_{i,new}$ Distance to the ith enemy from the new location;

$D_{i,old}$ Distance to the ith enemy from the current location

W_F Weighting towards the flag;

$D_{F,new}$Distance to the flag from the new location;

$D_{F,old}$ Distance to the flag from the current location.

Agents are encouraged to move closer to the opponent agent. The agent will always move to the cell with maximum weight. There is no tie in the real-time simulation sub-model, but in the Cell-DEVS simulation sub-model, the agent randomly selects a cell between the cells in the tie. This kind of randomization may affect the stability of the solution in the enemy section only; however, it is not a serious problem, as the enemy section is based on our assumptions

6.3 Cell-DEVS Agent Sub-model

In this section, we show that VCELL serves as a container to hold different software components without being specific to any implementation. This allows new components to be added without having to make modifications at the code level or in the framework architecture. VCELL also allows models to receive real-time external information, and the simulation parameters can be updated at any time, due to the continuous-time nature of the discrete-event specifications. We explain how VCELL reduces the time taken for scenario development, modification, and validation using the simple rules in the Cell-DEVS simulation, as discussed in Chapter 3.

6.3.1 Real-time Cell-DEVS

In this subsection, we note that we worked collaboratively with Mohamed Moallemi for the time advances in the DEVS and Cell-DEVS models based on the availability of the events. Thus, the simulation runs in virtual time in which, after servicing every

event, the simulation time advances to the next scheduled event time. To visualize the agent model, we need to run the agent model simulation in real time, so that the events can be transferred to the visual engine, resulting in a real-life visualization of the battlefield. CD++ is designed and implemented based on the DEVS abstract simulation mechanism. A Root Coordinator object acts as a coordinator with the top-coupled component in a CD++ model, which is responsible for advancing the time to the next event time and also sending and receiving the I/O of the DEVS model.

We modified the Root Coordinator event scheduler function to work in real time, in which the events are served at the time they are serviced and the time advances are based on the wall clock time. We added two new features to the CD++ simulator, which make CD++ capable of receiving DEVS inputs from the network and injecting them into the model, at the same time sending outputs of the DEVS model to the network. A separate thread was added to the CD++ software structure to allow it to listen to the network inputs without interrupting the main execution sequence. The input thread executes an added function of the Root Coordinator, which creates a network socket and listens to the network in a blocking mode. As soon as a network packet is received, the content of the packet is extracted and saved in the input bag of the Root Coordinator, which will service the input.

In order to send inputs to any specific atomic cell, we modified the CD++ MainSimulator post-registration function, which is responsible for creating the DEVS ports defined in the model file. In the modified version, the MainSimulator creates a default input port for each atomic cell, as shown in Fig. 6.2. These ports are used later by the Root Coordinator to inject inputs into the specific cell based on the coordinates indicated in the network message. Once an input is received, the Root Coordinator sends an input message to the input port of the Top coordinator, which is connected

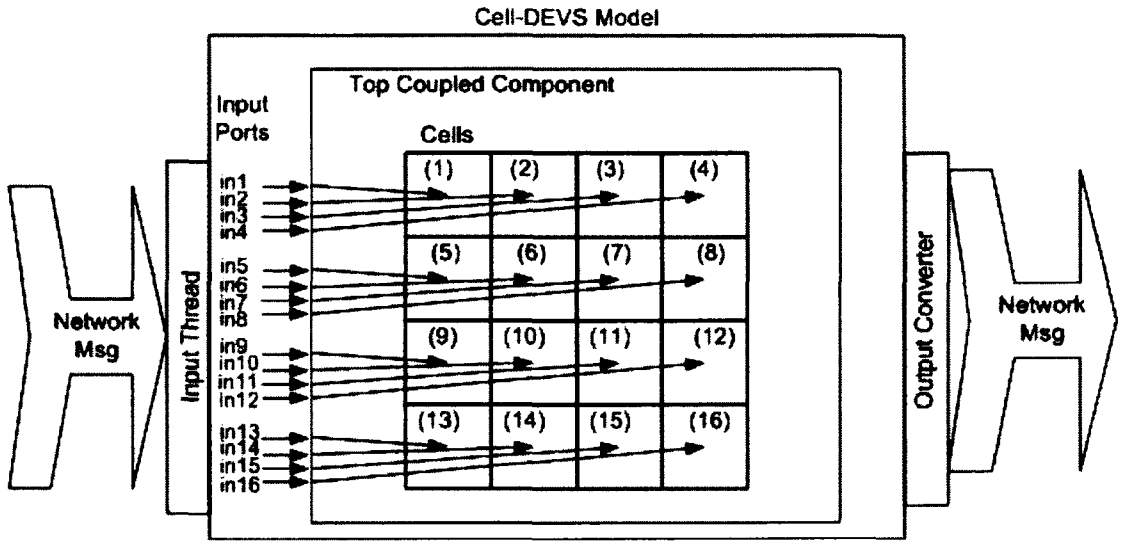


Figure 6.2: Sample Cell-DEVS Model Structure and Interfaces [129]

to the specific cell that is the destination of the message.

To submit the cell value changes, we added a function to the Root Coordinator that extracts the outputs of atomic cells from the Y messages (the output carrying message defined in the DEVS abstract simulation algorithm) and sends them to the network.

6.3.2 Global Message Structure

The collaboration of the sub-models is based on a global message structure transferred over a network infrastructure. The network_struct contains the following five data fields:

1. *msg_id*: an integer data type used to decode the type of message and the value of the next fields in the message. There are generally three types of messages:
   (a) The *dimension message* carries the size of the cell-space from the CellAgent sub-model to the 3D real-time visualization sub-model at the start of the execution.

(b) The *cell-space update message* carries the cell value changes during the execution from the CellAgent sub-model to the 3D real-time visualization sub-model. It also carries the initial coordinates and personalities of the blue agent from the 3D real-time visualization sub-model to the CellAgent sub-model and the initial coordinates of the blue agent's flag from the 3D real-time visualization sub-model to the CellAgent sub-model.

(c) The *visualization agent update message* carries the visualization changes during the execution from the 3D real-time visualization sub-model to the CellAgent sub-model. It also carries the initial coordinates of the red agent's flag from the CellAgent sub-model to the 3D real-time visualization sub-model and the initial coordinates of the blue agent's flag from the 3D real-time visualization sub-model to the CellAgent sub-model.

2. x: used to carry the horizontal axis value (the horizontal dimension or the horizontal coordinate).
3. y: used to carry the vertical axis value (the vertical dimension or the vertical coordinate).
4. z: used to carry the layer axis value (the layer dimension or the layer coordinate).
5. v: used to carry the value of the cell (x,y,z).

These messages are embedded in a UDP packet and transferred during the execution of the model through the network. The design of the system is such that the number of messages transferred through the network is a low as possible, thus preventing delay in the message transfer.

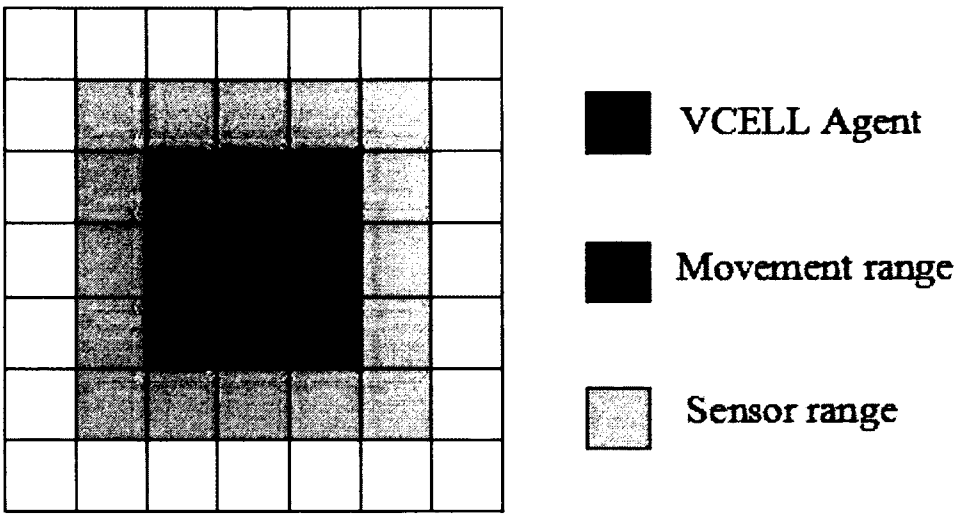


Figure 6.3: Movement and sensor range of VCELLA [129]

6.3.3 Cell-DEVS Agent Definition Model

The basic element of our CellAgent sub-model is a VCELL Agent (VCELLA), which represents a primitive combat unit (tank, transport vehicle, etc.). The combat battlefield is represented in the CellAgent sub-model as a two-dimensional cell space, as shown in Fig. 6.3. Each cell in the space can be occupied by the red agent of VCELLA. Each red agent can move to the next cell in the movement range or stay in the same cell. The sensor range is the area that is defined for each red agent to obtain the available number of friendly and enemy agents and their personality values. The user defines the dimension of the combat battlefield and the initial state of the red VCELLA agents at diagonally opposite corners to the red agents in the VisualAgent sub-model. The red flag is also positioned in the red VCELLA corner. The goal of the red VCELLA is to reach the blue flag successfully. The Combat CellAgent sub-model is defined using the modified CD++ version described in Subsection 6.3.1.

