Developing a DEVS-JAVA Model to Simulate and Pre-test Changes to Emergency Care Delivery in a Safe and Efficient Manner

Shrikant Pawar[1,2(✉)] and Aditya Stanam[3]

[1] Department of Computer Science, Georgia State University, 34 Peachtree Street, Atlanta, GA 30303, USA
spawar2@gsu.edu
[2] Department of Biology, Georgia State University, 34 Peachtree Street, Atlanta, GA 30303, USA
[3] College of Public Health, The University of Iowa, UI Research Park, #219 IREH, Iowa City, IA 52242-5000, USA

Abstract. Patients' overcrowding in Emergency Department (ED) is a major problem in medical care worldwide, predominantly due to time and resource constraints. Simulation modeling is an economical approach to solve complex healthcare problems. We employed Discrete Event System Specification-JAVA (DEVS-JAVA) based model to simulate and test changes in emergency service conditions with an overall goal to improve ED patient flow. Initially, we developed a system based on ED data from South Carolina hospitals. Later, we ran simulations on four different case scenarios. 1. Optimum number (no.) of doctors and patients needed to reduce average (avg) time of assignment (avg discharge time = 33 min). 2. Optimum no. of patients to reduce avg discharge time (avg wait time = 150 min). 3. Optimum no. of patients to reduce avg directing time to critical care (avg wait time = 58 min). 4. Optimum no. of patients to reduce avg directing time to another hospital (avg wait time = 93 min). Upon execution of above 4 simulations, 4 patients got discharged utilizing 3 doctors; 5 patients could be discharged from ED to home; 2 patients could be transferred from ED to critical care; 3 patients could be transferred from ED to another hospital. Our results suggest that the generated DEVS-JAVA simulation method seems extremely effective to solve time-dependent healthcare problems.

Keywords: DEVS-JAVA · Simulation · Emergency department (ED)

1 Introduction

1.1 Problem Statement

An emergency department (ED), also known as accident & emergency department (A&E), provides acute care for patients who attend hospital without prior appointment. The EDs of most hospitals customarily operate 24 h a day, 7 days a week and over-crowding of patients in EDs is an increasing problem in countries around the world. ED overcrowding has been shown to have many adverse consequences such as increased medical errors, decreased quality of care and subsequently poor patient outcomes,

I. Rojas et al. (Eds.): IWBBIO 2019, LNBI 11466, pp. 3–14, 2019.
https://doi.org/10.1007/978-3-030-17935-9_1

increased workload, ambulance diversions, increased patient dissatisfaction, prolonged patient waiting times and increased cost of care [1, 2]. There are different types of computer simulations techniques utilized to address these types of issues in the field of medicine and health. Some of such tools include discrete event simulation (DES), system dynamics (SD) and agent-based simulations (ABS). DES is identified to be one of the best method to address this problem due to its capability of replicate the behavior of complex healthcare systems over time. DEVS stands for Discrete Event System Specification which is a time extended finite state machine [3]. Even more suitable method of modeling for this problem is needed due to the fact that ED system is more sensitive for constraints and limitations imposed by time. We believe that the closest technique to replicate this process could be achieved through DEVS-JAVA [4], in the current article we have modelled the system in ED using DEVS-JAVA to explore the possibility of using the developed model to simulate changes of services conditions and understand the outputs so that a better service to its patients can be provide with the ED.

1.2 Modeling and Simulation Goals

To address these concerns, our specific objectives are to conduct patient flow simulation through ED in terms of key assumptions, systems requirements, and input and output data, and to identify the usefulness of this simulation method for service redesign and evaluating the likely impact of changes related to the delivery of emergency care. We compared our simulation model with current ED flow data from known existing database, which will report on differences in conclusions about ED performance with our simulation model and existing ER practice to solve the problem of ED overcrowding. The simulation model was implemented in DEVS-JAVA. There can be different targets of executing simulations based on this system. But for the purpose of simplification, we considered the following objectives as our goals of this project.

