**ASSIGNMENT – 1**

**ON**

**HEALTHY SMILE DENTAL CLINIC MODEL**

**Methodologies for Discrete Event Modelling and Simulation**

**(Course Code: SYSC 5104)**

**FALL-2017**

*Submitted by*

**Ekambir Sidhu**

U-Ottawa Student ID: 300036992

Carleton Student ID: 101098396

**Manohar Deep Singh Gill**

Carleton Student ID: 101055138

*Submitted to*

**Professor Gabriel A. Wainer**

Department of Systems and Computer Engineering

Carleton University

**Date of Submission: 3rd November, 2017 (Friday)**

# INDEX

[1. Introduction](#_Toc57980161) 3

2. [Basic Building Blocks/Components of the proposed system](#_Toc57980162) 3

3. Specifications[Cashier Model](#_Toc57980163) 4

4. [Objectives](#_Toc57980164) 4

5. [DEVS Specification 5](#_Toc57980165)

5.1 [Token Generator Model Specification](#_Toc57980166) 5

[5.2 Receptionist Model Specification](#_Toc57980167) 6

[5.3 Dentist Model Specification](#_Toc57980168) 8

5.4 [Comptroller Model Specification](#_Toc57980169) 9

5.5 Receptionist/Dentist Coupled Model Specification 10

[5.6 Healthy Smile Dental Clinic Model Specification](#_Toc57980171) 11

6. Test Cases for Token Generator Model12

7 Test Cases for Receptionist Model 12

8 Test Cases for Doctor Model12

9 Test Cases for Comptroller Model12

10. Model Testing Strategy/Methodology13

[10.1 Steps for testing the models](#_Toc57980174) 14

11. [Simulation Analysis with various examples](#_Toc57980177) 15

[11.1 Token Generator Model](#_Toc57980178) 16

11.2 Reception Model Test[Simulation](#_Toc57980177) 16

[11.3 Comptroller Model Test](#_Toc57980178) 17

11.4 Doctor Model Test17

[11.5 Doctor1 Model Test](#_Toc57980178) 18

11.6 Doctor2 Model Test18

[11.7 Doctor 3 Model Test](#_Toc57980178) 19

11.8 Doctor 4 Model Test19

[11.9 Doctor 5 Model Test](#_Toc57980178) 19

11.10 [Dental Clinic Model with doctors having unequal service rate](#_Toc57980177) 20

12. [Conclusion](#_Toc57980178) 21

**APPENDIX**21

A1 [readme.txt](#_Toc57980178) 21

A2 [dentalclinic.output.txt](#_Toc57980178) 23

**HEALTHY SMILE DENTAL CLINIC MODEL**

**PART – I**

1. **INTRODUCTION**

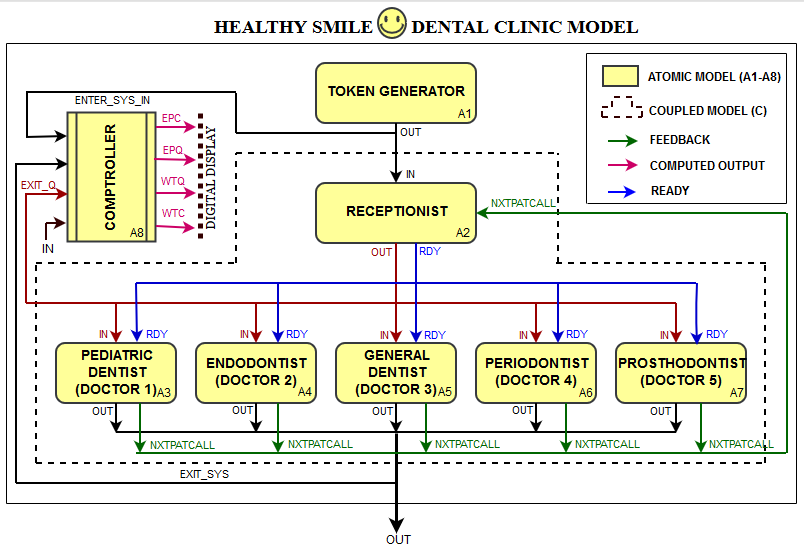
The Assignment 1 aims at designing, modelling and simulation of *‘Healthy Smile Dental Clinic’* as discrete event dynamic system model using DEVS (Discrete Event System Specification) formalism. The model aims at computing the efficiency of the dental clinic in terms of delivering the medical services to the patients in the stipulated timeframe. The proposed model of the Healthy Smile Dental Clinic has been shown in Figure 1 below. The various modules in the proposed model have been described as follows:

Figure 1 – Proposed Model of Healthy Smile Dental Clinic

1. **BASIC BUILDING BLOCKS/COMPONENTS IN THE PROPOSED MODEL**
2. **Token Generator –** The patients coming into the clinic first generate a token using Token Generator.
3. **Receptionist –** After generating the token, the patients goes to receptionist in sequence who allocates the respective dentist to the patient and also collects the respective dentist consultation fee from the patient.
4. **Dentist(s) –** The five types of dental dentists who aim at serving the different dental problems – general dentist (Doctor 1 - for regular dental check-up), endodontist (Doctor 2 - serves root canal, tooth tissue problems and emergency), periodontist (Doctor 3 - deals with gum problems), prosthodontist (Doctor 4 - deals in tooth replacement) and pediatric dentist (Doctor 5 - children and teen dentist) are available in the clinic. It is to be noted that the different dentists require different time to serve the patients e.g. the endodontist requires the maximum time for root canal surgery and emergency patients in comparison to other dental doctors. The final proposed model (dental clinic) will take into account the different time slots required by different dental dentists to serve their patients.
5. **Comptroller –** Comptroller is solely responsible for all the calculations based on the data collected from token generator (number of patients who entered clinic), receptionist (number of patients passed on by the receptionist to respective dentist’s cabins and being served) and patients exiting the clinic. The various computations(functions) to be performed by the comptroller are:
6. Total number of expected patients in the clinic (E\_P\_C),
7. Number of expected patients in the queue (E\_P\_Q) i.e. number of patients within the coupled model of token generator, receptionist and dentist’s cabin (shown by dotted line in the Figure 1),
8. Patient waiting time within the queue (W\_T\_Q) in hours, and
9. Patient waiting time in clinic (W\_T\_C) in hours

Therefore, the Comptroller can be called the *heart* of the clinic model as it is mainly responsible for computing the efficiency of dental services deliverability by the clinic to the patients within the specific time period. The output of comptroller can be displayed on Digital Display (e.g. LCD or LED Display) installed in front of receptionist and patient waiting area which displays the status i.e. the number of patients being served in dentist’s cabins and total number of patients present in clinic.