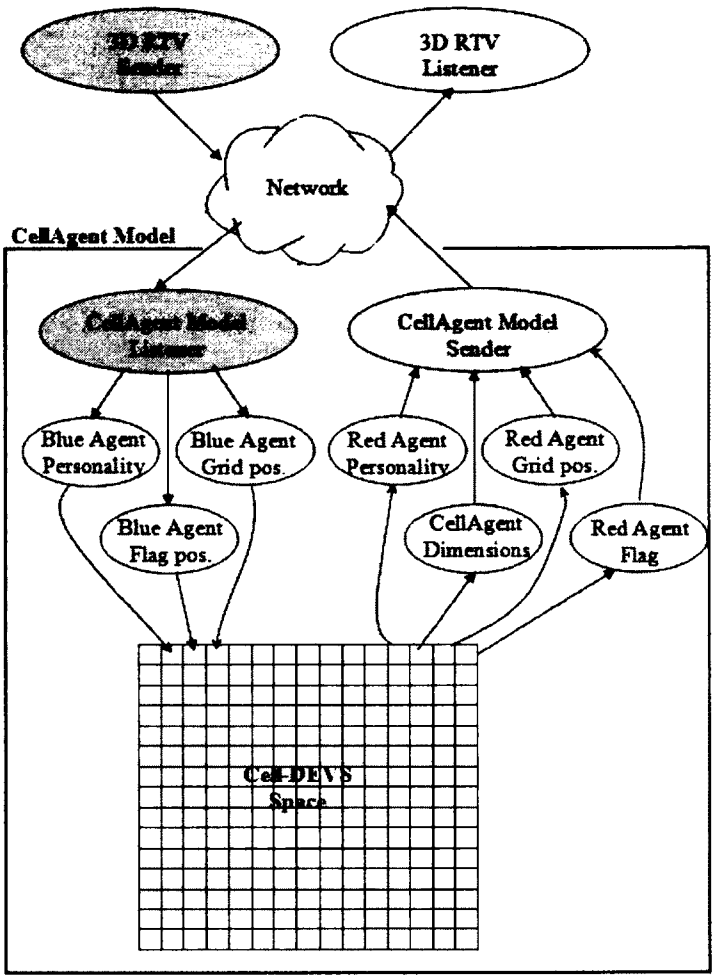


Figure 6.4: CellAgent Model Implementation in CD++ [129]

Fig. 6.4 illustrates the CellAgent sub-model in CD++ with the specifications and the network interface. The CellAgent sub-model is composed of CD++ O/P Driver Component CD++ Input thread. CD++ O/P Driver Component sends the battlefield dimensions to start up the VisualAgent sub-model. Then it sends the initial and updated values of the red agents, their weights, and the red flag position in real time to the VisualAgent sub-model. The CD++ Input thread receives the initial and updated values of the blue agents, their weights, and the blue flag position in real time from the VisualAgent sub-model. Then it invokes these values in the cell

space of the model.

The model file of the CellAgent sub-model reads the initialization data from a CD++ associated value file. The value file contains the initial values of the red agents, their weights, and the red flag position. The dimensions of the battlefield are defined in the model file. The model file is composed of six layers. The first layer contains the red agents, the second layer has their weights, and the third contains the red flag position, which gets its initial values from the value file. The other three layers are allocated to the blue agents, their weights, and the blue flag position, which get its values from the 3D real-time simulation sub-model.

The personality of VCELLA can be defined by the weight towards enemy agents and the weight towards the enemy flag which specify how VCELLA interacts with information within its sensor range. Each VCELLA has one of three states: alive, injured, or killed. The health state, $0 \leq H \leq 1$, is the measure of an agent's health. The agent's health can be defined as shown in equation (6.3):

$$H = H * \sigma\left(\frac{1 + F - E}{F}\right) \tag{6.3}$$

where

F Number of friendly entities within sensor range;

E Number of enemy entities within sensor range;

σ Function of x as defined in (6.4);

$$\sigma(x) = \begin{cases} 1, & 1 < x \\ x, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}, \tag{6.4}$$

As discussed earlier, the CellAgent sub-model can interact with the 3D visualization in real time. We have also added a generic interface to the simulation engine that enables it to interact with the external environment (e.g., a network). The simulator sends the dimensions of the cell-space at the start of the simulation, submits any cell updates, and at the same time receives input to the cellular model using the message structure described in Subsection 6.3.2.

Based on the above and as discussed in Chapter 3, we show that VCELL is flexible, as we can add any new desired features to a model. VCELL is more adaptive, as the model is modified to fit a changed environment and we can add user-specified algorithms.

6.4 Three-Dimensional Real-Time Visualization

3D visualization of combat can provide a number of benefits. First, it provides decision-makers with an interactive environment to verify the accuracy of these models by comparing the results of actual combat with the output of a simulated version. Once the model is validated, it can then be used to predict the behavior of existing combat. Displaying these predictions in a visually informative manner allows decision-makers to understand the view of the situations and their soldiers and the battle in order to make more effective decisions. Furthermore, interactive simulation along with the 3D visualization allows trainers to apply different combat tactics and enemy behaviors. While real training would be risky and costly to perform, these risks can be minimized by simulating untested approaches first. 3D visual interfaces provide more understanding of interaction. Additionally, high-fidelity graphics enable an observer to better compare a simulated combat with a traditional 2D visualization. In the following sections, we show that VCELL is not specific to any implementation,

as we need the output results. This implies the ability to add new components without having to make modifications at the code level or in the framework architecture, as discussed in Chapter 3.

6.4.1 Visualization Sub-model Description

3D real-time visualization is used to visualize the simulation output results of the CellAgent sub-model and also to implement a collaborative model that shares its components on two different simulation engines. The visualization renders the red agents and the creation of the blue agents and their characteristics based on their movement algorithm and personalities. The 3D real-time visualization model is implemented using Vega Prime and OpenGL. Vega Prime is a high-performance software environment and toolkit for real-time simulation and virtual reality applications. It serves as an API consisting of a graphical user interface called LynX Prime and Vega Prime libraries and C++ callable functions.

In the 3D real-time visualization sub-model, the combat can be seen in a 3D view for both red and blue agents. The blue agent personality calculations and position updates are done in the real-time visualization (RTV) sub-model, based on the data received regarding the red agents. The red agents' personalities and positions are received from the CellAgent sub-model in real time. The CellAgent sub-model obtains the blue agents' personalities and positions from the RTV in real time.

The 3D scenes are rendered using 3D Openflight models. The terrain model consists of trees, buildings, roads, and so on. The agents are represented by a 3D tank model. We can control the environmental effects and the time of the day in the 3D scene visualization. As illustrated in Fig. 6.5, a 3D scene is shown in a window that is divided into two channels: one with a perspective view of the 3D scene (on the left), and the other with the orthographical view of the 3D scene, which acts as

Figure 6.5: 3D Real-time Visualization View [129]

a 2D map of the area (on the right).

In the perspective view in the first channel, a 3D model for each agent (3D tank object) is displayed. The 3D scene is observed using a fixed camera. The observer view can be changed to five positions: back, front, left side, right side, or rotate around the object.

In the orthographical view in the second channel, a yellow grid is created, representing the cellular grid of the simulated combat area. The red agents' positions received from the CellAgent sub-model are rendered by red circles and the blue agents' positions represented by blue circles (see Fig. 6.5). The orthographical view can be zoomed in and out and the cellular grid can be removed for a better view.

6.4.2 RTV Sub-model Implementation

Fig. 6.6 illustrates the hierarchy of the 3D real-time visualization sub-model, which was implemented in Visual C++ and consists of three main components:

1. The *RTV Listener*, which receives the red agents' data from the CellAgent

sub-model

2. The *RTV Visualization*, which is responsible for creating the blue agents and the display of the 3D visualization scene
3. The *RTV Sender*, which sends the blue agents' data to the CellAgent sub-model.

The *RTV Listener* is a separate thread, spawned to receive the red agents' data. First, the *RTV Listener* is responsible for receiving the dimensions of the cell-space from the CellAgent sub-model in order to start the *RTV Visualization* to render the 3D scene. Then the *RTV Listener* receives the red agents' flag position. Finally, the *RTV Listener* receives the red agents' grid positions updates and their personalities in real time from the CellAgent sub-model.