1. Identifying how many doctors and ambulance are optimal for decreasing average time of assignment.
2. Understand how many patients are optimal for decreasing average time of discharge.
3. Identifying how many patients are optimal for decreasing average time of directing to critical care treatment.
4. Studying how many patients are optimal for decreasing average time of directing patient to another hospital.

These types of simulations are extremely useful for hospitals to make decisions on its resource allocations. The simulations importance and contribution is even more significant as the results can be observed without practically implementing a case. This approach can be utilized to reduce risks in any real world system with trustworthy estimations.

2 Materials and Methods

The main focus of this simulation process is to identify the possible modifications of the existing system in order to improve the overall efficiency of the system. One of the best approach to solve this problem is to first develop a system with current statistics and then

incorporate modifications (changing the parameters) to the system, to derive the conclusions. Furthermore, the experimental conclusions are needed to be compared to existing ED data, for which we went to ED timings in the South Carolina hospitals (Table 1). The South Carolina hospital data was collected between April 2016 and April 2017 from Centers for Medicare and Medicaid Services (CMS) [5]. The data was collected throughout the day for all the hospitals without any distinct day timings. CMS established standard definitions and a common metric to accurately compare different hospitals. In most cases someone must manually write down the time a patient was seen, so the times are not always precise. To combat this, emergency rooms outfit doctors and nurses with electronic badges now wirelessly record exact times. According to CMS, hospitals have 30 days to review their data before submitting it to the government. The agency places most of the responsibility on hospitals for making sure their data is correct before making it public. Average values of timings are strong enough to represent the whole population and comparing to our simulation results. Given the data in Table 1, we built DEVS-JAVA models for each of these four cases/columns for simulating respective timings. We ran different simulations on each of these cases to improve through put within the above stipulated time ranges. For example, the question for case 1 was, what will be the optimal number of doctors to be utilized in a situation so as to maximize the patients discharged within the average waiting time of 33 min.

Table 1. Waiting times of different stages of South Carolina hospitals (Waiting times (minutes) for a patient attended by a doctor). The first column refers to the waiting time taken by each hospital for the process of admitting a patient from the accident cite until the patient is attended by a physician. Second column represents the time taken to the process of going through emergency front desk to a doctor for prescription and then the time taken send the prescribed patient to his home. The third column is the time associated with a patient getting a critical care treatment. The fourth column is a transfer time, which is the total time required by a hospital to redirect a patient to a different hospital in a condition that it cannot be treated due to lack of human or physical resources.

Hospital	Waiting time	Time until sent home	Critical case time	Transfer time
Abbeville Area Medical Center	27	96	64	38
Aiken Regional Medical Center	47	189	60	149
Anmed Health	75	204	87	95
Baptist Easley Hospital	54	142	55	53
Beaufort County Memorial Hospital	19	152	34	120
Bon Secours-St Francis Xavier Hospital	35	170	50	68
Cannon Memorial Hospital	34	94	82	68
Carolina Pines Regional Medical Center	25	118	61	80
Carolinas Hospital System	22	163	60	89
Carolinas Hospital System Marion	19	134	26	65
Chester Regional Medical Center	24	109	52	51
Coastal Carolina Hospital	34	207	64	146

(*continued*)

Table 1. (*continued*)

