EVACART: Evacuation Assessment and Real-Time Decision Guiding Tool

NAN YANG

EMBRY-RIDDLE AERONAUTICAL UNIVERSITY

GRADUATE RESEARCH PROJECT

DATE 10/20/2015

VERSION 1.6

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>Nan Yang</td>
<td>01/08/2015</td>
</tr>
<tr>
<td>GRP Advisor</td>
<td>Dr. Shafagh Jafer</td>
<td>11/09/2015</td>
</tr>
<tr>
<td>MSE Coordinator / Department Chair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/15/15</td>
<td>1.1</td>
<td>Initial document</td>
</tr>
<tr>
<td>10/30/15</td>
<td>2.1</td>
<td>First edition based on Dr. Jafer’s comments</td>
</tr>
<tr>
<td>11/05/15</td>
<td>3.1</td>
<td>Second edition based on Dr. Jafer’s comments</td>
</tr>
<tr>
<td>11/09/15</td>
<td>3.2</td>
<td>Third edition based on Dr. Jafer’s comments</td>
</tr>
<tr>
<td>21/12/15</td>
<td>3.3</td>
<td>Fourth edition based on Dr. Jafer’s comments</td>
</tr>
</tbody>
</table>
# Table of Contents

Table of Figure ........................................................................................................ 5  
Acknowledgement .................................................................................................... 6  
1. Abstract ............................................................................................................. 7  
2. Introduction ......................................................................................................... 7  
3. Literature Survey ............................................................................................... 8  
   3.1 Discrete Event Modelling and Simulation: A Practitioner’s Approach .......... 8  
   3.2 Crowd Evacuation Modelling and Simulation Framework for Emergencies .. 9  
4. Methodology ....................................................................................................... 9  
   4.1 Methodology Overview ............................................................................... 9  
      4.1.1 DEVS ................................................................................................. 9  
      4.1.2 CELL-DEVS Formalism .................................................................. 10  
      4.1.3 CD++ TOOLKIT .............................................................................. 10  
      4.1.4 Evacuation Models ......................................................................... 11  
   4.2 Application Overview ............................................................................... 11  
      4.2.1 Objectives ....................................................................................... 12  
      4.2.2 Required Resources ...................................................................... 12  
      4.2.3 File Generated ................................................................................ 14  
      4.2.4 User Interface ................................................................................. 19  
      4.2.5 Demo Video ..................................................................................... 24  
   4.3 Software Architecture .................................................................................. 24  
      4.3.1 Class Diagram .................................................................................. 24  
   4.4 System Architecture ..................................................................................... 25  
   4.5 Design Considerations .............................................................................. 27  
      4.5.1 Application Type ............................................................................. 27  
   4.6 Demo Video .................................................................................................. 27  
   4.7 DEVS Unit: CD++ Unit Test Plug-in Project ............................................. 27  
      4.7.1 Unit Test Simulation Window ............................................................ 27  
      4.7.2 Unit Test Result View ...................................................................... 28  
      4.7.3 User Guide ...................................................................................... 29  
5. Conclusion ........................................................................................................ 33  
   5.1 Summary ...................................................................................................... 33
Future Work .......................................................... 33

References........................................................................ 34
Table of Figure