The *RTV Visualization* is the main part of the 3D real-time visualization sub-model. The *RTV Visualization* is responsible for setting up the 3D visualization of the 3D real-time visualization sub-model.

The *RTV Visualization* is composed of six main modules:

1. *3D Scene Generator*, which is responsible for setting up and synchronizing the 3D scene, drawing different 3D objects (terrain, tanks, buildings, etc.), defining and controlling different environmental effects (daytime, clouds, sun, etc.), and drawing the other modules. The 3D Scene Generator also removes the dead agents from the 3D scene.
2. *3D Red Agent Generator*, which creates a 3D object for the red agents. It positions the red agents based on the coordinates received from the *RTV Listener*. The 3D Red Agent Generator sets and updates the red agents with their personalities, which are received from the *RTV Listener*.
3. 2D Map Draw, which consists of:
    - DrawGrid, which receives the cellular space dimensions from the RTV

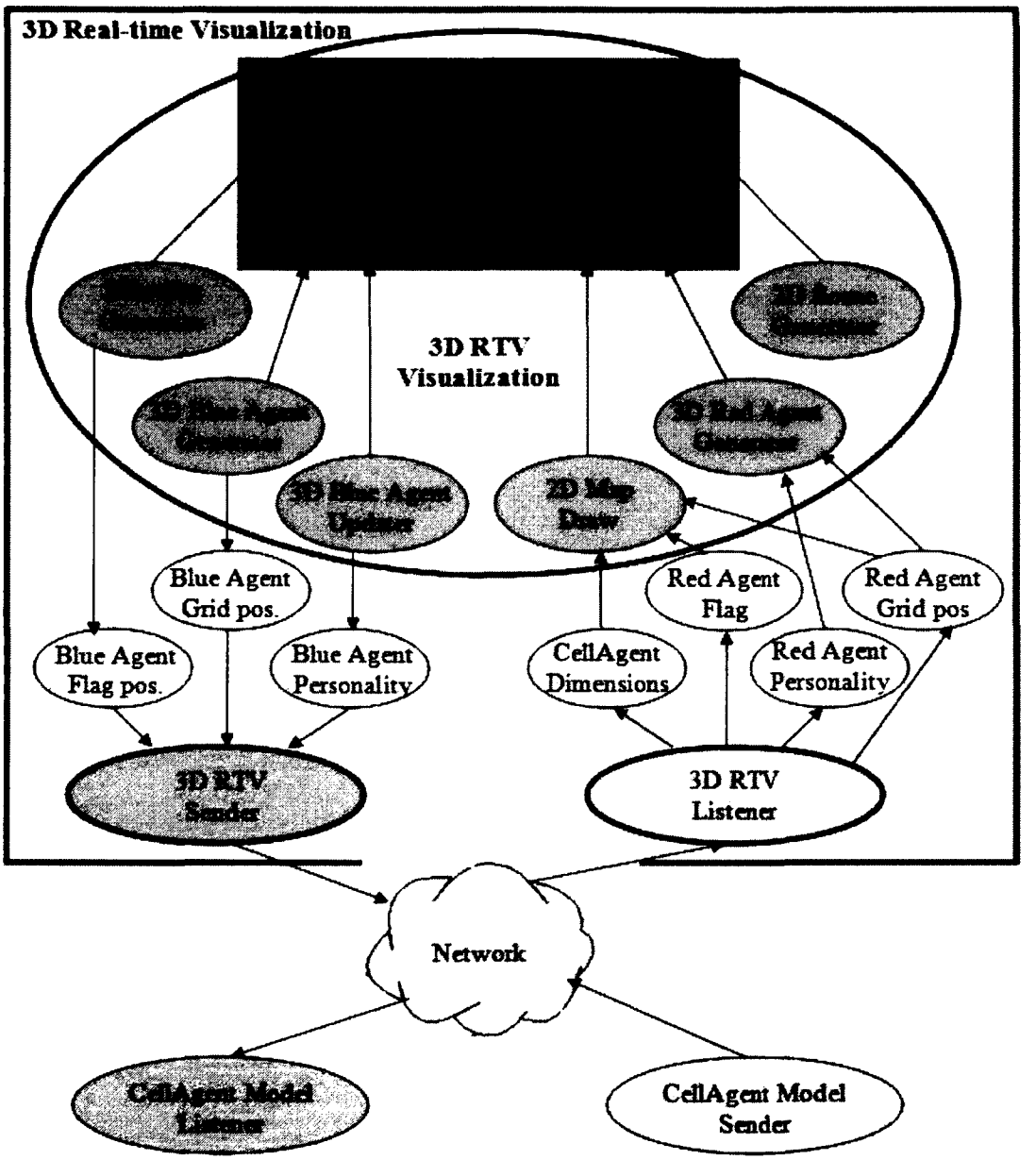


Figure 6.6: 3D Real-time Visualization Hierarchy

Listener and draws it in the 2D map channel.

- DrawCircle, which obtains the agents' positions and draws the red and blue circles according to the agents' type at the corresponding coordinates.
- DrawFlag, which draws a box for each flag according to its color at the corresponding coordinates.

4. *3D Blue Agent Generator*, which creates the blue agents (which include the 3D object model, personalities, and position). The *3D Blue Agent Generator* sends the position of the blue agents to the *RTV Sender*.
5. *3D Blue Agent Updater*, which calculates different personalities for the blue agents according to the received data for the red agents. It then sends the personalities of the blue agents to the *RTV Sender*.
6. *Blue Flag Generator*, which creates a 3D object for the blue agent's flag at the user-defined position, then sends the position to the *RTV Sender*.

The *RTV Sender* is a separate thread, spawned to send the blue agents' data. First, the *RTV Sender* sends the grid position of the blue agents' flag to the CellAgent sub-model running on the CD++ workstation. After that, it sends the blue agents' grid positions and their personality updates to the CellAgent sub-model while the model develops.

Finally, the 3D real-time visualization sub-model is capable of deploying different Openflight 3D terrain models and different cellular areas (dimensions and initial values) without changing the code of the visualization. As a result, we can add new components without the code of the visualization needing to be changed in the framework architecture, as discussed in Chapter 3. We can also visualize the output simulation on the 3D visualization scene, which improves and enhances decision-making. It also improves the training session by providing a 3D visualization scene and various environmental effects for the trainee on the RTV sub-system, which has different tactical

scenarios generated from the CellAgent sub-system.

6.5 VCELL Framework Scalability

The proposed implementation of VCELL has been tested with a variety of modeling scenarios, and several criteria have been applied for verification of the final implementation; for instance, a cell-agent model of 3 red agents on a 20X20 cell-space, each of them connected to an RTV engine and a visualization agent model of 3 blue agents. To ensure that the same scenario runs every time, the values coming from the cell-agent model were the same in all tests. All the cell-agent models follow the Cell-DEVS rules and the values were sent to the RTV engine successfully. This model is used to perform comprehensive performance tests in variable cell-space. The timing for the agent models varied in the different tests performed. The first test discussed in this section compared the average response time of the agents for different numbers of cell-spaces. The diagram in Fig. 6.7 shows the results of this test, for up to 60X60 cell-spaces.

The test was performed for 6 agents in the cell-agent model, increasing the cell-space dimension from 10X10 cells up to 60X60 cells with 6 layers. We used a Dell PC machine for the CellAgent model with the following configuration: Intel(R) Pentium(R) 4 CPU 3.20GHz, 512 MB memory, with operating system Linux Fedora Core 5. We used a Dell Precision T7500 workstation for the RTV model with the following configuration: Intel(R) Xeon(R) CPU W5580@3.2 GHz, 3GB, with operating system Windows 7 64-bit. As shown in the chart, by increasing the cell-space dimension, the average response time increases exponentially. This is due to the heavier workload produced in larger cell-space models and propagation of data in the model. The result demonstrates the integrity and persistence of the implementation in a small cell-space