Hospital	Waiting time	Time until sent home	Critical case time	Transfer time
Colleton Medical Center	6	103	28	70
Conway Medical Center	24	131	47	118
East Cooper Medical Center	26	116	36	56
Ghs Greenville Memorial Hospital	51	221	34	135
Ghs Greer Memorial Hospital	31	173	62	110
Ghs Hillcrest Memorial Hospital	30	149	45	101
Ghs Laurens County Memorial Hospital	36	141	60	104
Ghs Oconee Memorial Hospital	46	176	80	61
Grand Strand Regional Medical Center	8	122	21	90
Hampton Regional Medical Center	37	125	53	64
Hilton Head Regional Medical Center	12	167	66	125
Kershawhealth	33	116	48	70
Lake City Community Hospital	45	108	88	65
Lexington Medical Center	43	202	49	176
Mary Black Health System Gaffney	27	146	56	105
Mary Black Health System Spartanburg	18	150	71	110
Mcleod Health Cheraw	20	117	48	42
Mcleod Health Clarendon	36	150	108	63
Mcleod Loris Hospital	34	137	50	43
Mcleod Medical Center - Dillon	35	168	52	36
Mcleod Regional Medical Center-Pee Dee	70	224	101	104
Mount Pleasant Hospital	5	96	26	41
Musc Medical Center	24	162	47	99
Newberry County Memorial Hospital	32	138	54	68
Palmetto Health Baptist	36	193	59	174
Palmetto Health Baptist Parkridge	35	164	58	144
Palmetto Health Richland	28	205	78	207
Palmetto Health Tuomey Hospital	70	190	70	124
Pelham Medical Center	39	150	70	81
Piedmont Medical Center	38	176	72	143
Providence Health	32	152	65	152
Roper Hospital	10	103	31	61
Self-Regional Healthcare	49	154	79	50
Spartanburg Medical Center	68	218	80	145
Springs Memorial Hospital	22	126	58	75
St Francis-Downtown	44	172	75	100
Tidelands Health	20	122	40	93
Tidelands Waccamaw Community Hospital	28	138	44	89
Trident Medical Center	8	115	24	100
Trmc Of Orangeburg & Calhoun	47	172	64	93
Union Medical Center	31	130	86	66
Associated average time (Minutes)	33	150	58	93

2.1 Models Developed

1. **Simulation case 1 (Simulation for admitting a patient to ED with doctor allocation and performing initial treatments referred as "door to doctor time"):** Compilation of all the java codes was conducted on a computer with a 3.07 GHz quad-core CPU and 24 G memory supported by a NVIDIA GTX 580 GPU card with 3G memory on NetBeans integrated development environment (IDE). All the java classes and compilation codes are submitted on GitHub repository which can be found on the corresponding authors account (https://github.com/spawar2/Devs_Java_Patient_Flow_SimulationinEMS). The DEVS-JAVA provides a collection of two methods of execution. One is using the *SimView*, which is a graphical representation and the other being running a program directly through a java class to run a simulation for given time using *genDevs.simulation.coordinator*. *SimView* approach is more suitable to graphically understand the flow of each activity, while the second approach is useful for getting associated observations for a specified time of simulation. The main setup for the project is shown in Fig. 1(a). This case simulation is shown in Fig. 1(b). The simulation is based on the assumptions that ambulances will be available every 10 min and a doctor becomes available every 5 min to the queue. Also, we assume that each patient is taking an average time for 10 min to undergo a treatment process. We have utilized the ambulance and doctor generator models which are namely *Accident Site* and *Doctor Assigned*. Also, we are using a simulator for ED that we name as *Emergency Dept*. *Operator* here is basically a transducer that observes number of patients and doctors flowing through EMS. Running this simulation provides an output for one iteration of this case (Fig. 1(f)). The logging of the data is important to make sure that simulation is working properly and also to identify how many resources utilized for the given time range and what is the corresponding throughput (essentially the number of patients discharged). With the significant number of iterations with for-loops, identifying patters in simulations is possible. The final goal of this case simulation is to identify the fact *"How many doctors and ambulances are optimal for decreasing average time of assignment (average wait time = 33 min)?"*.
2. **Simulation case 2 (Simulation of the patient being sent home after treatment referred as "door to discharge time"):** This case simulation is shown in Fig. 1(c). This simulation will identify the time required to do the process of a patient discharged to home. It will have a *Front Desk* generator that generates the patients to the process of medication prescribed which is further redirected to the generator *Sent Home*. An *Operator* is seen sitting beside and working as a transducer that works on the count of patients flowing in this flow. We have utilized a range of simulation cycles to answer this case question *"How many patients are optimal for decreasing average time of discharge (Average wait time = 150 min)?"*.
3. **Simulation case 3 (Simulation of a patient being transferred to the critical care):** This case simulation is shown in Fig. 1(d). This simulation identifies the time constraints associated with a patient who is assigned to critical care. The transferred to critical care criteria according to CMS only applies to the "initiation of a critical

care level of intervention" for the current data. This simulation is having the generators for *Emergency Front Desk* and *Critical Care*. Also, it is having a transducer called *Operator* which keeps the track of all the ongoing information. The question of interest in this simulation is "How many patients are optimal for decreasing average time of directing them to critical care treatment (Average wait time = 58 min)?".