**Figure 1** Cell Neighborhood 1 (Wainer G. A., 2009) ................................................................. 10
**Figure 2** Cell Neighborhood 2 (Lawler R. J., 2014) ................................................................. 10
**Figure 3** Cygwin set up 1 ........................................................................................................ 13
**Figure 4** Cygwin set up 2 ........................................................................................................ 13
**Figure 5** Cygwin set up 3 ........................................................................................................ 14
**Figure 6** Cygwin set up 4 ........................................................................................................ 14
**Figure 7** Bat file example ....................................................................................................... 15
**Figure 8** Log file example ..................................................................................................... 16
**Figure 9** MaFile example 1 ................................................................................................... 17
**Figure 10** Ma file example 2 ................................................................................................ 18
**Figure 11** Val file example ..................................................................................................... 19
**Figure 12** UI-Main View ........................................................................................................ 20
**Figure 13** UI-Main View Simulation Result ........................................................................... 20
**Figure 14** Edit Environment View 1 ...................................................................................... 21
**Figure 15** Edit Environment View 2 ...................................................................................... 21
**Figure 16** UI-New Environment View 1 ................................................................................ 22
**Figure 17** UI-New Environment View 2 ................................................................................ 22
**Figure 18** Compare Model View (Default) ........................................................................... 23
**Figure 19** Compare Model View Result 1 .............................................................................. 23
**Figure 20** Compare Model View Result 2 .............................................................................. 23
**Figure 21** Animation View .................................................................................................... 24
**Figure 22** UML class diagram .............................................................................................. 25
**Figure 23** System Architecture ............................................................................................. 26
**Figure 24** Unit Test Simulation Project window ..................................................................... 28
**Figure 25** Unit Test Result View ........................................................................................... 29
**Figure 26** Input information .................................................................................................. 29
**Figure 27** Output Information ................................................................................................ 29
**Figure 28** Instruction on how to open the plug-in ................................................................. 30
**Figure 29** Plug-in User Interface 1 ......................................................................................... 30
**Figure 30** Plug-in User Interface 2 ........................................................................................ 31
**Figure 31** Plug-in User Interface 3 ........................................................................................ 32
**Figure 32** Plug-in User Interface Result View ......................................................................... 33
Acknowledgement

Since this project is built on top of the CD++ Toolkit and Lawler’s evacuation models, credits are given to Wainer and Lawler for their work. Grateful for all the help from Dr. Jafer.
1. Abstract

It is well documented that in an evacuation emergency the well-prepared and trained evacuators have higher chance to survive than those not trained. The objective of this project is to provide an application for emergency evacuation training and assimilating information about an emergency evacuation. In this project, EVACART, Evacuation Assessment and Real-Time Decision Guider Tool, has been developed on top of an open-source eclipse-based modeling and simulation environment. Based on formal modeling and simulation approaches, EVACART provides an environment to easily define an area of interest, apply various egress strategies, run simulations, conduct statistical analysis, and provide decision guidance on optimal evacuation methods. During this project, the entities on top of which the project was built and the application details have been reviewed, including its software architecture, design considerations, and the general functionalities. Future research can be conducted to enhance and expend the functionalities of EVACART.

**Keywords:** Discrete-Event Modeling and Simulation, evacuation, egress, eclipse

2. Introduction

A community could have a special event (e.g. seasonal festivals) during which a natural event (tornado) or intentional incident (explosion) occurs. The impact of the incident would influence how people react and how the movement of the people must be managed to best protect them while expediting the activity.

For companies, government departments, and any other organizations that manage a physical area, lack of preparations for emergency might result in a failure in protecting the physical safety of its staff and customers, as well as its valuable assets. In this way, efficient emergency evacuation plans and real-time decision making capabilities are necessary for people to safely evacuate from crowded environments.

The objective of this project is to develop an emergency evacuation training and decision support tool: the EVACART, which is built for emergency evacuation training and assimilating information about an emergency evacuation in real-time, allowing an emergency operation center to evaluate alternative courses of action in real-time prior to making a decision.

As specified before, EVACART is an evacuation simulation application, which is based on the CD++ Toolkit (Wainer G., 2002). The evacuation simulation and 2D visualization functionalities of EVACART have been developed base on the CD++ Toolkit. EVACART enables the user to perform evacuation simulation for a user-defined environment under a user-selected evacuation strategy. Moreover, the application enables the user to compare the efficiency of multiple strategies by comparing the simulation time (evacuation time) of various simulation
models under different evacuation strategy for the same environment. In this way, the user will be able to choose a more efficient strategy for any specific environment. The evacuation strategies embedded in the application is from the research project of Lawler Ryan (Lawler R. J., 2014).

As a result of this project, the application EVACART have been developed with a side project named CD++ unit testing tool. A detailed user manual is included to provide detail guidelines on how the two applications can be used. The materials required by the application will also be stated, so that the reader will be able to build a platform on which the application can successful perform its designed functionalities.

In section III, technologies, that have been used while developing this project, will be reviewed. Also, this section will provide some detail application information on both users’ perspective and developers’ perspective. In section IV and V, the research results and conclusions will be presented.

3. Literature Survey
In order to gain sufficient understanding of the project scope, a literature review of previous related researches has been conducted. For this project, the two prime research topics, which will be reviewed later in this section, are DEVS simulation and the crowd evacuation strategies. In this section, related books and research papers will be reviewed.