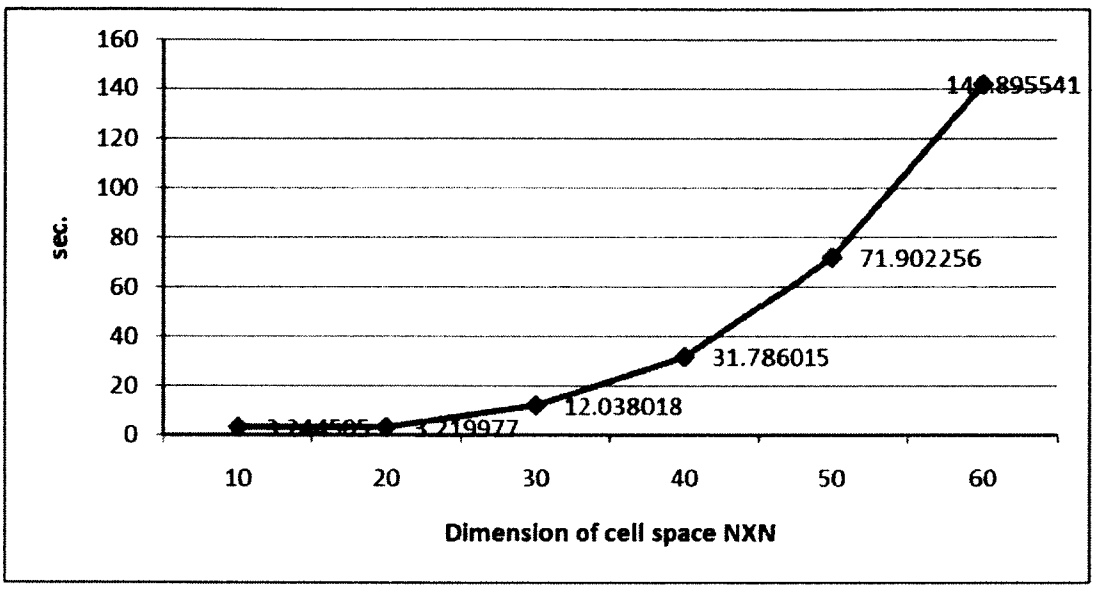


Figure 6.7: Number of Cell-Spaces vs. Average Response Time

dimension up to 40X40 cells with 6 layers, but it grows exponentially after that, which demonstrates that the system is non-scalable after a 50X50 cell-space with 6 layers. This is due to the heavier workload produced in larger cell-space (more than 15,000 cells) models and propagation of data in the model. This problem can be solved by upgrading the hardware resources of the CellAgent machine. In addition, we could use Parallel CD++, which minimizes the cell-space dimensions for 50X50 cells with 6 layers from 15,000 cells to 2,500 cells, which results in reducing the average response time. The exponential function of the diagram shows that the integrity of the functionality of VCELL for different numbers of cell spaces on the processor is good in the range of 60X60 cells.

To run the visualization sub-system in a real-time visual scene, the frame time should be less than 40 ms, as the visualization sub-system runs based on the number of frames per second (it should be more than 25 frames per second). However, the Cell-DEVS sub-system must update the agent state in less than 40 ms to get accurate

results. In all the tests, the system update time was less than 40 ms, which is acceptable to run in a smooth scene and without loss of data.

Fig. 6.8 shows the average response time of the model versus the number of agents on each cell-space. The test was performed by increasing the number of agents to check the system scalability on a 20X20 cell-space and 30X30 cell-space. For the 20X20 cell-space, we ran the system many times for 6 agents, 12 agents, 24 agents, 48 agents, and 96 agents. As for the 30X30 cell-space, we ran the system many times for 6 agents, 12 agents, 24 agents, 48 agents, and 96 agents. As is shown in the chart, by increasing the number of agents for the same cell-space, the processor utilization increases linearly. The result demonstrates the integrity and persistence of the implementation in a medium-load scenario. In addition, as the system gets busier, the response time also increases. The slope of the diagram for different configurations stays the same, showing the integrity of the functionality of VCELL for different levels of load on the processor.

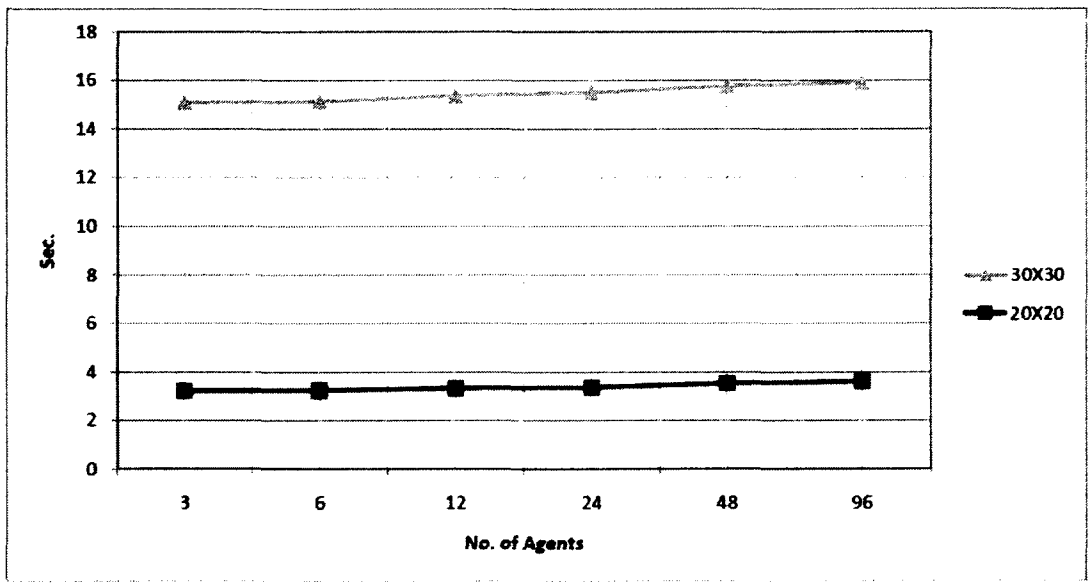


Figure 6.8: Number of Agents vs. Average Response Time

Based on the above, the performance of the system depends on the underlying hardware. As the number of cell spaces increases, the average response time also increases. This means that the tasks are executed later to their release time, when the system scales up. A simple solution might include upgrading the underlying hardware resources in order to solve the scalability problem. We might also use parallel CD++, which has a better performance and reduces the time and hardware resources compared with CD++.

6.6 Summary

We present a 3D real-time visual cellular agent model (VCELL) for collaborative cell agent simulation with 3D visualization in real time for different battlefield combat scenarios. This work is done using a 3D real-time visual engine and the CD++ simulator running Cell-DEVS models. The VCELL model is not only used for prediction, but also to improve the understanding and learning process in land combat. VCELL is designed to help analysts and trainers get maximum gain by interactively simulating the scenarios, validating the results, and training soldiers in an effective environment. This work incorporates two sub-models: a Cell-DEVS agent simulation and a 3D visualization agent. The two sub-models provide model reusability and interoperability, allowing for any of them to be integrated or replaced. The visual Cell-DEVS agent for land combat is a new way of enhancing the randomization movement problem of movement algorithms in agent-based simulation by using the 3D visualization agent, which obtains the real position of agents in combat. The Cell-DEVS agent simulation reduces the time taken to create or modify tactical scenarios in the 3D visualization simulation by using the Cell-DEVS formalism and the CD++, which includes an interpreter to write simple rules. These rules are transformed to be viewed in 3D by using 3D scenario generation. To implement the Cell-DEVS communication interface,

we modified the CD++ simulator core engine. Our model was implemented using robust software tools to make the real-time visual Cell-DEVS agent model more flexible. VCELL is open source, which means the source code can be read and modified, while other ABD toolkits are not open source. In addition, VCELL is more flexible than other ABDs toolkits in terms of the ability to add any new, desired features to a model, which allows user-specified algorithms to be used, while other ABD toolkits are restricted to a few algorithms. Moreover, VCELL is more adaptive than other existing ABD toolkits and includes good analysis capabilities. It also supports 3D visualization, which is not applicable to other selected ABD toolkits.

Chapter 7

Conclusions and Suggestions for Future Work

7.1 Conclusion and Summary

3D visualization has become important in simulation, as it presents a 3D graphical interface that is effective when used for training simulations. 3D visualization also gives decision-makers an overview of the problem at hand before they begin working with a real system. 3D visualization simulation can be combined with different applications-for instance, Building Information Modeling (BIM) software-in order to increase productivity in building design and construction. 3D visualization can also be combined with emergency simulations and battlefield simulation to enhance training preparation, allowing decision-makers to investigate different scenarios. As discussed in Chapter 1, the central theme of this thesis was to develop a framework that allows different simulation models to receive real-time external parameters so that they can be interactive, collaborative, and adaptive to the events in a simulation, by integrating DEVS, Cell-DEVS, and 3D visualization with different modeling techniques, which improves simulation interoperability at the software level and enhances the simulation results. Our objective was to develop a generic framework to be applied to simulation

environments and applications. We used BIM simulation, emergency simulation, and land combat simulation as case studies. This thesis presented a simulation adaptation and interoperability methods using DEVS, Cell-DEVS, and 3D visualization.