1. **SPECIFICATIONS**

Number of Atomic Models generated for testing = 04

Number of Coupled Models generated for testing = 06

Total Models generated for testing = 10

1. **OBJECTIVES**

**Objective I**

To implement the dental clinic model with the different dentists (doctors) serving the patients at different service rates (patients per hour) and the patients arriving at different arrival rates and to compute the following parameters for the clinic model

1. Total number of expected patients in the clinic (E\_P\_C),
2. Number of expected patients in the queue (E\_P\_Q)
3. Patient waiting time in queue (W\_T\_Q) in hours, and
4. Patient waiting time in clinic (W\_T\_C) in hours

**Objective 2**

To implement the dental clinic model with the different number of dentists (doctors) with the different patient service rates and to check the performance of system in terms of the above parameters.

**Objective 3**

To implement the different atomic models (tokengenerator, receptionist, doctor and token generator) individually and simulate and check their performance by generating a suitable test benches (events).

**PART – II**

1. **DEVS FORMALISM**

The various parameters and variables used in the DEVS coupled/atomic specifications have been tabulated below in Table 1 below:

Table 1 – Parameters/Variables used for DEVS Model Specification

|  |  |
| --- | --- |
| **ABBREVIATION** | **DESCRIPTION** |
| **DCM** | Dental Clinic Model |
| **Dentist** | Total Dentists in Clinic |
| **IL** | Internal Links |
| **EIL** | External Incoming Links |
| **EOL** | External Outgoing Links |

### TOKEN GENERATOR Model Specification

|  |
| --- |
| **Specification:**  tokengenerator = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}  X = None  Y = {TOKEN\_OUT}  S = {Active}  EF = None  IF = IF (Active) = Active  OF (Active) = TOKEN\_OUT  TA (Active) = delay for reconfigurable distribution |

#### **Implementation:**

|  |
| --- |
| ***Initialization Function****:*  {  Initialize all values  σ = 0  } |

|  |
| --- |
| ***Internal Function:***  {  If (all patients were generated for previous time unit)  then  get the patients generation rate based on a distribution for this time unit  elseif (a patient is required to be generated in this time unit)  σ = patient interval time within this time unit  else  σ = time unit  } |

|  |
| --- |
| ***Output Function****:*  {  if (a patient is required to be generated in this time unit)  {  increment the ID by one  send the D on the TOKEN\_OUT port  decrement the number of patients to be generated in this time unit  }  } |

### RECEPTIONIST Model Specification:

|  |
| --- |
| **Specification:**  receptionist = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}  X = {REP\_IN, NXT\_PAT\_CALL}  Y = {REP\_OUT, RDY}  S = {(receptionist. span >0 with done), (receptionist.span >0 without done),  (receptionist.span = 0 with done), (receptionist.span = 0 without done}  EF: EF ((receptionist.span >0 with done), in) = (receptionist.span >0 with done)  EF ((receptionist.span >0 without done), in) = (receptionist.span >0 without done)  EF ((receptionist.span = 0 with done), in) = (receptionist.span >0 with done)  EF ((receptionist.span = 0 without done), in) = (receptionist.span >0 without done)  EF ((receptionist.span >0 with done), done) = (receptionist.span >0 with done)  EF ((receptionist.span >0 without done), done) = (receptionist.span >0 with done)  EF ((receptionist.span = 0 with done), done) = (receptionist.span = 0 with done)  EF ((receptionist.span = 0 without done), done) = (receptionist.span =0 with done)  IF = IF (receptionist.span >0 with done) =  (receptionist.span = 0 with done) → If receptionist.span = 1 and done > 1  (receptionist.span = 0 without done) → If receptionist.span = 1 and last done  (receptionist.span > 0 with done) → If receptionist.span >1 and done > 1  (receptionist.span > 0 without done) → If queue.span >1 and last done    IF (any other case) = passivate  OF (receptionist.span >0 with done) = REP\_OUT and RDY  TA (receptionist.span >0 with done) = 0 |

#### **Implementation:**

|  |
| --- |
| ***Initialization Function****:*  {  Initialize NXT\_PAT\_CALLs list, NXT\_PAT\_CALL count and receptionist element list  } |

|  |
| --- |
| ***External Function****:*  {  if (message arrives on the “REP\_IN” port)  then  {  push the element to the back of the queue in receptionist  elseif (this is the first element and there are NXT\_PAT\_CALL’s)  then  {  σ = 0  }  else if (there are no NXT\_PAT\_CALLs)  then  {  passivate  }  }  if (message arrives on the “NXT\_PAT\_CALL” port and valid ID)  then  {  register NXT\_PAT\_CALL  increment NXT\_PAT\_CALL count by one  if (this is the first NXT\_PAT\_CALL and there are elements in the queue)  then  {  σ = 0  }  else  {  passivate  }  }  } |

|  |
| --- |
| ***Internal Function:***  {  pop the front element from the queue  if (there are more NXT\_PAT\_CALL’s and there are elements in the queue) then  σ = 0  else  passivate  } |

|  |
| --- |
| ***Output Function****:*  {  get the ID.  send the ID on the “RDY” port  send the value on the “REP\_OUT” port  } |

### DENTIST Model Specification:

|  |
| --- |
| Specification: dentistN = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}  X = {PAT\_IN, DOC\_RDY}  Y = {PAT\_OUT, PAT\_CALL}  S = {pat\_treated, pat\_under\_treatment}  EF: EF (pat\_treated, in) = pat\_under\_treatment  EF (pat\_treatedt, RDY) = pat\_treated  EF (pat\_under\_treatment, in) = NA  EF (pat\_under\_treatment, RDY) = NA  IF = IF (pat\_under\_treatment) = passivate  OF (pat\_treated) = PAT\_EXIT  OF (pat\_under\_treatment) = PAT\_EXIT, NXT\_PAT\_CALL  TA (pat\_under\_treatment) = serving time (depends on type of Dentist) |