4. **Simulation case 4 (Simulation of a patient being transferred to another hospital):** This case simulation is shown in Fig. 1(e). The task of this simulation is to clearly identify the timings associated with a patient being transferred to another hospital. The setup will have an *Emergency Front Desk* generator, *Another Hospital* generator and the *Operator* as an observer. The question of interest here is "How many patients are optimal for decreasing average time of directing a patient to another hospital (Average wait time = 93 min)?".

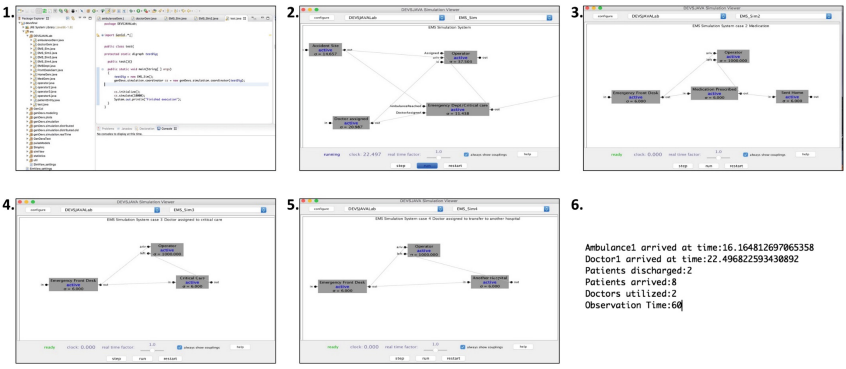

Fig. 1. (a) The main setup for the project. (b)–(e): Different simulation cases. (f): The command line execution output for one of the test simulation. Different generators that we developed for these simulations are *ambulanceGenr*: This model is a ViewableAtomic that is used to generate and simulate patients taken form accident site to the hospital though ambulances. We have set 10 min as the default time for the frequency of ambulance availability. *doctorGenr*: This is also one of the main important generators that will represent one of the main resources of the simulated hospital system which is the doctor. We will be assuming that on average the doctors will be available in each 5 min, while we develop this model. *FrontDeskGenr*: This is also a generator model that will simulate the patient flow thought the Emergency Front Desk. *HomeGenr*: This referees to the generator model that refers to the patients being set to home. *MedGenr*: This is a DEVS model that will represent the process of medications being prescribed by a doctor. *patientEntity*: This is the most important entity that we utilized during these simulations to represent the patient. It represents the behavior of a patient and hence it is utilized in different stages of medical treatments to measure the time for each step. This is one of the most important measurements we used in the discussion of our observations. *Operators*: We are utilizing transducers that we name as operator for each of these simulations. So, we have 4 different transducers that we name as operator0 to operator4.

3 Results

1. *"How many doctors and patients are optimal for decreasing average door to doctor time (average wait time of 1 patient discharge = 33 min)?"*

Results are obtained using execution of simulation using *genDevs.simulation.coordinator* for a range of iterations and observing and summarizing the results against the simulation goal questions. Table 2 states the simulation results of case 1. Using these results and selecting optimized value highlighted in bold, we can calculate the optimal patients discharged by a simple linear Eq. 1 as follows (average wait time = 33 min):

$$\begin{aligned} Patients\,Discharged = {} & (No\ of\ patients\ discharged / Observed.time) \\ & * Current\ average\ comparison\ time. \end{aligned} \tag{1}$$

When applied ((8/60) * 33 = 4.4 $\sim$ 4), it gives 4 patients discharged (current time is 1 patient discharge) utilizing 3 doctors as an optimal solution for this situation. We followed the same approach for the rest of cases with a range of simulations with following results.

