Methodologies for Discrete-Event Modelling and Simulation

#### Assignment1: Basketball Park Simulator

# Shichao Guan

# 6962744

UOttawa

**3v3 Basketball Park**

**--Shichao Guan**

**Conceptual Model**

The entity that is focused on is one 3V3 Street Basketball Park. We are interested in the behavior of the park throughout a basketball game.

The park model is composed of three main different components: Ticket Window, Basketball Court and Audience.

Ticket Window is the entrance to the park, through which audiences and workers can get into the park. Workers will come first to prepare for the game. After they finish preparing, audiences will be able to come in. There will be no fee for workers, bur each Audience should be charged 3 dollars as entrance fee. It will output the total amount of money earned as well as the current audience number.

The Basketball Court component will model how the game goes on. We are mainly focus on the host team’s performance. The host team will have a total attacking round initially. When their current attacking round equals the total attacking round, the game will be over. The host team includes 3 players (Guard, Forward, and Center). Assuming that the grand is the one who controls the pace of the game and its skill decides whether the team is able to find a suitable chance to shot before losing the possession of the ball(lose one attacking round). The forward will mainly focus on shot and its skill plays a dominate role in the ratio of successful shot. The Center, based on its rebound ability, decides whether its team is able to get another attack opportunity if the ball is not in(gain another attacking round). Also, the defend skill of the visit team can reduce the ratio of successful shot of the host team. Game is based on the 24 second rule, which means within 24 seconds, the possession of ball will change between the host team and visit team. That is to say the internal transaction time is 48 seconds for the Basketball Court (the host team is able to get a score during a time period of 48 seconds : the host team attack within 24 seconds, then the visiting team attack as well).With the game going on, the Basketball Court will output the score of the host team’s current attacking round and as well, when game is over, it will output the final score of the host team.

As to the Audience part, audiences have three behaviors: ‘Excited’, if the host team wins 3 scores consecutively in their 3 attacking rounds; ’Angry’, if the host team cannot get any score consecutively in their 10 attacking rounds; else, the behavior is ‘normal’. Some of them will decide to leave the park if they are angry. The total amount of audiences who still want to watch the game will be shown as an output. Another output is the audience’s behavior.

A very brief graph illustrating the relationship of different components is as below:



**Formal Specifications**

The formal specifications <S, X, Y, δint, δext, λ, ta> for the atomic models are defined as follows:

**Audience:**

State:

count1 = 0; **//if count1 >=3, out=1 ,which means audience behavior: exicetd;**

count2 =0; **//if count2>=10, out=0 ,which means audience behavior: angry!**

**//and 10 audiences will decide to leave**

**//if audience behavior is not excited and angry, it will be //normal.then out=2**

audienceNum=0;

S = {passive, active,count1,count2}

X = {in,audienceIn}

count2

audienceIn

Y = {out,audienceOut}

in

in

audienceOut

audienceNum

out

count1

δext (count1, count2, audienceNum ,e, x)

{ case phase

passive:

if x is from audieneIn and x>0{ **//receive audience**

audienceNum=audienceNum+audienceIn;

phase = active;}

if x is from in { **//receive current attacking round score**

if(x=0){

count2++;

count1=0;}

if(x=1){

count1++;

count2=0}

phase = active;}

if(count2>=10){

if(audienceNum<10)

audienceNum=0;

if(audienceNum>=10)

audienceNum= audienceNum-10**;}//if audience behavior is angry, 10 of // them will decide to leave**

}

δint (phrase,sigma, audienceNum){

passivate();

}

λ(active)

{

send audienceNum to the port audienceOut**//output current //audiences who still want to watch**

if(count1>=3)

send 1to the port out **//audience excited!**

if(count2>=10)

send 0 to the port out **//audience angry!**

if(count1<3 && count2<3 &&(count1!=0 || count2!=0)) **//audience normal**

send 2 to the port out

}

**TicketWindow:**

State Variables:

sigma = INFINITY, phase = Passive;

currentQueue1Num = 0; //the current number for audience

currentQueue2Num=0; //the current number for worker

totalQueue1Num=0; // total audienceIn number, if the accumulated audience number //is equal or larger than 500, the TicketWindow will close //because seats are full

totalQueue2Num=0; //total worker number

money=0; //money =3\*audience number

Formal specification:

X = {audienceIn,workerIn }

Y = {moneyOut,audienceOut}

S = {phase, sigma, currentQueue1Num, currentQueue2Num,money }

audienceIn

moneyOut

Sigma

moneye

currentQueue1Num

workerIn

audienceOut

currentQueue2Num

phase

δext (money, currentQueue1Num, currentQueue2Num,e, x)