#### **Implementation:**

|  |
| --- |
| ***Initialization Function****:*  {  Send the initial NXT\_PAT\_CALL to the queue in the receptionist model  } |

|  |
| --- |
| ***External Function****:*  {  if (passive and message is received on “RDY” port with ID)  then  {  dentist is ready for next patient treatment  }  else if (passive and message is received on “PAT\_IN” port and Dentist is ready for next patient treatment)  {  σ = serving time based on the type of dentist  }  } |

|  |
| --- |
| ***Internal Function***  {  dentist is not ready for patient treatment  passivate  } |

|  |
| --- |
| ***Output Function****:*  {  if (dentist is ready for patient treatment)  then  {  Send patient out on PAT\_EXIT  Output the ID on the NXT\_PAT\_CALL  }  } |

### COMPTROLLER Model Specification:

|  |
| --- |
| **Specification:**  Efficiency = {X, Y, S, Internal Function (IF), External Function (EF), Output Function (OF), TA}  X = {TOKEN\_IN, REP\_PAT\_OUT, SERVED\_PAT, COMPUTE}  Y = {E\_P\_C, W\_T\_C, E\_P\_Q , W\_T\_Q}  S = {active, passive}  EF = EF (active || passive, ) = active  EF (active || passive, TOKEN\_IN || REP\_PAT\_OUT || SERVED\_PAT) = passivate  IF = IF (active) = passivate  OF (active) = E\_P\_C, W\_T\_C, E\_P\_Q, W\_T\_Q  TA (active) = 0 |

#### **Implementation:**

|  |
| --- |
| ***Initialization Function****:*  {  initialize all data collection values and lists.  } |

|  |
| --- |
| ***External Function****:*  {  if (message is received from TOKEN\_IN)  then  {  count this patient  time stamp this patient  compute E\_P\_C  compute E\_P\_Q  }  elseif (message is received from REP\_PAT\_OUT)  then  {  count this patient  compute E\_P\_C  compute W\_T\_C  }  elseif (message is received from SERVED\_PAT)  then  {  count this patient  compute E\_P\_Q  compute W\_T\_Q  }  elseif (message is received from COMPUTE)  then  {  σ = 0  } |

|  |
| --- |
| ***Internal Function:***  {  passivate  } |

|  |
| --- |
| ***Output Function****:*  {  output E\_P\_C  output W\_T\_C  output E\_P\_Q  output W\_T\_Q  } |

## Receptionist/Dentist Coupled Model Specification

|  |
| --- |
| ***Specification:***  DCM = {X, Y, {tokengenerator, receptionist, Dentist, comptroller}, EIL, EOL, IL, Select}  X = {COMPUTE}  Y = {PAT\_EXIT, W\_T\_Q, W\_T\_C, E\_P\_Q, E\_P\_C}  IL = { (tokengenerator.TOKEN\_OUT, receptionist.REP\_IN),  (receptionist.REP\_OUT, dentist.PAT\_IN),  (receptionist.RDY, dentist.DOC\_RDY),  (dentist.PAT\_CALL, receptionist.NXT\_PAT\_CALL),  (tokengenerator.TOKEN\_OUT, comptroller.TOKEN\_IN),  (receptionist.REP\_OUT, comptroller.REP\_PAT\_OUT),  (dentist.PAT\_OUT, comptroller.SERVED\_PAT) }  EOL = {(dentist.PAT\_OUT, DCM.PAT\_EXIT), (comptroller.E\_P\_C, DCM.E\_P\_C),  (comptroller.W\_T\_C, DCM.W\_T\_C), (comptroller.E\_P\_Q, DCM.E\_P\_Q),  (comptroller.W\_T\_Q, DCM.W\_T\_Q)}.  EIL = {(DCM.COMPUTE, comptroller.COMPUTE)}  Select: (tokengenerator, receptionist, Dentist, comptroller) = tokengenerator  Select: (comptroller, any module) = any other module  Select: (receptionist, Dentist, comptroller) = receptionist  Select: Dentist = any Dentist |

## Healthy Smile Dental Clinic Coupled Model Specification

|  |
| --- |
| ***Specification:***  DCM = {X, Y, {tokengenerator, receptionist, {Dentist\_1, Dentist\_2, Dentist\_3, Dentist\_4, Dentist\_5}, comptroller}, EIL, EOL, IL, Select}  X = {COMPUTE}  Y = {PAT\_EXIT, W\_T\_Q, W\_T\_C, E\_P\_Q, E\_P\_C}  IL = { (tokengenerator.TOKEN\_OUT, receptionist.REP\_IN),  (receptionist.REP\_OUT, dentistN.PAT\_IN),  (receptionist.RDY, dentistN.DOC\_RDY),  (dentistN.PAT\_CALL, receptionist.NXT\_PAT\_CALL),  (tokengenerator.TOKEN\_OUT, comptroller.TOKEN\_IN),  (receptionist.REP\_OUT, comptroller.REP\_PAT\_OUT),  (dentistN.PAT\_OUT, comptroller.SERVED\_PAT) }  EOL = {(dentistN.PAT\_OUT, DCM.PAT\_EXIT), (comptroller.E\_P\_C, DCM.E\_P\_C),  (comptroller.W\_T\_C, DCM.W\_T\_C), (comptroller.E\_P\_Q, DCM.E\_P\_Q),  (comptroller.W\_T\_Q, DCM.W\_T\_Q)}.  EIL = {(DCM.COMPUTE, comptroller.COMPUTE)}  Select: (tokengenerator, receptionist, {Dentist\_1, Dentist\_2, Dentist\_3, Dentist\_4, Dentist\_5}, comptroller) = tokengenerator  Select: (receptionist, {Dentist\_1, Dentist\_2, Dentist\_3, Dentist\_4, Dentist\_5}, comptroller) = receptionist  Select: ({Dentist\_1, Dentist\_2, Dentist\_3, Dentist\_4, Dentist\_5}) = any Dentist  Select: (comptroller, any module) = any other module |

#### **Note:** The proposed clinic model works on Poisson Distribution i.e. the events occur with known constant rate and independently of time since the last event occur.