The VCELL framework combines Cell-DEVS models, DEVS-based controllers, and 3D visualization engines. This component-oriented approach provides model reusability and interoperability, allowing integration or replacement of any of the components. VCELL serves as a container to hold different software components without being specific to any implementation, which allows new components to be added without having to make modifications at the code level or in the framework architecture. Using VCELL to add various hardware devices allows hardware-in-the-loop, which achieves more accurate simulation results than when simulated hardware is used, and also improves training in different domains (such as in military training) by allowing the use of real equipment. Model-continuity from the early simulation stages to its finally embedding on the hardware speeds up the development process while increasing the reliability of the product and reducing risk and cost. VCELL can be used not only for real-time simulation, but also in virtual time, which allows for better and more accurate simulation results through comparing both the real-time and virtual-time output result and hence modifying the simulation rules. VCELL can use predefined inputs, actual inputs of the simulated system, or a combination of both predefined and actual inputs, which provide flexibility to modify the simulated system to obtain accurate results. VCELL also reduces the time taken for scenario development, modification, and validation by using the simple rules in Cell-DEVS simulation. The research first extended CD++ with a new component to hide CD++ internal implementation. This component consists of functions to support synchronization and adaptation and communicates using network messages. An interface was developed between DEVS, Cell-DEVS simulations, and 3D visualization, so that various instances of each domain can cooperate with each other during the

same simulation session. This means that simulation can be manipulated in real time. VCELL was applied successfully to the different application domains discussed above: BIM, emergency simulation, and land combat simulation.

First, we applied VCELL and integrated it with BIM models (this means that different simulations can be performed on BIM models to improve and enhance the BIM simulation results). The VCELL framework is adaptive and collaborative, as we can obtain the actual building parameters and carry out a simulation on the requested parameters, which results in improvement and enhancement of the BIM simulation. We implemented an Interactive Environment System (IES), which was described as an integration of Cell-DEVS formalism into BIM. By using the IES, we can improve the performance of BIM in different emergency situations, which is not covered in current BIM simulations. VCELL uses Cell-DEVS to simulate different emergency situations based on the external parameters received from the BIM models. The results of the Cell-DEVS simulation are sent back to the BIM model to be presented in a 3D visualization scene. The visualization sub-system integrates Cell-DEVS simulation and the BIM model to visualize the output result of the Cell-DEVS simulation on the BIM model. This facilitates the improvement and enhancement of the BIM model, as we can visualize the output results directly on the BIM model, which is more effective of 2D visualization of Cell-DEVS. It also facilitates making decisions and modifications. The feasibility of the proposed IES has been verified by using a diffusion limited aggregation (DLA) prototype.

When VCELL was applied to an emergency simulation, we implemented an integrated emergency management system based on the DEVS sub-system to develop new classes of cellular models for emergency response applications. The Cell-DEVS sub-system allows models to receive external information, and the simulation parameters can be updated at any time due to the continuous-time nature of the discrete-event specifications. The emergency simulation is based on the Cell-DEVS sub-system, and

the emergency management is based on the DEVS sub-system, which uses a robotic agent as an example to respond to the emergency simulation in real time. We also use the 3D visualization sub-system, which takes the results of the emergency simulation and the emergency management to be visualized in the 3D real-time visualization. This system offers a software framework to make the real-time emergency response system more flexible and more scalable. A robotic agent acting as a first responder is placed in a virtual environment generated from a Cell-DEVS emergency simulation. The controller of the robot is a DEVS-based emergency response model that interacts with the emergency simulation via messaging and is informed about the map of the area and the location of the incident (e.g., roadside bomb, fire, explosion, etc). Both the emergency simulation and the emergency response sub-systems run in real time and communicate with each other and with a 3D visualization engine. The purpose of the visualization system is to provide 3D scenes and to visually monitor the activities of the robotic first responder, which not only helps the modeler to enhance the emergency simulation models, due to the 3D virtual environment display of the real scene, but also helps decision-makers make the best choice. Therefore, the 3D visualization system will improve and enhance emergency simulation. Although the emergency model is a simulation, it is simple to replace it with more complex emergency simulation models or a real emergency database fed from real-world data. The proposed generic interface and message structure that enables the emergency simulation, the emergency response, and visualization sub-systems interchange data, enabling our system to simulate emergency management in real time under various conditions. Moreover, it can be integrated with stochastic optimization models that use the scenario results from the simulation to determine an optimal mix of emergency planning resources to dispatch to an emergency situation. The proposed system is intended not only to train emergency response personnel, but also to be used as a core real-time strategy and response system.

Finally, when we applied VCELL to a land combat simulation, we incorporated two sub-systems: a Cell-DEVS agent simulation and a 3D visualization agent, which is based on a 3D visualization sub-system. The visual Cell-DEVS agent for land combat is a new way of solving the randomization movement problem of movement algorithms in an agent-based simulation by using the 3D visualization agent, which obtains the real position of agents in combat. The Cell-DEVS agent simulation reduces the time taken to create or modify tactical scenarios in the 3D visualization simulation by using the Cell-DEVS formalism and CD++, which includes an interpreter to write simple rules. The results can be visualized in 3D using 3D scenario generation. This work was done using a 3D real-time visual engine and the CD++ simulator running Cell-DEVS models. The VCELL is used not only for prediction, but also to improve the understanding and learning process in land combat. VCELL is designed to help analysts and trainers get maximum gain by simulating the scenarios interactively, validating the results, and training the soldiers in an effective environment. To implement the Cell-DEVS communication interface, the CD++ simulator core engine was modified. VCELL was implemented using robust software tools to make the real-time visual Cell-DEVS agent model more flexible and more scalable.

7.2 Suggestions for Future Research

7.2.1 Interfacing DEVS and Visualization Models for Emergency Management

One of the future extension plans of this work is to integrate the Google Earth file free virtual reality web service with the visualization engine and inject the terrain data into the Cell-DEVS engine. Therefore, if there is an emergency anywhere in the world, such as the Fukushima Daiichi Nuclear Plant disaster, the model can read the information about the environment from the Google Earth file and simulate it using the Cell-DEVS formalism and then lead the robot.

7.2.2 Land Combat Simulation

VCELL is a multi-agent simulation combat system that facilitates the analysis and understanding of land combat and 3D real-time visualization simulation for tactics in land combat, as discussed in Chapter 6. We propose to modify and improve the 3D real-time visual cellular agent model (VCELL) by developing a hardware interface for different military equipment. This will make VCELL interactive, with real fighters in the 3D visualization agent simulation, and update the agent simulation data with a real situation.

The proposed agent, as shown in Fig. 7.1, could be composed of three main subsystems:

- The CellAgent sub-model, which will be implemented using a Cell-DEVS model running in real time
- The 3D real-time visual simulation (RTV) sub-model
- The human-in-the-loop training simulator.