2. *"How many patients are optimal for decreasing average door to discharge time (Average wait time = 150 min)?"*.

Our optimal experimental results states that at most 5 patients can be discharged (current time is 1 patient discharge) from ED to home, for the average time range of 150 min.

3. *"How many patients are optimal for decreasing average time of directing them to critical care treatment (Average wait time = 58 min)?"*

Our experimental results states that at most 2 patients can be transferred (current time is 1 patient transferred) from ED to critical care, for the average time range of 58 min.

4. *"How many patients are optimal for decreasing average time of directing a patient to another hospital (Average wait time = 93 min)?"*.

Our experimental results states that at most 3 patients can be transferred from transferred (current time is 1 patient transferred) from ED to another hospital, for the average time range of 93 min.

Table 2. Simulation results of case 1.

Observed time (minutes)	No. of doctors	Finish time of last doctor (minutes)	Patients discharged
60	1	9.7	3
60	2	10.4	5
60	**3**	**8.7**	**8**
60	4	10.6	6

4 Discussion and Conclusions

Based on our simulation results, the patient flow from ED can be significantly improved with suggested numbers of doctors and patient arrival timings. There is no mechanism in place to control the number of patients who arrive in the ED for evaluation, and an ideal ED input number cannot be derived. However, the current model can be adopted to the input of patients known to occur at certain day and time. A preliminary survey with average number of input patients arriving in specific ED is needed for the model to operate. This can be easily done in any hospital and a custom parameters can be applied to models evaluation. There are also some other types of computer simulation techniques utilized to address similar issues in the field of medicine and health. Some of such tools include Discrete Event Simulation (DES), System Dynamics (SD) and Agent-based Simulations (ABS). DES is identified to be one of the best method to address this problem due to its capability of replicating the behavior of complex healthcare systems over time. Future validation of our simulated data can be performed using this technique.

Acknowledgments. 1. Support from the Georgia State University Information Technology Department (GSU IT) for server space is gratefully acknowledged.

2. Support from the Artificial Intelligence and Simulation Research Group, Department of Electrical and Computer Engineering, The University of Arizona for providing DEVS-JAVA architecture is gratefully acknowledged.

Supporting Information. No external funding has been utilized for this analysis.

Appendix - Selected Pseudo Code Segments

This section contains selected pseudo code segments that are related to the simulation of case 1 discussed in this article. This could help the readers to reproduce and improve our simulation results.

1. Generator: ambulanceGenr

```
package DEVSJAVALab;
import simView.*;
import java.lang.*;
import genDevs.modeling.*;
import genDevs.simulation.*;
import GenCol.*;
import util.*;
import statistics.*;
public class ambulanceGenr extends ViewableAtomic{
  protected double int_gen_time;
  protected int count;
  protected rand r;
  public ambulanceGenr() {this("ambulanceGenr", 7);}
public ambulanceGenr(String name,double period){
   super(name);
   addInport("in");
   addOutport("out");
   int_gen_time = period ;
}
public void initialize(){
   holdIn("active", int_gen_time);
   r = new rand(123987979);
   count = 0;
}
public void  deltext(double e,message x)
{
Continue(e);
   for (int i=0; i< x.getLength();i++){
     if (messageOnPort(x, "in", i)) { //the stop message from tranducer
       passivate();
     }
   }
}
public void  deltint( )
{
if(phaseIs("active")){
   count = count +1;
//   holdIn("active",int_gen_time);
   holdIn("active",6+r.uniform(int_gen_time));
}
else passivate();
}
public message  out( )
{
//System.out.println(name+" out count "+count);
   message  m = new message();
//   content con = makeContent("out", new entity("car" + count));
   content con = makeContent("out", new patientEntity("Ambulance" + count, 5+r.uniform(20), 50+r.uniform(100), 1));
   m.add(con);
  return m;
}}
```