{ case phase

passive:

if x is from audieneIn and x>0{

totalQueue1Num = audienceIn; **//get input**

currentQueue1Num = totalQueue1Num-1; **//set initial queue num**

money=3; //set initial money

sigma = (0,0,3,0); **//transit to the state of sending moneyout**

phase = active;}

if x is from workerIn and x>0{

totalQueue2Num=workerIn; **//get input**

currentQueue2Num= totalQueue2Num-1; **//set initial queue num**

sigma = (0,0,2,0); **//transit to the state of sending moneyout，faster // than audience because workers do not need to pay**

phase = active;}

}

δint (money, currentQueue1Num, currentQueue2Num,e, x)

{ case phase

active:

if (currentQueue1Num >0)

{ currentQueue1Num--;

sigma= (0,0,3,0);

money=money+3; **//set total moneyOut**

if(money>=1500){

phrase=passive;

sigma=INFINITY;

} **//the park is full now!**

}

If(currentQueue2Num>0)

{ currentQueue2Num--;

sigma=(0,0,2,0);}

else // all rounds are over

{ phase = passive; **//change back to initial passive // state**

sigma = INFINITY;

}

}

λ(active)

{ send money to the port moneyOut**//send total money out**

send audience(money/3)to the port audienceOut **//send audience to the park**

}

**BasketballCourt:**

State Variables:

sigma = INFINITY, phase = Passive;

currentRoundNum = 0; //current attacking round number

totalRoundNum = 0; //total attacking round number

tempPG; //decide if the host team will lose one attack opportunity

tempSF; //influence the ratio of successful shot

tempC; //decides if the team can get another attack opportunity

tempDefence; //influence the ratio of successful shot

score = 0; // 0 means the host team did not get a sore in this round ,1 means //the host team got a score in this round.

totalScore=0; //total score number

Formal specification:

X = {totalRoundNumIn }

Y = {scoreOut,totalOut}

S = {phase, sigma, currentRoundNum, score ,totalScore}

totalOut

totalRoundNumIn

scoreOut

Sigma

score

currentRoundNum

totalRoundNum

totalScore

Phrase

δext (score, currentRoundNum, totalScore,e, x)

/\* totalRoundNumIn ( the input )contains not only the total attacking round number, but other information. The format of the totalRoundNumIn is a 6 or 7 bit number. The left 2 (6bit) or 3bit number indicates the total attacking round number. The fourth number from the right stands for tempPG, and the third number ,the second number , the first number from the right represents tempSF, tempC, tempDefence respectively.

Take number n=5005350 as an example: n/10000 is the total round number; (the fourth number from the right \*20)/100 is the opportunity to lose one attacking round;(the third number from the right \*20-the first number from the right\*10)/100 is the ratio of successful shot;( the second number from the right\*4)/100 is the opportunity to gain another attacking round.\*/

{ case phase

passive:

if x is from totalRoundNumIn and x>0{

totalRoundNum = totalRoundNumIn/10000; **//get input**

tempPG=20\*( int((temp-totalRoundNum\*10000)/1000));

tempSF=20\*( int((temp-totalRoundNum\*10000-tempPG\*50)/100));

=20\*( int((temp-totalRoundNum\*10000-tempPG\*50-tempSF\*5)/10));

tempDefence=20\*((temp-totalRoundNum\*10000-tempPG\*50-tempSF\*5-tempC/2));

currentRoundNum = 1;

if(rand()%100>tempPG){

score=0;

totalScore=0; **//PG made a mistake, the host team lose the** **//attack opportunity**

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

else{

if(rand()%100>tempSF-tempDefence/2){

score=0;

totalScore=0; **//SF did not make a successful shot**

if(rand()%100<tempC/5)

currentRoundNum--; **//C get the rebound , the host team get //another attack opportunity**

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

else{

score=1;

totalScore=1; **//A successful attacking round. The ball is in.**

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

}

}

δint (score ,currentRoundNum,totalScore, e, x)

{ case phase

active:

if (currentRoundNum < totalRoundNum) **//begin next attacking round**

{ currentRoundNum ++ ;

if(rand()%100>tempPG)

{

score=0;

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

else

{

if(rand()%100>tempSF-tempDefence/2)

{

score=0;

if(rand()%100<tempC/5)

currentRoundNum--;

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

else

{

score=1;

totalScore++;

holdIn(active, Time( static\_cast<float>( fabs( distribution().get() ) ) ) );