3.1 Discrete Event Modelling and Simulation: A Practitioner’s Approach
This text book (Wainer G. A., 2009) is the guide line on how to develop practical modelling and simulation models, using DEVS as the foundational modelling and simulation formalism.

This book have been divided into the following four sections: “Concepts”, “Building Simulation Models: The CD++ Toolkit”, “Applications”, and “Simulation and Visualization”.

The “Concept” section provides a general perspective on discrete-event modelling and simulation. Also, this section introduces the DEVS and Cell-DEVS formalisms. The “Building Simulation Models: The CD++ Toolkit” section introduces the basic concept of creating a model using CD++. According to the book (Wainer G. A., 2009), the “Applications” section starts with a chapter on biology and medicine and then moves to defense and emergency planning, after which it discusses architecture and construction, environmental sciences, physics and chemistry, artificial systems, and urban traffic. The “Simulation and Visualization” section introduces how to create a simulation software for DEVS models. And then, it presents the process of creating 3D Visualization environments associated with CD++ Toolkit.
3.2 Crowd Evacuation Modelling and Simulation Framework for Emergencies
This paper (Lawler & Jafer, 2015) aims to create a modelling and simulation framework for human behaviour in emergency situations that can be easily accessed and integrated into future research. According to the paper (Lawler & Jafer, 2015), in Lawler’s research, the existing literature on human behaviour have been investigated. As a result, Lawler’s research developed a set of movements and behaviours that could then be translated into models and simulations using the Cell-DEVS formalism.

4. Methodology
This section reviews the technologies used when developing the application, as well as guidelines on how the application can be used. This section will also provide details on the architecture perspective and talk about the design considerations.

4.1 Methodology Overview
This section introduces the major mathematical methods, simulation formalisms, models, and tools that have been used in the application developing process.

4.1.1 DEVS
According to Wainer (Wainer G. A., 2009), Discrete-event systems specification (DEVS) M&S theory is one of the discrete-event modeling and simulation approaches that is highly based on systems theory concepts. DEVS was designed for modeling and simulating discrete-event dynamic systems (DEDS); thus, it defines a way to specify systems whose states change either upon the reception of an input event or due to the expiration of a time delay. (Wainer G. A., 2009)

A basic DEVS model can be expressed in the following method:

- \( M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle \) (Wainer G. A., 2009)

Where:

- \( X = \{(p,v) | p \in \text{IPorts}, v \in Xp \} \) is the set of input events, where IPorts represents the set of input ports and Xp represents the set of values for the input ports;
- \( Y = \{(p,v) | p \in \text{OPorts}, v \in Yp \} \) is the set of output events, where OPorts represents the set of output ports and Yp represents the set of values for the output ports;
- \( S \) is the set of sequential states;
- \( \delta_{ext} : Q \times X \rightarrow S \) is the external state transition function, with \( Q = \{(s,e)/s \in S, e \in [0, ta(s)]\} \) and e is the elapsed time since the last state transition;
- \( \delta_{int} : S \rightarrow S \) is the internal state transition function;
- \( \lambda : S \rightarrow Y \) is the output function;
- \( ta : S \rightarrow R^0 \cup \infty \) is the time advance function.
In this project, the evacuation models are based on DEVS.

4.1.2 CELL-DEVS Formalism

Cellular Automata (CA) is one of the formalisms that are designed to capture system behaviors based on cell movement. CELL-DEVS is the combination of CA and DEVS, in which each cell block is a unique atomic model (a DEVS model), and every cell share the same set of previously defined simulation rules.

In a CELL-DEVS model, the cell map is defined as a two-dimensional grid (as shown in below figure Cell Neighborhood 1), each cell computes its future state based on its value, the values of the neighborhood according to the previously define rules.

![Cell neighborhood](image)

*Figure 1 Cell Neighborhood 1 (Wainer G. A., 2009)*

In CELL-DEVS, neighborhoods of a cell block is defined as all the cell blocks that locates directly by the cell itself as shown in figure Cell Neighborhood 1 and Cell Neighborhood 2.

![Cell neighborhood](image)

*Figure 2 Cell Neighborhood 2 (Lawler R. J., 2014)*

In this project, CELL-DEVS formalism is used to present the evacuation environment and used to capture the human beings’ movement. Human beings, walls, blocks are represented by different values in each cell block. Rules, which guide the CELL-DEVS simulation, are based on previous research (Crowd Evacuation Modeling and Simulation Framework for Emergencies).