#### **5.3 Test Cases for TOKEN GENERATOR MODEL**

To generate the tokens for patients such that;

1. patient token follows signifies the arrival rate of patients (poison distribution) in clinic, and
2. the long-term mean should be closed for provided distribution mean

**Note:** *The simulation is made to run for the specific period of time (4 hours).*

|  |
| --- |
| Generate testing related files: (a) tokengeneratorMA.ma   1. tokengeneratorMAOUT.out   **Note:** *There is no .ev file as there is no input to token generator required for simulation* |

#### **5.4 Test Cases for RECEPTIONIST MODEL**

1. To test that the patient is being forwarded to the dentist for treatment by the receptionist upon call.
2. To test the queue receiving a patient call (request) from dentist in the sequential order.

|  |
| --- |
| Cashier testing related files: (a) receptionistMA.ma   1. receptionist.ev   **Note:** *The output of the simulation can be observed and analyzed in the console window in Eclipse.* |

#### **5.5 Test Cases for DENTIST MODEL:**

#### **Note: The five dentist models have been coupled into one model.**

1. To test the dentist receiving protocol: It should first receive ID on DOC\_RDY followed by a value on NXT\_PAT port.
2. To check if the dentist receives a value on the NXT\_PAT without receiving anything on the DOC\_RDY port
3. To test the service time for different number of doctors in the clinic having different service rate following the poison distribution.

|  |
| --- |
| Counter testing files: (a) doctorMA.ma   1. doctor.ev 2. doctorMAOUT.out |

#### **5.6 Test Cases for COMPTROLLER MODEL:**

1. To feed the comptroller model with the inputs: SERVED\_PAT, REP\_PAT\_OUT, COMPUTE and TOKEN\_IN and to verify the output values of W\_T\_C, W\_T\_Q, E\_P\_Q and E\_P\_C.
2. The above values will be sent when the clinic model will receive the value at its COMPUTE port.

|  |
| --- |
| Test files: (a) comptrollerMA.ma   1. comptroller.ev 2. comptrollerMAOUT.out.   **Note:** *The standard unit of time for W\_T\_C and W\_T\_Q is hours.* |

# 6. Model Testing Strategy/Methodology

The testing has been carried out for each atomic/coupled model as per the test models/cases discussed above in the following sequential order:

1. Token Generator Model (Atomic Model)
2. Receptionist Model (Atomic Model)
3. Comptroller Model (Atomic Model)
4. Doctor Model (Atomic Model)
5. Coupled model comprising of tokengenerator, receptionist and doctor 1
6. Coupled model comprising of tokengenerator, receptionist, doctor 1 and doctor 2
7. Coupled model comprising of tokengenerator, receptionist, doctor 1, doctor 2 and doctor 3
8. Coupled model comprising of tokengenerator, receptionist, doctor 1, doctor 2, doctor 3 and doctor 4
9. Coupled model comprising of tokengenerator, receptionist, doctor 1, doctor 2, doctor 3, doctor 4 and doctor 5 (with same average service time)
10. Healthy Smile Dental Clinic Model (model with different service time for different doctors)

Table 2 – Various .ma and .ev files required for simulating the models and validating test cases

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **TEST MODEL** | **.ma file** | **.ev file** |
| 1 | Token Generator Model | tokengeneratorMA.ma | tokengenerator.ev |
| 2 | Receptionist Model | receptionistMA.ma | receptionist.ev |
| 3 | Comptroller Model | comptrollerMA.ma | comptroller.ev |
| 3 | Doctor model | doctorMA.ma | doctor.ev |
| 4 | Doctor 1 Model | doctor1MA.ma | doctor1.ev |
| 5 | Doctor 2 Model | doctor2MA.ma | doctor2.ev |
| 6 | Doctor 3 Model | doctor3MA.ma | doctor3.ev |
| 7 | Doctor 4 Model | doctor4MA.ma | doctor4.ev |
| 8 | Doctor 5 Model | doctor5MA.ma | doctor5.ev |
| 9 | Healthy Smile Dental Clinic Model | dentalclinicMA.ma | dentalclinic.ev |

**6.1 Steps for testing the models:**

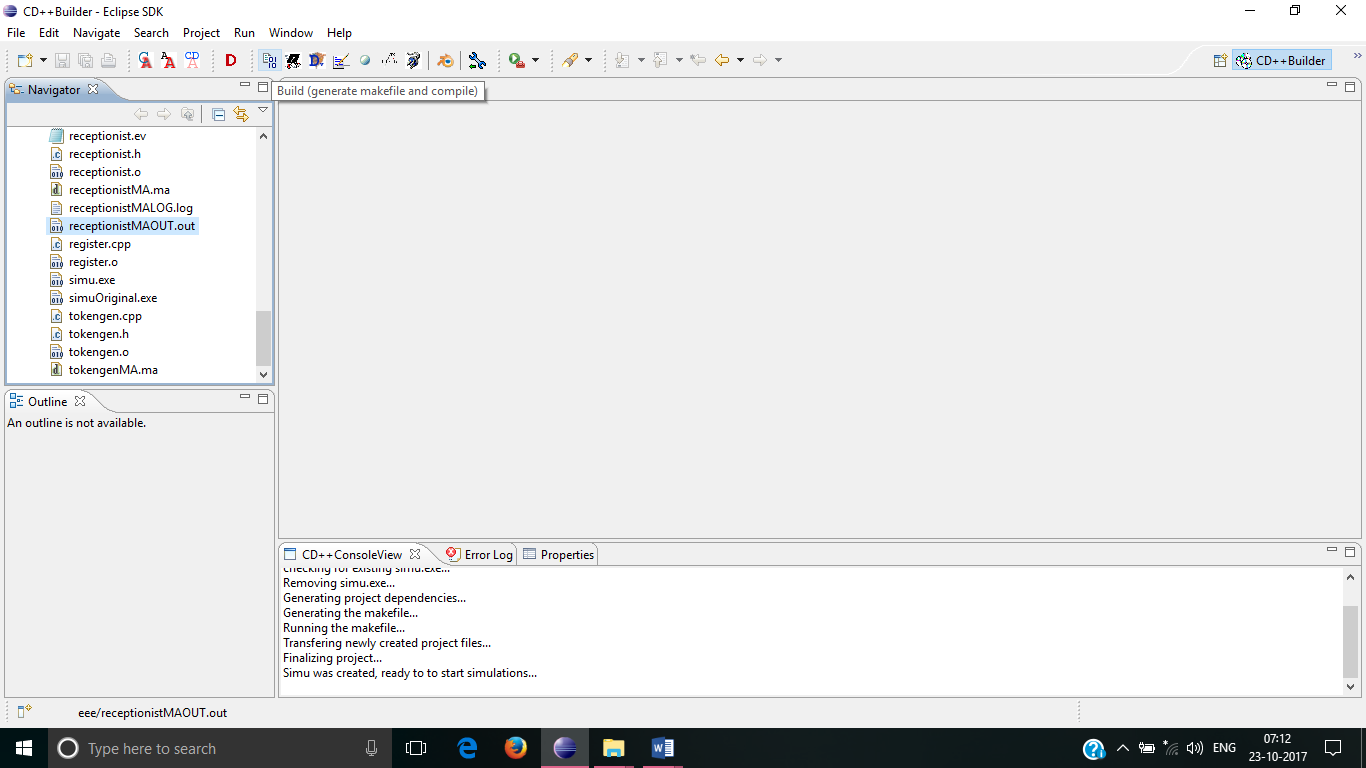
1. Firstly, build and compile the project in Eclipse 3.6 as shown in Figure 2 below