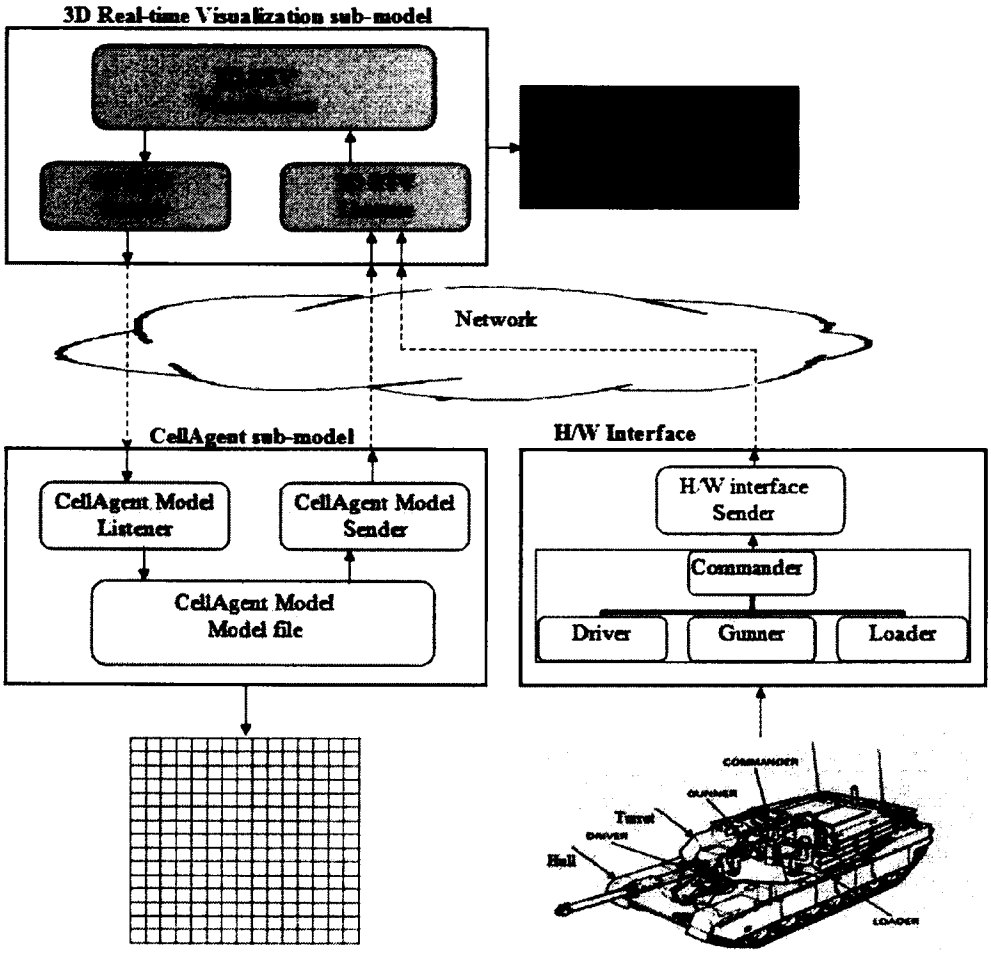


Figure 7.1: The Visual CELL-DEVS Agent (VCELL) Architecture

Each sub-model would run on a different machine, communicating by messages sent over a network. The CellAgent sub-model, presented in Section 6.3, the 3D real-time visualization sub-model, defined in Section 6.4, and the proposed human-in-the-loop training simulator would run in real time and communicate via messages transferred through a network infrastructure. The proposed human-in-the-loop training simulator will be designed, built, and used to train a tank crew in a virtual simulation environment. This will lend reality in training by using real equipment or a manufactured replica of it. As mentioned above, in our model, we will use a tank

as an example. The simulation environment of the tank will integrate four stations, which implement a driver's station, a gunner's station, a loader's station, and the commander's station of a simulated tank. The simulated tank systems will be built on the real body of a tank that includes real crew stations and its main weapon systems. This system will be integrated with the visual CELL-DEVS agent using RT-DEVS. RT-DEVS formalism is used for developing real-time embedded applications, which integrates simulation models with hardware components. The use of RT-DEVS reduces the development cycle and its costs by allowing DEVS models to be used for real-time and embedded applications with almost no modifications. It enhances the quality and the reliability of the final product and makes it portable on different hardware. Also we need to develop a model to predict the computational complexity as a function of the space being simulated, the number of active objects in that space and the real-time scale of the actions or change of state of the various active objects.

7.2.3 Traffic Systems Simulation

VCELL could be applied to other domains, such as understanding, examining, and solving traffic systems. We propose applying VCELL to a traffic system simulation to facilitate the analysis and understanding of various traffic systems and improve and enhance the output results. The proposed system could be composed of three main sub-systems:

- The DEVS sub-system, which will be implemented using a DEVS model
- The Cell-DEVS sub-system, which will be implemented using a Cell-DEVS model running in real time
- The 3D real-time visual simulation (RTV) sub-system.

We propose applying the DEVS sub-system to a traffic light system, which will allow various traffic light algorithms to be run to achieve better results. The Cell-DEVS sub-system can be assigned to the road flow and obstacles provided by the 2D map of the area of study, which allows for which allows for the best direction for traffic flow to be ascertained, and different traffic situations to be tested. The visualization engine can present the scene in 3D, so the flow of traffic can be observed in the area of study, which facilitates choosing the best traffic solution. With the visualization engine, we can use the system as a driving simulator, which allows a trainee to learn driving skills and rules, thereby improving the trainee's capabilities and saving lives. It can also be used when applying for a driver's license to confirm driver skills before the actual road driving test is taken, which saves time and lives.

List of References

[1] M. Macedonia. "Games soldiers play." *IEEE Spectrum* **39**(3), 32 –37 (2002).

[2] K. Stanney. *Handbook of virtual environments [electronic resource]: design, implementation, and applications.* Lawrence Erlbaum Associates (2002).

[3] P. Castonguay and G. Wainer. "Aircraft Evacuation DEVS Implementation & Visualization." In "Proceedings of Springsim 2009 (DEVS Symposium)," (2009).

[4] M. Prensky. *Digital Game-Based Learning.* Paragon House (2007).

[5] G. Lee, R. Sacks, and C. Eastman. "Specifying parametric building object behavior (BOB) for a building information modeling system." *Automation in Construction* **15**(6), 758–776 (2006).

[6] S. Mihindu and Y. Arayici. "Digital Construction through BIM Systems will Drive the Re-engineering of Construction Business Practices." In "International Conference Visualisation," pages 29–34 (2008).

[7] Bentley. "Bentley Architecture." `http://www.bentley.com/en-US/Products/Bentley+Architecture/` (Acessed Nov., 2011).

[8] Graphisoft. "ArchiCAD 14." `http://www.graphisoft.com/products/archicad/` (Acessed Nov., 2011).

[9] VectorWorks. "Vectorworks Architect." `http://www.nemetschek.net/architect/index.php` (Acessed Nov., 2011).

[10] AutoDesk. "Revit Architecture Building Information Modeling." `http://en.autodesk.ca/adsk/servlet/pc/index?id=14608943&siteID=9719649` (Acessed Nov., 2011).

[11] AutoDesk. "3ds Max - 3D Modeling, Animation, and Rendering software." `http://en.autodesk.ca/adsk/servlet/pc/index?id=14551042&siteID=9719649` (Acessed Nov., 2011).

[12] J. Kincaid, J. Donovan, and B. Pettitt. "Simulation Techniques for Training Emergency Response." *International Journal of Emergency Management* **1**(3), 238–246 (2003).

[13] A. Boukerche, M. Zhang, and R. Pazzi. "An adaptive virtual simulation and real time emergency response system." In "Proceedings of the IEEE international conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems," pages 360 –364. Hong Kong, China (2009).

[14] D. McGrath, A. Hunt, and M. Bates. "A simple distributed simulation architecture for emergency response exercises." In "Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real Time Applications," pages 221 – 226 (2005).

[15] K. Liu, X. Shen, A. El Saddik, A. Boukerche, and N. Georganas. "SimSITE: The HLA/RTI Based Emergency Preparedness and Response Training Simulation." In "Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real Time Applications," pages 59 –63 (2007).

[16] R. Cioarga, M. Micea, V. Cretu, and E. Petriu. "Movement in Collaborative Robotic Environments Based on the Fish Shoal Emergent Patterns." *Sensors & Transducers Journal* **5**, 18–36 (2009).

[17] R. Abielmona, E. Petriu, and T. Whalen. "Distributed intelligent sensor agent system for environment mapping." *Journal of Ambient Intelligence and Humanized Computing* **1**(2), 95–110 (2010).

[18] R. Abielmona, E. Petriu, M. Harb, and S. Wesolkowski. "Mission-Driven Robotic Intelligent Sensor Agents for Territorial Security." *IEEE Computational Intelligence Magazine* **6**(1), 55–67 (2011).

[19] B. Zeigler. *Theory of Modeling and Simulation.* New York: Wiley-Interscience (1976).

[20] B. Zeigler. *Multifacetted Modelling and Discrete Event Simulation.* Orlando: Academic Press (1984).

[21] B. Zeigler. *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.* Boston: Academic Press (1990).

[22] B. Zeigler and S. Vahie. "DEVS Formalism and Methodology: Unity of Conception/Diversity of Application." In "Proceedings of the 1993 Winter Simulation Conference," pages 573–579 (1993).

[23] G. Wainer and N. Giambiasi. "N-dimensional Cell-DEVS models." *Discrete Events Systems: Theory and Applications* **12**(1), 135–157 (2002).

[24] A. Khan, W. Venhola, G. Wainer, and M. Jemtrud. "On the use of CD++/Maya for visualization of discrete-event models." In "Proceedings of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation," (2005).

[25] A. Khan and G. Wainer. "Advanced Visualization of DEVS and Cell-DEVS Models in CD++/Maya." In "Proceedings of SISO Fall Interoperability Workshop," (2005).

[26] R. Hughes. "Simulation for design, test and evaluation, and training: reconciling the differences." In "Proceedings of the 22nd conference on Winter simulation," pages 231–236 (1990).

[27] P. GARRITY. "PC-Based Technology Invades Army Simulation." `http://www.metavr.com/downloads/PEOSTRI-PC-BasedTechInvadesArmySim2000.pdf` (Acessed Nov., 2011).

[28] J. Manojlovich, P. Prasithsangaree, S. Hughes, J. Chen, and M. Lewis. "Agent models I: UTSAF: a multi-agent-based framework for supporting military-based distributed interactive simulations in 3d virtual environments." In "Proceedings of the 35th conference on Winter simulation: driving innovation," pages 960–968 (2003).

[29] SONY. "Playstation 3." `http://us.playstation.com/playstation3/` (Acessed Nov., 2011).