4.1.3 CD++ TOOLKIT

To accomplish above objectives, this application was built on top of an existing open-source discrete-event modeling and simulation toolkit named CD++ (Lawler R. J., 2014). The CD++ toolkit is an eclipse-based toolkit designed on top of the DEVS theory and its extensions.
In this project, the source code of CD++ have been referred and directly used to the simulation engine (CD++) in the Java application EVACART, which turns out to be one of the major challenges of this project.

4.1.4 Evacuation Models
To accomplish major objectives of this project, appropriate evacuation strategies shall be used to generate high efficiency evacuation plans. In this project, the evacuation models used are designed by Lawler (Lawler R. J., 2014) in his research project Crowd Evacuation Modeling and Simulation Framework for Emergencies.

Among all Lawler’s evacuation models, the standard movement model, the directional movement model, the follow the herd movement model, the patrol movement model, and the panic movement model have been integrated into the application. Proper modification against evacuation the models have been made to fit in the java environment. A brief introduction of each evacuation models will be given below according to Lawler’s research (Lawler R. J., 2014):

4.1.4.1 Standard Movement Model
In a standard movement model, evacuators will choose the optimistic direction as the simulation start.

4.1.4.2 Directional Movement Model
In a directional movement model, evacuators will follow the directions provided by directional cues such as signage and finally get to the exit.

4.1.4.3 Follow the Herd Movement Model
In a follow the herd movement model, evacuators will follow a group of people. In this situation, an evacuator assumes the group of people knows the correct direction to evacuate from the building.

4.1.4.4 Patrol Movement Model
In a patrol movement model, a person of authority will follow an establish path, patrolling the building to determine whether or not everyone has been evacuated. (Lawler R. J., 2014)

4.1.4.5 Panic Movement Model
In a panic movement model, evacuators will perform a relatively random movement out of fear. There are still chances they are able to get to the exit, but the chances turns out to be relatively low.

4.2 Application Overview
In this section, an overview of the application EVACART will be presented, including its objectives, the required resources, files it generates during the simulation process, and a brief user interface introduction.
4.2.1 Objectives
This project is developed to perform simulations for emergency evacuation on real time demand. Using this application, users shall be able to load existing area maps, creating their own maps, and editing existing maps for simulation purpose. Also, before simulation, users will be able to select from the list of evacuation strategies and set up a simulation end time. After simulations, with the help of EVACART the user will be able to compare evacuation efficiency between each evacuation strategy, and visualize the evacuation process in a 2D visualizer.

4.2.2 Required Resources
In this section, the resources, which are required for a functional EVACART, is demonstrated and analyzed.

4.2.2.1 Microsoft Windows Operating System
Due to the limitation of the CD++ toolkit, the EVACART application can only be used in the Microsoft Windows environment. So windows operating system is required for the application.

4.2.2.2 Java Virtual Machine (JVM)
As a typical java application, EVACART requires functional Java Virtual machine to be installed in the operating system.

4.2.2.3 CD++ Simulator
The DEVS simulator in this project is from CD++ Toolkit (simuOrig.exe), which locates in the CD++ Toolkit source folder. Its absolute address is show as below: C:\eclipse\plugins\cdBuilder.simulator_1.0.0.201109022248\internal\simuOrig.exe. In this project, the same simulator has been used in the same way as it is used in CD++ Toolkit. In order to accomplish that, the source code from the CD++ Toolkit have been analyzed.

4.2.2.4 Cygwin
The application simulator depends on a functional Cygwin, which is a large collection of GNU and Open Source tools which provide functionalities similar to a Linux distribution on Windows, and a DLL (cygwin1.dll) which provides substantial POSIX API functionality stated in (Red Hat, n.d.). For referencing, the Cygwin shall be installed in the root address of its C drive in a Windows Operating System. Also, the absolute address of Cygwin shall be specified in the environment variables as shown below.

Follow below steps to add environment variables. Window system property -> advanced system settings -> Advanced -> Environment Variables -> System Variables -> Path. Double click path, and enter the Cygwin address “C:\Cygwin\bin;”
Figure 3 Cygwin set up 1

Figure 4 Cygwin set up 2
4.2.3 File Generated
This section introduces the types of files (including bat files, ma files, val files, and log files), which are generated during the application execution process.