Figure 2 – Building the project in Eclipse 3.6

1. Click on *Simu! Run a simulation for coupled model* and the below window appears for simulating the model.

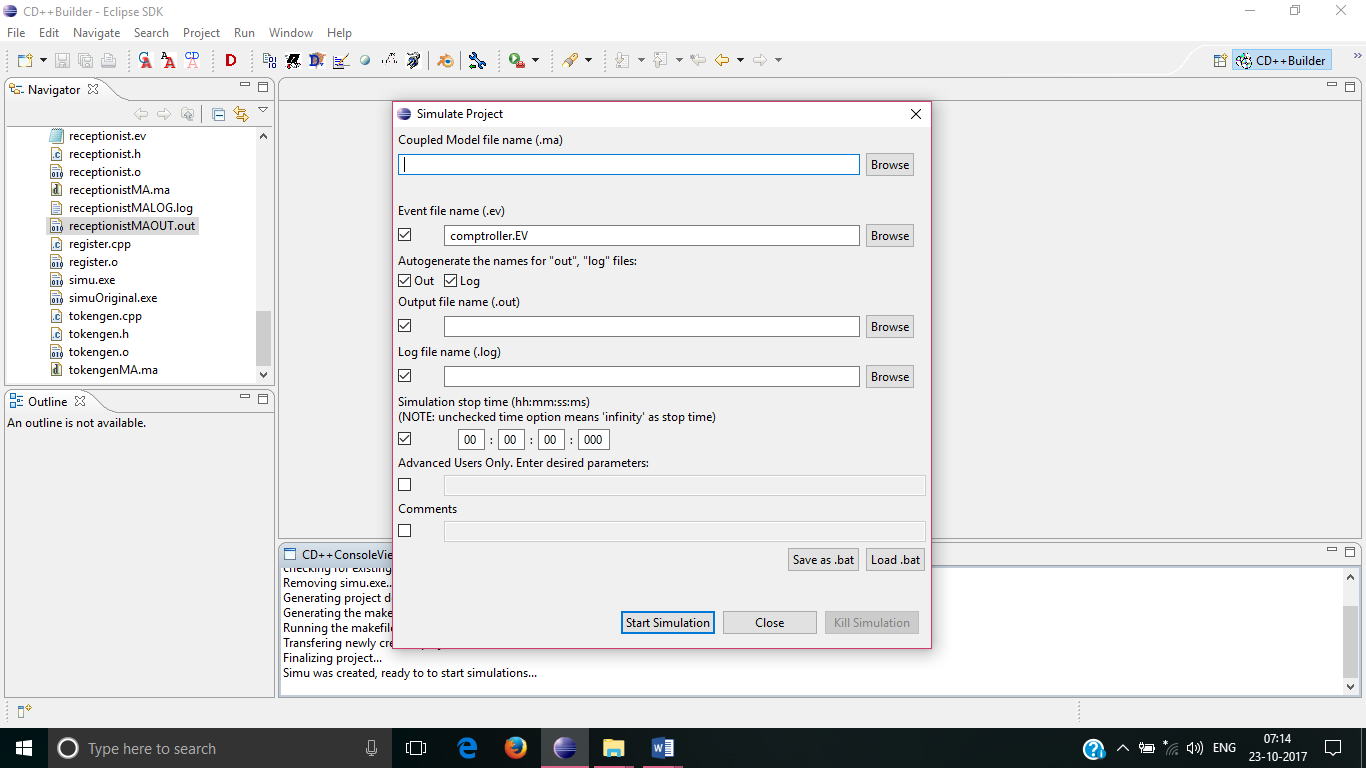


Figure 3 – Simulation window in Eclipse 3.6

1. Select the respective *Coupled Model File (.ma)* and *Event File (.ev)* and tick the Out and Log boxes to generate .log and .out files for generating the output files. Set the Simulation stop time (4 hours in clinic model testing) and click *Start Simulation* to simulate the respective model.