[30] NINTENDO. "Nintendo Gamecube." `http://www.nintendo.com/3ds` (Acessed Nov., 2011).

[31] MICROSOFT. "Mech Warrior 4." `http://www.microsoft.com/games/mechwarrior4/` (Acessed Nov., 2011).

[32] D. Michael. *Serious games: games that educate, train, and inform.* Boston, MA: Thomson Course Technology PTR (2006).

[33] D. Gibson and Y. Baek. *Digital simulations for improving education: learning through artificial teaching environments.* Hershey, PA: Information Science Reference (2009).

[34] Noptel. "Noptel Shooter Training." `http://www2.noptel.fi/eng/nts/index.php` (Acessed Nov., 2011).

[35] L3. "F-22 Pilot Training Devices." `http://www.link.com/fa22ptd.html` (Acessed Nov., 2011).

[36] S. Bayarri, M. Fernandez, and M. Perez. "Virtual reality for driving simulation." *Communications of the ACM.* **39**, 72–76 (1996).

[37] M. Zyda, A. Mayberry, C. Wardynski, R. Shilling, and M. Davis. "The MOVES institute's America's army operations game." In "Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry," pages 219–220 (2003).

[38] NOVALOGIC. "Delta force 2." `http://www.novalogic.com/games.asp?GameKey=DF2` (Acessed Nov., 2011).

[39] Social-Impact. "Battle Command 2010." `http://www.socialimpactgames.com/modules.php?op=modload&name=News&file=index&catid=9&topic=&allstories=1` (Acessed Nov., 2011).

[40] G. Fong. "Adapting cots games for military simulation." In "Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry," pages 269–272 (2004).

[41] C. Grigore, C.and Philippe. *Virtual Reality Technology.* Wiley-IEEE Press (2003).

[42] J. Vince. *Introduction to Virtual Reality.* Springer Verlag (2004).

[43] F. Brooks. "What's Real About Virtual Reality?" In "Proceedings of the IEEE Computer Graphics and Applications," pages 2–3 (1999).

[44] R. SCHROEDER, A. STEED, A. AXELSSON, I. HELDAL, A. ABELIN, J. WIDESTROEM, A. NILSSON, and M. SLATER. "Collaborating in networked immersive spaces: as good as being there together?" *Computers and Graphics* **25**(5), 781–788 (2001).

[45] D. Bowman, J. Gabbard, and D. Hix. "A study of influential factors on effective closely-coupled collaboration based on single user perceptions." *Presence: Teleoperators and Virtual Environments* **11**(4), 404–424 (2002).

[46] D. Roberts, R. Wolef, and O. Otto. "Constructing a gazebo: Supporting team work in a tightly coupled, distributed task in virtual reality." *Presence: Teleoperators and Virtual Environments* **12**(6), 644–668 (2003).

[47] AutoDesk. "Autodesk Maya." `http://en.autodesk.ca/adsk/servlet/pc/index?id=14557791&siteID=9719649`, (Acessed Nov., 2011).

[48] MilkShape3D. `http://chumbalum.swissquake.ch/index.html` (Acessed Nov., 2011).

[49] PLANIT3D. `http://www.planit3d.com/source/index.htm` (Acessed Nov., 2011).

[50] AC3D. `http://www.inivis.com/` (Acessed Nov., 2011).

[51] Blender. `http://www.blender.org/` (Acessed Nov., 2011).

[52] I. Palmer. *Essential Java 3D: Developing 3D Graphics Applications in Java.* Springer (2001).

[53] J. Hartman and J. Wernecke. *The VRML 2.0 handbook: building moving worlds on the web.* Addison-Wesley (1996).

[54] OpenGL. `http://www.sgi.com/products/software/opengl/` (Acessed Nov., 2011).

[55] A. Walsh and M. Bourges-Sevenier. *Core Web 3D.* Prentice Hall PTR (2000).

[56] M. Griffiths. "The educational benefits of videogames." *Education and Health* **20**(3), 47 - 51 (2002).

[57] T. C. 25-20. *A Leader's Guide To After-Action Reviews.* Department of the Army, Washington, DC. (1993).

[58] C. on Modeling, Simulation, and Games. *The Rise of Games and High Performance Computing for Modeling and Simulation.* The National Academies Press (2010).

[59] Y. Adachi, T. Kumano, and K. Ogino. "Intermediate Representation for Stiff Virtual Objects." In "Proceedings of the 1995 Annual International Symposium on Virtual Reality," pages 203 –210 (1996).

[60] R. Pausch, T. Burnette, A. Capehart, M. Conway, D. Cosgrove, R. DeLine, J. Durbin, R. Gossweiler, S. Koga, and J. White. "Alice: Rapid prototyping for virtual reality." *IEEE Computer Graphics and Applications* **15**(3), 8–11 (1995).

[61] Mechdyne. "Cavelib." `http://www.mechdyne.com/cavelib.aspx` (Acessed Nov., 2011).

[62] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. "VR Juggler: A Virtual Platform for Virtual Reality Application Development." In "Proceedings of the 2001 IEEE Virtual Reality," pages 89 –96 (2001).

[63] J. Kelso, L. Arsenault, S. Satterfield, and R. Kriz. "DIVERSE: A Framework for Building Extensible and Reconfigurable Device Independent Virtual Environments." In "Proceedings of the 2002 IEEE Virtual Reality," pages 183 –190 (2002).

[64] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavári, L. M. Encarnaçäo, M. Gervautz, and W. Purgathofer. "The studierstube augmented reality project." *Presence: Teleoperators and Virtual Environments* **11**(1), 33–54 (2002).

[65] C. Shaw, M. Green, J. Liang, and Y. Sun. "Decoupled simulation in virtual reality with the mr toolkit." *ACM Transactions on Information Systems (TOIS)* **11**(3), 287–317 (1993).

[66] G. Wainer and N. Giambiasi. "Timed Cell-DEVS: modelling and simulation of cell spaces." In "Discrete event modelng and simulation technologies," chapter 10, pages 187–214. Springer-Verlag New York, Inc. (2001).

[67] G. Klir. *Trends in general systems theory.* New York: Wiley-Interscience (1972).

[68] L. Zadeh and C. Desoer. *Linear system theory; the state space approach.* New York, McGraw-Hill (1963).

[69] G. Wainer. *Discrete-Event Modeling and Simulation: a Practitioner approach.* Taylor and Francis (2009).

[70] B. Zeigler, H. Praehofer, and K. T. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press (2000).

[71] Y. Labiche and G. Wainer. "Towards the Verification and Validation of DEVS Models." In "Proceedings of the 1st Open International Conference on Modeling & Simulation," Clermont-Ferrand, France (2005).

[72] B. Zeigler, Y. Moon, D. Kim, and J. Kim. "DEVS-C++: A High Performance Modelling and Simulation Environment." In "Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences," volume 1, pages 350 – 359 (1996).

[73] H. Sarjoughian and B. Zeigler. "Devs and hla: Complementary paradigms for m&s?" *Transactions of the Society for Computer Simulation* **17**(4), 167–197 (2000).

[74] B. Zeigler. "Implementation of the DEVS Formalism over the HLA/RTI: Problems and Solutions." In "Simulation Ineroperability Workshop," (1999).

[75] H. Sarjoughian and B. Zeigler. "DEVSJAVA: Basis for a DEVS-based collaborative M&S." In "Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation," (1998, month=, volume=, number=, pages=29-36,).

[76] F. J. and B. P. "Jdevs: An implementation of a devs based formal framework for environmental modelling." *Environmental Modelling and Software* **19**(3), 261–274 (2004).

[77] G. Wainer. "CD++: a toolkit to define discrete-event models." *Software, Practice and Experience* **32**(3), 1261–1306 (2002).

[78] T. Chandrupatla and A. Belegundu. *Introduction to finite elements in engineering.* Upper Saddle River, N.J. : Prentice Hall (1997).

[79] S. Wolfram. *Theory and applications of cellular automata(Advances Series on Complex Systems), Volume 1.* World Scientific (1986).

[80] S. Wolfram. *A new kind of science.* Champaign, IL : Wolfram Media (2002).

[81] T. Toffoli and N. Margolus. *Cellular automata machines : a new environment for modeling.* Cambridge, Mass. : MIT Press (1987).

[82] J. V. Neumann. *Theory of Self-Reproducing Automata.* University of Illinois Press (1966).

[83] J. Ameghino, A. Troccoli, and G. Wainer. "Models and simulation of complex physical systems using Cell-DEVS." In "Proceedings of the 33rd SCS Summer Computer Simulation," pages 266–273 (2001).

[84] G. Wainer. "Modeling and simulation of complex systems with Cell-DEVS." In "Proceedings of the Winter Simulation Conference," volume 1 (2004).