4.2.3.1 Bat File
Bat files, as shown in the picture below, are the temperate execution files, which specify the simulation variables including the execution path, the simulator path, the simulation time, the
ma file, the out file, and the log file. The execution of a bat file will trigger the whole evacuation simulation.

Figure 7 Bat file example

4.2.3.2 Log File
Log files describe the simulation process in an event-based method in which time, action, and parameters are stated. The log files can also be used to create a 2D visualization model using CD++ library.
Ma files, as shown in below figures, are blueprints of the simulation, in which the CELL-DEVS method, the cell size, the initial cell values, and the simulation rules are specified. The simulation rules are detail rules that describe basic instructions that the simulation follows in each step.
% Author: Ryan Lawler
% Description: This model describes a follow-the-herd evacuation procedure
% for a mix of people who are familiar or unfamiliar with the surroundings.
% Those unfamiliar with the surroundings will fall in line and follow
% the people surrounding them.

[components : FTH]

type : cell
dim : (17, 17, 2)
delay : transport
defaultDelayTime : 1000
border : narrow

%floor plane neighbors
neighbors : FTH(-2,-2,0) FTH(-2,-1,0) FTH(-2,0,0) FTH(-2,1,0) FTH(-2,2,0)
neighbors : FTH(-1,-2,0) FTH(-1,-1,0) FTH(-1,0,0) FTH(-1,1,0) FTH(-1,2,0)
neighbors : FTH(0,-2,0) FTH(0,-1,0) FTH(0,0,0) FTH(0,1,0) FTH(0,2,0)
neighbors : FTH(1,-2,0) FTH(1,-1,0) FTH(1,0,0) FTH(1,1,0) FTH(1,2,0)
neighbors : FTH(2,-2,0) FTH(2,-1,0) FTH(2,0,0) FTH(2,1,0) FTH(2,2,0)

%Distance plane neighbors. Plan to use the neighbors two cells away in order
to determine direction
neighbors : FTH(-2,0,1)
neighbors : FTH(-1,-1,1) FTH(-1,0,1) FTH(-1,1,1)
neighbors : FTH(0,-2,1) FTH(0,-1,1) FTH(0,0,1) FTH(0,1,1) FTH(0,2,1)
neighbors : FTH(1,-2,1) FTH(1,-1,1) FTH(1,0,1) FTH(1,1,1)
neighbors : FTH(2,0,1)

%Initialize Floor Map
initialvalue : 0
initialCellsValue : FTH.val

%Distance plane
zone : dist_plane { (0,0,1)..<16,16,1 ) }

%Maze plane
zone : floor_plane { (0,0,0)..<16,16,0 ) }

Figure 9 MaFile Example 1
4.2.3.4 Val File

As shown in below figure, the Val files states the initial cell value of the evacuation environment. The cell environment is described in 3D, which have x, y, and z value. The value of a cell block can indicate a specific type of entity including walls, blocks, people, exits, and etc.
Assignments are as follow:

Floor Plan Setup

-1 WALL
0 EMPTY
1-8 PERSON WITH DIRECTION
9 EXIT
Distance Plane

-1 WALL
0 EXIT
1-8 GUIDANCE DIRECTION

... (more assignments)

Figure 11 Val File Example

4.2.3.5 Out File
Out files specify the simulation output. In our case, the evacuation simulation does not have an output, so the output files remain empty.

4.2.4 User Interface
This section contains the user interface introduction of the application with guidelines that users can refer to when using the application. Figures will be presented in each sub-section to support the description.
4.2.4.1 Main View

The main view is the top view of EVACART, in which user can run an evacuation simulation by creating a new environment map (or loading an existing map), selecting an egress strategy, specifying the execution stop time, and pressing the Run Simulation button. As shown in the *UI-Main View Simulation Result* figure, a dialog pop out, which states the delay time and the evacuation time. The evacuation time here is the time it takes for all the people to evacuate from the specified environment under the selected egress strategy.

From the main view, users are able to access the edit environment view, the new environment view, the compare model view, and the 2D animation view.
4.2.4.2 Edit-Environment View

As shown in figures below, the edit-environment view can be different for various egress strategies, because the number values for each entity defers from each egress strategy. Users are able to change the map cell value by selecting an entity tab and then clicking the target cell block, or by typing the desired value in keyboard.