The model output can be analysed from the .log and .out files of the respective models generated during the simulation and the various test cases that have been generated for testing the models specified in .ev files of the respective models. The structure and links/connections of the models have been specified in .ma files of the respective models. The various models and the respective .ma and. ev files to be used for simulating the respective models have been tabulated in Table 2 below:

**PART - III**

# 7. Simulation Analysis with the various examples:

Assume:

λ = Mean Patient Arrival Rate (patients per hour)

µ = Mean Doctor Processing rate (patients per hour)

p = probability

The various test cases that has been used in the testing of the model(s) has been tabulated below:

Table 3 – Test cases for the proposed models

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Test Model** | **Mean Patient Arrival Rate**  **(Patients per hour)** | **Mean Patient Service Rate**  **(Patients per hour)** |
| 1 | Token Generator Test Model | 1 patients per hour | Not applicable (NA) |
| 2 | Receptionist Model  (The values have been calculated as per specifications in event file) | 6 patients per hour | 3 patients per hour |
| 3 | Comptroller Model  (The values have been calculated as per specifications in event file) | 6 patients per hour | 3 patients per hour |
| 3 | Doctor model | Random  (defined in .ev file) | Random  (defined in .ev file) |
| 4 | Doctor 1 Model | 4 patients per hour | 3 patients per hour |
| 5 | Doctor 2 Model | 6 patients per hour | 4 patients per hour |
| 6 | Doctor 3 Model | 10 patients per hour | 3 patients per hour |
| 7 | Doctor 4 Model | 15 patients per hour | 3 patients per hour |
| 8 | Doctor 5 Model | 20 patients per hour | 3 patients per hour |
| 9 | Healthy Smile Dental Clinic Model | 20 patients per hour | Specified in Table 4 |

## Token Generator Model Test (Atomic Model Test)

The token generator assigns the token to the patients and inputs the patients into the system through receptionist. The token generator does not need any. ev file to test as it does not have any input.

Test the tokengenerator by simulating the model with no event (.ev) file for four hours. The following output can be analysed in tokengeneratorMAOUT.out file:

01:02:39:000 out 1

02:31:39:000 out 2

03:47:30:000 out 3

## Receptionist Model Test (Atomic Model Test)

The receptionist model has been tested by generating an event file named receptionist.ev and using it for simulating the receptionist model. The event file has been defined for various input events as mentioned below:

00:10:00:100 in 1 // Patient 1 reports receptionist

00:20:00:200 in 2 // Patient 2 reports receptionist

00:30:00:300 in 3 // Patient 3 reports receptionist

00:20:00:500 nxtpatcall 1 // Call for patient 1

00:40:00:501 nxtpatcall 2 // Call for patient 2

01:00:00:600 nxtpatcall 3 // Call for patient 3

01:20:00:801 nxtpatcall 4 // Call for patient 4

The following output can be analysed in the console view in Eclipse (instead of .out file):

The receptionist can be connected with 5 doctors

00:10:00:100 / in / 1.00000

00:20:00:200 / in / 2.00000

00:30:00:300 / in / 3.00000

00:20:00:500 / nxtpatcall / 1.00000

00:40:00:501 / nxtpatcall / 2.00000

01:00:00:600 / nxtpatcall / 3.00000

01:20:00:801 / nxtpatcall / 4.00000

Simulation ended!

## Comptroller Model Test (Atomic Model Test)

The comptroller model has been tested by generating an event file named comptroller.ev and using it for simulating the comptroller model. The event file has been defined for various input events as mentioned below:

00:20:00:000 enter\_sys 1

00:22:00:100 exit\_q 1

00:30:11:000 enter\_sys 2

00:30:12:000 exit\_q 2

00:40:13:000 enter\_sys 3

01:00:14:500 exit\_q 3

00:42:20:500 exit\_sys 1

01:00:21:500 exit\_sys 2

01:18:20:500 exit\_sys 3

04:00:00:000 in 1

The following output can be analyzed in the comptollerMAOUT.out file in workspace in Eclipse:

04:00:00:000 epc 1.28571

04:00:00:000 epq 0.428571

04:00:00:000 wtq 5.55556e-05

04:00:00:000 wtc 0.000138889

## Doctor Model Test

The sample doctor model has been generated and tested by generating an event file named doctor.ev. The event file has been defined for various input events as mentioned below:

00:10:00:000 rdy 3

00:12:00:000 in 3

00:20:00:000 rdy 4

00:23:00:000 in 4

00:30:00:000 rdy 5

00:33:00:000 in 5

The following output can be analyzed in the doctorMAOUT.out file in workspace in Eclipse:

00:53:00:000 out 5

00:53:00:000 nxtpatcall 5

## Doctor1 Model Test

## (Coupled Model consisting of tokengenerator, receptionist and one doctor)

Assume Doctor 1has the following specifications:

* Assume single doctor and single queue (constant distribution)
* Assume λ = 4 patients per hour
* Assume µ = 3 patients per hour

**Simulation Method:**

The tokengenerator is configured for Poisson distribution with mean value of 4 patients per hour (0.00083 patients per second) and the doctor is configured for constant distribution with mean of 3 patients per hour (0.00138 patients per second).

After running the simulation (using doctor1MA.ma. and doctor1.ev files) for **4 hours**, the following results were obtained in doctor1MAOUT.out:

**Simulation Result:**

04:00:00:000 epc 6.5

04:00:00:000 epq 5.40541

04:00:00:000 wtq 0.416737

04:00:00:000 wtc 0.454622

## Doctor2 Model Test

Doctor2 model consists of two doctors (dentists) in the clinic with the following parameters:

* Assume two doctors and single patient queue
* Assume λ = 6 patients per hour
* Assume µ = 4 patients per hour

After running the simulation for **4 hours**, the following results were obtained in doctor2MAOUT.out

04:00:00:000 epc 3.22727

04:00:00:000 epq 1.71739

04:00:00:000 wtq 2.36404e-05

04:00:00:000 wtc 2.18954e-05

## Doctor3 Model Test

The Doctor3 model consists of three doctors (dentists) in the clinic with the following parameters:

* Assume three doctors and single patient queue (Poisson distribution)
* Assume λ = 10 patients per hour
* Assume µ = 3 patients per hour

**Simulation Method:**

After running the simulation for **4 hours**, the following results were obtained in doctor3MAOUT.out

04:00:00:000 epc 10.4935

04:00:00:000 epq 7.575

04:00:00:000 wtq 0.100032

04:00:00:000 wtc 0.259276

## Doctor4 Model Simulation

The Doctor4 model consists of four doctors (dentists) in the clinic with the following parameters:

* Assume four doctors and single patient queue (Poisson distribution)
* Assume λ = 15 patients per hour
* Assume µ = 3 patients per hour

**Simulation Method:**

After running the simulation for **4 hours**, the following results were obtained in doctor4MAOUT.out

04:00:00:000 epc 13.3333

04:00:00:000 epq 9.3211

04:00:00:000 wtq 0.0250788

04:00:00:000 wtc 0.250107

## Doctor5 Model Test

The Doctor5 model consists of five doctors (dentists) in the clinic with the following parameters:

* Assume five doctors and single patient queue (Poisson distribution)
* Assume λ = 20 patients per hour
* Assume µ = 3 patients per hour

**Simulation Method:**

After running the simulation for **4 hours**, the following results were obtained in doctor5MAOUT.out

04:00:00:000 epc 10.6835

04:00:00:000 epq 5.625

04:00:00:000 wtq 0.000136403

04:00:00:000 wtc 0.000132237

## Dental Clinic Model with doctors (dentists) having unequal service rate

Assume that the dental clinic model consists of five doctors (dentists) in the clinic and the average number of patients entering the clinic (Arrival rate) is 20 patients/hour.