2. Generator: doctorGenr

```
package DEVSJAVALab;
import simView.*;
import java.lang.*;
import genDevs.modeling.*;
import genDevs.simulation.*;
import GenCol.*;
import util.*;
import statistics.*;
public class doctorGenr extends ViewableAtomic{
  protected double int_gen_time;
  protected int count;
  protected rand r;
  public doctorGenr() {this("doctorGenr", 30);}
public doctorGenr(String name,double period){
   super(name);
   addInport("in");
   addOutport("out");
   int_gen_time = period ;
}
public void initialize(){
   holdIn("active", 2);
   r = new rand(2);
   count = 0;
}
public void  deltext(double e,message x)
{
Continue(e);
}
public void  deltint( )
{
if(phaseIs("active")){
   count = count +1;
   holdIn("active",20+r.uniform(int_gen_time));
}
else passivate();
}
public message  out( )
{
//System.out.println(name+" out count "+count);
   message  m = new message();
//   content con = makeContent("out", new entity("truck" + count));
   content con = makeContent("out", new patientEntity("Doctor" + count, 10+r.uniform(30), 100+r.uniform(100), 1));
   m.add(con); return m;
}}
```

3. **Generator: patientEntity**

```
package DEVSJAVALab;
import GenCol.*;
public class patientEntity extends entity{
  protected double processingTime;
  protected double price;
  protected int priority;
  public patientEntity(){
                this("patient", 10, 10, 1);
  }
  public patientEntity(String name, double _procTime, double _price, int _priority){
                super(name);
                processingTime = _procTime;
                price = _price;
                priority = _priority;
  }
  public double getProcessingTime(){
                return processingTime;
  }
  public double getPrice(){
                return price;
  }
  public int getPriority(){
                return priority;
  }
  public String toString(){
                return name+"_"+(double)((int)(processingTime*100))/100;
                //return name+"_"+((double)((int)(processingTime*100)))/100;
  }}
```

4. **Generator: EMS_Sim**

```
package DEVSJAVALab;
import simView.*;
import java.awt.*;
import java.io.*;
import genDevs.modeling.*;
import genDevs.simulation.*;
import GenCol.*;
public class EMS_Sim extends ViewableDigraph{
public EMS_Sim(){
   this("EMS Simulation System");
}
public EMS_Sim(String nm){
   super(nm);
   emsConstruct();
}
public void emsConstruct(){
   this.addOutport("out");
   ViewableAtomic amb_genr = new ambulanceGenr("Accident Site",10);
   ViewableAtomic doc_genr = new doctorGenr("Doctor assigned",1);
   ViewableAtomic ems_dpt = new EMSDept("Emergency Dept/Critical care");
   ViewableAtomic operator = new operator ();
    add(amb_genr);
    add(ems_dpt);
    add(doc_genr);
    add(operator);
    addCoupling(amb_genr,"out",ems_dpt,"AmbulanceReached");
    addCoupling(amb_genr,"out",operator,"ariv");
    addCoupling(doc_genr,"out",operator,"ariv");
    addCoupling(ems_dpt,"out",operator,"Assigned");
    addCoupling(doc_genr,"out",ems_dpt,"DoctorAssigned");
    addCoupling(ems_dpt,"out",this,"out");}
```

References

1. Mohiuddin, S., Busby, J., Savović, J.: Patient flow within UK emergency departments: a systematic review of the use of computer simulation modelling methods. BMJ Open (2017)
2. Institute of Medicine (US) Committee on Assuring the Health of the Public in the 21st Century: The Future of the Public's Health in the 21st Century, vol. 5. National Academies Press, Washington (DC) (2002). The Health Care Delivery System. https://www.ncbi.nlm.nih.gov/books/NBK221227/
3. Zeigler, B., Hessam, S.: Introduction to DEVS Modeling and Simulation with JAVA: Developing Component-Based Simulation Models. http://www.cs.gsu.edu/xhu/CSC8350/DEVSJAVA_Manuscript.pdf
4. Artificial Intelligence and Simulation Research Group: DEVS-Java Reference Guide (1997). https://grid.cs.gsu.edu/~fbai1/paper/4560.pdf
5. Hospital Compare datasets: Data.medicare.gov (2018). https://data.medicare.gov/data/hospital-compare/