![Edit Environment View 1](image1)

![Edit Environment View 2](image2)

4.2.4.3 New Environment View

In the new environment view, users are able to create their own environment maps with their desired size (default: 10*10, up to 100*100) for later usage.
4.2.4.4 Compare Model View
As shown in figure below, the compare model view enable the users to compare performance of various egress strategies using the generated log files from previous simulation.

As a result, EVACART generates a bar chat to demonstrate performance of the target egress strategies. The best solution will be displayed in the message window as shown in figure Compare Model View Result 1. The creation of the bar charts relies on the JFreeChart library (Gilbert, 2000). The JFreeChart project, which was found by David Gilbert in February 2000, is one of most widely used chart library for Java Applications.
Model Comparison Bar Chart Display Area

Please make sure you fill in every data entry.
Please Add/Delete data entry using “Add”/“Delete” button.
Press “Compare Model” button to see the comparison results.

Figure 18 Compare Model View (Default)

Figure 19 Compare Model View Result 1

Figure 20 Compare Model View Result 2
4.2.4.5 Animation View
As shown in the figure below, the Animation View (based on source code of CELL-DEVS animation) enables the user to visualize the simulation in a 2D platform. Certain block colors are specified by loading the existing pal files in each existing egress strategies.

![Animation View](image)

*Figure 21 Animation View*

4.2.5 Demo Video
The Demo Video of EVACART can be found on (Yang, EVACART Demo Video, 2015).

4.3 Software Architecture

4.3.1 Class Diagram
The below class diagram is generated using ObjectAid UML Explorer plugin, which is an agile and lightweight code visualization tool, developed for eclipse IDE. (ObjectAid, n.d.) Class relationships, which contains association, dependency, and generalization, are shown in the diagram along with public and static variables.
4.4 System Architecture
As displayed in Figure 1 below, the EVACART consists of five major subcomponents, which are the user interface, the visualization engine, the assessment and the simulation engine (CD++).
User interface (UI): UI is the point which users access to the system features through. UI enables the user to invoke EVACART features to select: Egress Strategy (crowd with different movement behavior such as: random, patrolled, follow-the-herd, etc.), Scenario Parameters (number of people, number and location of exits, location of security officers, exit signs, etc.), and a baseline model from model repository (airport terminal, school corridor, hospital reception area, etc.). The user has the opportunity to fine-tune the area model by using mouse-clicks and keyboard typing to identify location of barriers, exits, or crowd. Combination of all these information will be used to construct an evacuation model. Once a simulation scenario starts, the user will be provided visualization scenery in a 2D form. Emergency officers can use the UI to provide input data to the system in real-time, to make the simulation as close to reality as possible.

2D Visualization: to render 2D visualization images of the disaster event and the emergency response scenery. This component provides an interactive environment to analyze the efficiency of the emergency response process. Furthermore, the visualization results provide an excellent educational and training facility to improve emergency planning and response activities.
4.5 Design Considerations

4.5.1 Application Type
After consideration, the decision has been made to implement EVCUACRT in Java language, the reasons are stated below:

- Java is one of the most popular high level programing languages.
- The portability of the application will be enhanced by using Java.
- The original CD++ Toolkit is implemented in Java. By using Java, it saves reasonable amount of time by importing some of the library classes without converting these to a new programing language.

4.6 Demo Video
The Demo Video of the CD++ unit testing tool can be found on (Yang, DEVSUnit Demo, 2015).

4.7 DEVS Unit: CD++ Unit Test Plug-in Project
In the developing process of EVACART, to simplify the testing approach, a side project was been developed.

This project is a unit-testing plug-in for CD++Builder, which is a Discrete-Event modeling and simulation toolkit based on DEVS/Cell-DEVS formalisms for the eclipse platform. The objective of this plug-in is to conduct unit testing for DEVS models, incorporating the model’s source file (“.Cpp”) file, the event file, and expected output file.

The DEVS Unit Testing project is based on the CD++ Builder project (Wainer G., 2002). Eclipse plug-in development tutorials are used for an initial draft. Analysis has been done against the CD++ builder source code to gain sufficient understanding about how a single model get executed. According to the entered cpp model, MA files can be generated. The simulation will be executed, and the simulation output file will be compared with the pre-entered output file. Then a brief testing result will be given.