Assume that the average number of patients per unit time being served by the various dentists follows the following profile as specified in Table 4:

Table 4 – Average processing time per patient by the various dentists

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE OF DENTIST** | **TREATMENT** | **AVERAGE PATIENT PROCESSING RATE**  **(per hour)** | **AVERAGE PATIENT PROCESSING RATE (per seconds)** |
| General Dentist  (Doctor 1 in code) | regular dental  check-up | 6 patients/hour | 0.00167 patients/minute |
| Endodontist  (Doctor 2 in code) | root canal and emergency | 3 patients/hour | 0.00083 patients/minute |
| Periodontist  (Doctor 3 in code) | gum problems | 6 patients/hour | 0.00167 patients/minute |
| Prosthodontist  (Doctor 4 in code) | tooth replacement | 3 patients/hour | 0.00083 patients/minute |
| Pediatric Dentist  (Doctor 5 in code) | children and teen dentist | 6 patients/hour | 0.00167 patients/minute |

After running the simulation for **4 hours**, the following results were obtained in dentalclinicMAOUT.out

04:00:00:000 epc 6.44667

04:00:00:000 epq 2.62581

04:00:00:000 wtq 3.08177e-05

## 04:00:00:000 wtc 9.80766e-05

**8. Conclusion:**

The clinic model has been successfully implemented and the various atomic and coupled models have been successfully implemented, simulated and tested for different test cases discussed above. The various atomic models have been successfully tested by generating the test-benches with the use of event files (.ev). The output .ma and .log files (generated during the simulation) have been used to generate. gam and. gcm files for the various atomic and coupled models, respectively.

**APPENDIX**

A1 - readme.txt

The Healthy Smile Dental Clinic sample model sample has been developed under DEVS - Eclipse 3.6 on a HP Pavilion laptop (Intel® Core(TM)i5-5200CPU @ 2.20GHz) running Windows 10 home.

The CD++ Builder Installer (with Eclipse 3.6) has been downloaded as setup\_Eclipse3.6.exe from the course website

<http://cell-devs.sce.carleton.ca/mediawiki/index.php/Installation> on prior getting the username and password from the Professor Gabriel Weiner.

The Cygwin (setup-x86 64bit.exe) has been installed separately from the following website link:

<https://www.cygwin.com/>

The dentalclinic.bat runs the complete dental clinic coupled model having dentists with different patient serving rate using dentalclinicMA.ma and dentalclinic.ev files. Note that the time of simulation used for testing the model is 4 hours.

The doctor5 model employing the 5 dentists(doctors) in the clinic having equal patient serving rates have been designed, simulated and tested with help of doctor5.bat using doctor5MA.ma and doctor5.ev. Note that the time of simulation used for testing the model is 4 hours.

The doctor4 model employing the 4 dentists(doctors) in the clinic having equal patient serving rates have been designed, simulated and tested with help of doctor4.bat using doctor4MA.ma and doctor4.ev. Note that the time of simulation used for testing the model is 4 hours.

The doctor3 model employing the 3 dentists(doctors) in the clinic having equal patient serving rates have been designed, simulated and tested with help of doctor3.bat using doctor3MA.ma and doctor3.ev. Note that the time of simulation used for testing the model is 4 hours.

The doctor2 model employing the 2 dentists(doctors) in the clinic having equal patient serving rates have been designed and simulated with help of doctor2.bat using doctor2MA.ma and doctor2.ev. Note that the time of simulation used for testing the model is 4 hours.

The doctor1 model employing the 1 dentist(doctor) in the clinic having equal patient serving rates have been designed, simulated and tested with help of doctor1.bat using doctor1MA.ma and doctor1.ev. Note that the time of simulation used for testing the model is 4 hours.

The sample doctor model (atomic model) have been simulated and tested with help of doctorc.bat using doctorMA.ma and doctor.ev. Note that the events to test the doctor model have been directly specified in the respective event file (doctor.ev). Note that the time of simulation used for testing the model is 4 hours.

The receptionist model (atomic model) test has been conducted by using doctor.bat with the help of receptionistMA.ma and the receptionist.ev files. Note that the time of simulation used for testing the model is 4 hours.

The tokengenerator model (atomic model) has been tested using the tokengeneratorMA.ma. Note that the token generator model has no event file required for the simulation. Note that the time of simulation used for testing the model is 4 hours.

The comptroller model (atomic model) has been tested with the help of comptroller.bat file using the comptrollerMA.ma and comptroller.ev files. Note that the time of simulation used for testing the model is 4 hours.

NAMING CONVENTIONS FOR COUPLED MODELS –

The related components have identical name roots, for example, dentalclinic.bat, dentalclinicMA.ma and dentalclinic.ev for dental clinic model.

NAMING CONVENTIONS FOR ATOMIC MODELS –

There are comptroller.cpp, comptroller.h, comptrollerMA.ma and comptroller.cpp file names have been used for naming the atomic component – Comptroller.

OUTPUT FILES

The .bat files have been generated and configured to organize the output information for easier analysis. Each .bat file runs the simulator with its respective.ev and .ma files.

The CD++ console view messages, .log file output and the diagnostic messages from the code are output to componentname.outputs.txt i.e. when we run dentalclinic.bat, we have the respective output of simulation in dentalclinic.outputs.txt for convenient viewing and model analysis.

The last instruction of each .bat file generates the messages showing the loading of respective .ma and .ev files and the respective .out and .log files will be generated in the project workspace on execution of respective bat file.

After the execution of .bat file, press any key to exit the .bat prompt window and finish the execution of .bat file.