[85] G. Wainer, S. Jafer, B. Al-Aubidy, A. Dias, R. Bain, M. Dumontier, and J. Cheetham. "Advanced DEVS models with application to biology and medicine." In "Proceedings of Artificial Intelligence, Simulation and Planning in High Autonomy Systems," Buenos Ares, Argentina (2007).

[86] B. Nayfeh. "Cellular automata for solving mazes." *Dr. Dobb's Journal* **18**, 32–38 (1993).

[87] K. Lam and G. Wainer. "Modeling of maze-solving problems using Cell-DEVS." In "Proceedings of the 2003 SCS Summer Computer Simulation Conference," Montreal, QC., Canada (2003).

[88] K. Kidisyuk and G. Wainer. "CD++: A graphical viewer for DEVS models." Technical report, SCE-017, Ottawa, ON., Canada (2007).

[89] E. Poliakov, G. Wainer, J. Hayes, and M. Jemtrud. "Modeling Space-Shaped Defense Applications with Cell-DEVS." In "Proceedings of AIS, Artificial Intelligence, Simulation and Planning," Buenos Aires, Argentina (2007).

[90] C. Zhang, T. Zayed, A. Hammad, and G. Wainer. "Cell-based representation and analysis of spatial resources in construction simulation." *Elsevier Journal of Automation in Construction* **16**(4), 436–448 (2007).

[91] "Advanced Visualization of DEVS and Cell-DEVS Models in CD[++/Maya, author = Khan, A. and Wainer, G. , booktitle = Proceedings of SISO Fall Interoperability Workshop, year = 2005, address = San Diego, CA. U.S.A,."

[92] A. Khan, W. Venhola, G. Wainer, and M. Jemtrud. "On the use of CD++/Maya for visualization of discrete-event models." In "Proceedings of IMACS World Congress on Scientific Computation, Applied Mathematics and Simulation," Paris, France (2005).

[93] R. Madhoun and G. Wainer. "Modeling Space-Shaped Defense Applications with Cell-DEVS." In "Proceedings of SISO Fall Interoperability Workshop," San Diego, CA. U.S.A (2005).

[94] R. Madhoun and G. Wainer. "Creating Spatially-Shaped Defense Models Using DEVS and Cell-DEVS." *SCS Journal of Defense Modeling and Simulation* **2**(3), 121–143 (2005).

[95] H. Nam-Hyuk, M. Kyung-Min, K. Ju-Hyung, L. Yoon-Sun, and K. Jae-Jun. "A Study on Application of BIM (Building Information Modeling) to Pre-design in Construction Project." In "Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on," volume 1, pages 42–49 (2008).

[96] H. Brad. *BIM and Construction Management: Proven Tools, Methods, and Workflows.* Wiley (2009).

[97] ACS. "Architectural services & steel detailing." `http://www.acscad.ie/3d_visualisation.html` (Acessed Nov., 2011).

[98] IAI/IFC. "International Alliance for Interoperability, Industrial Foundation Classes." `http://www.iai-tech.org/` (Acessed Nov., 2011).

[99] I. 10303-1. "Industrial automation systems and integration - product data representation and exchange -." Part 1: Overview and fundamental principles (1994).

[100] C. Eastman. *Building Product Models: Computer Environments Supporting Design and Construction.* London: CRC Press (1999).

[101] C. Eastman, P. Teicholz, R. Sacks, and K. Liston. *BIM Handbook.* John Wiley & Sons (2008).

[102] EQUA. "Equa simulation, simulation software, simulation consultancy." `http://www.equa.se/` (Acessed Nov., 2011).

[103] DesignBuilder. "Building design, simulation and visualization." `http://www.designbuilder.co.uk/` (Acessed Nov., 2011).

[104] P. Sanguinetti, C. Eastman, and G. Augenbroe. "COURTHOUSE ENERGY EVALUATION: BIM AND SIMULATION MODEL INTEROPERABILITY IN CONCEPT DESIGN." In "Eleventh International IBPSA Conference," pages 1922 – 1929. Glasgow, Scotland (2009).

[105] J. Wang, J. Li, and X. Chen. "Parametric Design Based on Building Information Modeling for Sustainable Buildings." In "Challenges in Environmental

Science and Computer Engineering (CESCE), 2010 International Conference on," volume 2, pages 236 –239 (2010).

[106] S. Jain and C. McLean. "A framework for modeling and simulation for emergency response." In "Proceedings of the Winter Simulation Conference," volume 1, pages 1068 - 1076 (2003).

[107] S. Jain and C. McLean. "Integrated simulation and gaming architecture for incident management training." In "Proceedings of the Winter Simulation Conference," (2005).

[108] R. Gonzalez. "Analysis and design of a multi-agent system for simulating a crisis response organization." In "Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation," Amsterdam, Netherlands (2009).

[109] F. Lanchester. *Aircraft in Warfare: The Dawn of the Fourth arm.* Constable and Company Limited, London. ISBN 9781409776192 (1916).

[110] A. Ilachinski. "Irreducible semi-autonomous adaptive combat (isaac): An artificial life approach to land combat." Technical Report CRM 97-61, Center for Naval Analyses, Alexandria, VA (1997).

[111] A. Ilachinski. "Irreducible semi-autonomous adaptive combat (isaac): An artificial life approach to land combat." *Military Operations Research* **5**, 29–46 (2000).

[112] M. Barlow and A. Easton. "Crocadile: An open, extensible agent-based distillation engine." *Information & Security* **8**(1), 17–51 (2002).

[113] A. Ilachinski. *Enhanced Isaac Neural Simulation Toolkit (einstein), an Artificial-Life Laboratory for Exploring Self-Organized Emergence in Land Combat.* Center for Naval Analyses, Beta-Test Users Guide CIM 610.10. (1999).

[114] M. Lauren. "Modeling combat using fractals and the satistics of scaling ssystems." *Military Operations Research* **5**(3), 47–58 (2000).

[115] D. Alberts and T. Czerwinski. *Complexity, Global Politics, and National Security*, chapter 9, pages 99–111. National Defense University, Washington, D.C. (1997).

[116] R. Brooks and L. Steels. *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*, chapter 2, pages 25–81. Lawrence Erlbaum Associates, Hillsdale, NJ (1995).

[117] A. Yang, H. Abbass, and R. Sarker. "Characterizing warfare in red teaming." *IEEE Transactions on ISystems, Man, and Cybernetics, Part B: Cybernetics* **36**(2), 268 –285 (2006).

[118] M. Lauren and R. Stephen. "Map-aware non-uniform automata - a new zealand approach to scenario modelling." *Battlefield Technology* **5**(1), 27–31 (2002).

[119] G. White. "The mathematical agent a complex adaptive system representation in bactowars." In "First workshop on complex adaptive systems for defence," (2004).

[120] A. Gill. "Improvement to the movement algorithm in the mana agent-based distillation." *Battlefield Technology* **7**(2), 19–22 (2004).

[121] A. Yang, H. Abbass, and R. Sarker. "Wisdom-ii: A network centric model for warfare." In "Knowledge-Based Intelligent Information and Engineering Systems," pages 173–173. Springer Berlin / Heidelberg (2005).

[122] A. Gill and D. Grieger. "Validation of agent based distillation movement algorithms." Technical report, Defence Science and Technology Organization, Australia. DSTO-TN-0476 (2003).

[123] D. Grieger. "Comparison of two alternative movement algorithms for agent based distillations." Technical Note DSTO-TN-0777, Defence Science and Technology Organisation Edinburgh (Australia) Land Operations Div (2007).

[124] W. Ding, C. Lin, L. Chechiu, X. Wu, and G. Wainer. "Definition of Cell-DEVS Models for Complex Diffusion Systems." In "Proceedings of the SCS Summer Computer Simulation Conference," (2005).

[125] M. Moallemi, S. Jafer, A. Sayed Ahmed, and G. Wainer. "Interfacing DEVS and Visualization Models for Emergency Management." In "Proceedings of 2011 Spring Simulation Conference (SpringSim11)," (2011).

[126] G. Christen, A. Dobniewski, and G. Wainer. "Modeling State-Based DEVS Models in CD++." In "Proceedings of MGA, Advanced Simulation Technologies Conference," (2004).

[127] M. Moallemi and G. Wainer. "Designing an Interface for Real-Time and Embedded DEVS." In "Proceedings of 2010 Spring Simulation Conference (SpringSim10), DEVS Symposium," pages 154–161 (2010).

[128] PRESAGIS. "Vega prime." `http://www.presagis.com/products_services/products/ms/visualization/vega_prime/` (Acessed Nov., 2011).

[129] A. Sayed Ahmed, M. Moallemi, , and G. Wainer. "VCELL: A 3D Real-time Visual Simulation in Support of Combat." In "Proceedings of 2011 Summer Simulation Multiconference (SummerSim11)," (2011).