By using this plugin, the user avoids the difficulty of building MA files for each atomic model in order to carry out the testing phase.

This project includes two views: the “Unit Test Simulation Window” and the “Unit Test Result View”. In the following section, a brief description is given for the both views.

4.7.1 Unit Test Simulation Window
The “Unit Test Simulation Window” is triggered by the “DEVSUnit” button in the toolbar. This window enable the user to input necessary resources for unit test simulation, which includes the cpp model class, the expected output file, the real output file, the output log file, and the simulation stop time. In the CPP file name text box, the name of the DEVS model file shall be entered. In the second textbox (expected output file name), the name of the expected output
file shall be entered. In the Event file name text box, the name of the DEVS simulation event file shall be entered. For the output file and the log file, keep the checkbox checked to auto-generate correspond file. In the Simulation stop time text box, the simulation stop time shall be entered in the format of (hour: minute: second: millisecond).

![Unit Test Simulation project window](image)

**Figure 24 Unit Test Simulation project window**

4.7.2 Unit Test Result View
The “unit test result view” shows the detail result description for the unit test simulation. In this view, the overall test result and the detail unit test result are includes with descriptions.
This section presents the detail steps on how a user can successfully access the designed functionalizes on the unit-test plug-in.

1. The user opens eclipse application and switches to CD++ Builder perspective.

2. The user implements cpp model class.

3. The user implements the event file (.ev file) for the class model. The event file shall include time, port name and value which are separated by “Space”. Example is given below.

4. The user implements the expected output file (.out file). The expected output file shall include time, port name and value which are separated by “Space. Example is given below.

5. The user selects a CD++ Builder project or a file within a CD++ Builder project.

6. The user clicks the “DEVSUnit” button in the eclipse tool bar as shown below. Then the “Unit Test Simulate Project” window will pop out.
Figure 28 Instruction on how to open the plug-in

Figure 29 Plug-in User Interface 1
7. The user enters the full file name of the existing cpp model class file. Or, the user can press the “Browse” button and select the cpp file in the file chooser window.

8. The user enters the full file name of the existing expected output file. Or, the user presses the “Browse” button and selects the output file in the file chooser window.

9. The user enters the full file name of the existing event file or press the “Browse” button and select the event file in the file chooser window.

10. The output files can be auto-generated by checking the “Out” checkbox shown below. The user can also enter the full file name of the real output file or press the “Browse” button and select the output file in the file chooser window to save the output information for this unit test simulation.

![Figure 30 Plug-in User Interface 2](image)
11. The Log files can be auto-generated by check the “Log” checkbox shown below. The user can also enter the full file name of the existing log file or press the “Browse” button and select the log file in the file chooser window to save the log information for this unit test simulation.

![Unit Test Simulate Project](image)

*Figure 31 Plug-in User Interface 3*

12. The user enters the Simulation stop time. For example, 01:00:1:150.

13. The user clicks the “Start Simulation” Button.

14. The user observes the unit test result in the “Unit Test Result View” to see whether this test pass or not.
5. Conclusion

5.1 Summary
This project aimed at creating an application for emergency evacuation using modeling and simulation techniques, visualization, and open-source tools. As a result, an eclipse-based application with the name of EVACART has been developed which allows an emergency operation center to evaluate alternative courses of action in real-time prior to making a decision, as well as for training first responders as an alternative to tabletop and physical emergency exercises.

This research project involved learning Discrete-Event System Specification (DEVS) formalism, working with open-source M&S development environments, as well as creating eclipse-based software tools. The existing CD++ tool was modified to include automated unit testing capabilities. DEVSUnit was added as a plug-in to CD++ environment, allowing easy and fast unit testing of DEVS atomic models. The EVACTART tool, as the major outcome of this research, is open-source and freely available to all researchers and users, wishing to use an M&S framework for training and analyzing emergency evacuation scenarios. The products of this project are freely available on (Yang, Evacart & DEVS Unit Source Code, 2015).

5.2 Future Work
Even though the application EVACART has been successfully implemented, future work can be conducted to enhance the application in the following ways:

- Enhancing the visualization performance by implementing the 3D visualization module for the application.
- Extending the scope of the evacuation strategies by adding and integrating new evacuation strategies into the application.
- Improving the user interface by conducting usability tests and re-implementing the application UI.
References