Ekambir Sidhu (Student ID 300036992)

Manohar Deep Singh Gill (Student ID 101055138)

**A2 – dentalclinic.output.txt**

--------------------------------------------------------------------------------

“DentalclinicMA.ma”

--------------------------------------------------------------------------------

[top]

components : tokengen@tokengenerator receptionist@receptionist doctor\_0@doctor doctor\_1@doctor doctor\_2@doctor doctor\_3@doctor doctor\_4@doctor comptroller@comptroller

out : out epc epq wtq wtc

in : in

Link : out@tokengen in@receptionist

Link : out@receptionist in@doctor\_0

Link : out@receptionist in@doctor\_1

Link : out@receptionist in@doctor\_2

Link : out@receptionist in@doctor\_3

Link : out@receptionist in@doctor\_4

Link : rdy@receptionist rdy@doctor\_0

Link : rdy@receptionist rdy@doctor\_1

Link : rdy@receptionist rdy@doctor\_2

Link : rdy@receptionist rdy@doctor\_3

Link : rdy@receptionist rdy@doctor\_4

Link : nxtpatcall@doctor\_0 nxtpatcall@receptionist

Link : nxtpatcall@doctor\_1 nxtpatcall@receptionist

Link : nxtpatcall@doctor\_2 nxtpatcall@receptionist

Link : nxtpatcall@doctor\_3 nxtpatcall@receptionist

Link : nxtpatcall@doctor\_4 nxtpatcall@receptionist

Link : out@doctor\_0 out

Link : out@doctor\_1 out

Link : out@doctor\_2 out

Link : out@doctor\_3 out

Link : out@doctor\_4 out

Link : in in@comptroller

Link : epc@comptroller epc

Link : epq@comptroller epq

Link : wtq@comptroller wtq

Link : wtc@comptroller wtc

Link : out@tokengen enter\_sys@comptroller

Link : out@receptionist exit\_q@comptroller

Link : out@doctor\_0 exit\_sys@comptroller

Link : out@doctor\_1 exit\_sys@comptroller

Link : out@doctor\_2 exit\_sys@comptroller

Link : out@doctor\_3 exit\_sys@comptroller

Link : out@doctor\_4 exit\_sys@comptroller

[tokengen]

distribution : poisson

mean : 20

[receptionist]

doctors : 5

[doctor\_0]

myId : 0

distribution : constant

value : 6

[doctor\_1]

myId : 1

distribution : constant

value : 3

[doctor\_2]

myId : 2

distribution : constant

value : 6

[doctor\_3]

myId : 3

distribution : constant

value : 3

[doctor\_4]

myId : 4

distribution : constant

value : 6

---------------------------------------------------------------------------

"dentalclinic runtime messages "

N-CD++: A Tool to Implement n-Dimensional Cell-DEVS models

---------------------------------------------------------------------------

Version 2.1-R.45 Jun-2010. StandAlone with DEVS-Graphs

Daniel Rodriguez, Gabriel Wainer, Amir Barylko, Jorge Beyoglonian

Departamento de Computacion. Facultad de Ciencias Exactas y Naturales.

Universidad de Buenos Aires. Argentina.

Loading models from dentalclinicMA.ma

Loading events from dentalclinic.ev

Message log: dentalclinicMALOG1.log

Output to: dentalclinicMAOUT1.out

Tolerance set to: 1e-08

Configuration to show real numbers: Width = 12 - Precision = 5

Quantum: Not used

Evaluate Debug Mode = OFF

Flat Cell Debug Mode = OFF

Debug Cell Rules Mode = OFF

Temporary File created by Preprocessor = /tmp/t3484.0

Printing parser information = OFF

DEVS-Graphs debug level: 0 (No debug info. Use -g parameter to specify debug level)

Starting simulation. Stop at time: 04:00:00:000

The receptionist can be connected with 5 doctors

04:00:00:000 / in / 1.00000

Simulation ended!

--------------------------------------------------------------------------------

DentalclinicMAOUT.out

--------------------------------------------------------------------------------

00:13:09:473 out 1

00:19:28:419 out 3

00:25:47:365 out 5

00:26:18:946 out 2

00:28:56:838 out 6

00:32:06:311 out 7

00:32:37:892 out 4

00:35:47:365 out 8

00:41:34:730 out 10

00:44:44:203 out 11

00:47:53:676 out 12

00:48:25:257 out 9

00:54:12:622 out 14

00:57:22:095 out 15

01:00:31:568 out 16

01:01:03:149 out 13

01:06:50:514 out 18

01:09:59:987 out 19

01:13:41:041 out 17

01:14:17:129 out 20

01:22:51:413 out 22

01:27:08:555 out 23

01:28:34:271 out 21

01:35:42:839 out 25

01:39:59:981 out 26

01:41:25:697 out 24

01:48:34:265 out 28

01:52:51:407 out 29

01:54:17:123 out 27

02:01:25:691 out 31

02:05:42:833 out 32

02:07:08:549 out 30

02:14:17:117 out 34

02:18:34:259 out 35

02:19:59:975 out 33

02:27:08:543 out 37

02:31:25:685 out 38

02:32:51:401 out 36

02:39:59:969 out 40

02:44:17:111 out 41

02:45:42:827 out 39

02:52:51:395 out 43

02:57:08:537 out 44

02:58:34:253 out 42

03:05:42:821 out 46

03:09:59:963 out 47

03:11:25:679 out 45

03:13:19:963 out 49

03:15:42:821 out 50

03:19:59:963 out 51

03:21:39:963 out 48

03:23:19:963 out 53

03:25:42:821 out 54

03:29:59:963 out 55

03:31:25:679 out 52

03:33:19:963 out 57

03:35:42:821 out 58

03:39:59:963 out 59

03:41:39:963 out 56

03:43:19:963 out 61

03:45:42:821 out 62

03:49:59:963 out 63

03:51:25:679 out 60

03:53:19:963 out 65

03:55:42:821 out 66

03:59:59:963 out 67

04:00:00:000 epc 6.44667

04:00:00:000 epq 2.62581

04:00:00:000 wtq 3.08177e-05

04:00:00:000 wtc 9.80766e-05

--------------------------------------------------------------------