}

}

else // all rounds are over

{ phase = passive; **//change back to initial passive // state**

sigma = INFINITY;

}

}

λ(active)

{ sendOutput( msg.time(), scoreOut, score ) **; //send current attacking round score**

if(currentRoundNum>= totalRoundNum)

sendOutput( msg.time(), totalOut, totalScore ) **;//send total score of host team in the //end**

}

**Score:**

S = {passive, active}

X = {in}

out

in

in

out

Y = {out}

δint (active) = passive

δext (in, passive) = active

δext (in, active) = active

λ(active)

{ send *in* to port *out* //**extract the final score**

}

ta(passive) = INFINITY

ta(active) = receiving\_time

The formal specifications <X, Y, D, {Mi}, {Ii}, {Zij}, SELECT > for the coupled model Court and Basketball Park are defined as follows:

**Court:**

X = { totalRoundNumIn };

Y = { totalScoreOut, scoreOut };

D = { basketballcourt, score };

I(basketballcourt) = {self, score};

I(score) = self;

Z(basketballcourt) = self; Z(basketballcourt) = score;

Z(score) = self;

SELECT: ({su basketballcourt, score bnet1, subnet2}) = basketballcourt;

**Basketball Park:**

X = { totalRoundNumIn, audienceIn, workerIn };

Y = { audienceOut, moneyOut ,out, totalScoreOut };

D = { audience, ticketwindow, Court };

I(Court) = { audience,self };

I(ticketwindow) = { audience,self };

I(audience) = {self};

Z(Court) = audience; Z(Court) = self;

Z(ticketwindow) = audience; Z(ticketwindow) =self;

Z(audience) = self;

SELECT: ({audience, ticketwindow, Court }) = ticketwindow;

({audience, Court }) = Court;

**Test Strategies**

The atomic models and coupled models will be tested using the “black box” testing method. Test cases are created by adding different combinations of inputs to the event file (*.ev*). The outputs in the output file (*.out*) will be checked to see if the result is expected.

**Atomic Model** **ticketwindow:**

Both inputs of the ticketwindow are supposed to a positive integer. Workers will come first to prepare for the game. Then, the audiences will come. Due to the limitate seats of the park, the total number of audiences should be less than 500. In this case, if the ‘audienceIn’ is larger than 500, after the 500th audience comes in, the ticketwindow will be passive. The ta() of workers is 1 second less than that of audiences because workers can come through the gate without paying , which is quicker.

The *.ev* file is created as follows:

00:00:00:00 workerIn 5

00:20:00:00 audienceIn 501

The following is an example of the output file *.out*. In this output file,the events with **bold** fonts mean that worker is coming in and they are not being charged; as to the event with underline indicates that after the 500th audience comes in, the model is passive.

**00:00:02:000 moneyout 0**

**00:00:02:000 audienceout 0**

**00:00:04:000 moneyout 0**

**00:00:04:000 audienceout 0**

**00:00:06:000 moneyout 0**

**00:00:06:000 audienceout 0**

**00:00:08:000 moneyout 0**

**00:00:08:000 audienceout 0**

**00:00:10:000 moneyout 0**

**00:00:10:000 audienceout 0**

**00:00:12:000 moneyout 0**

**00:00:12:000 audienceout 0**

**00:00:14:000 moneyout 0**

**00:00:14:000 audienceout 0**

**00:00:16:000 moneyout 0**

**00:00:16:000 audienceout 0**

**00:00:18:000 moneyout 0**

**00:00:18:000 audienceout 0**

**00:00:20:000 moneyout 0**

**00:00:20:000 audienceout 0**

**00:00:22:000 moneyout 0**

**00:00:22:000 audienceout 0**

**00:00:24:000 moneyout 0**

**00:00:24:000 audienceout 0**

00:20:03:000 moneyout 3

00:20:03:000 audienceout 1

00:20:06:000 moneyout 6

00:20:06:000 audienceout 2

00:20:09:000 moneyout 9

00:20:09:000 audienceout 3

00:20:12:000 moneyout 12

00:20:12:000 audienceout 4

00:20:15:000 moneyout 15

00:20:15:000 audienceout 5

……

00:44:54:000 moneyout 1494

00:44:54:000 audienceout 498

00:44:57:000 moneyout 1497

00:44:57:000 audienceout 499

00:45:00:000 moneyout 1500

00:45:00:000 audienceout 500

**Atomic Model** **audience:**

One of the input is the audience, that should be an integer. The other input is the current attacking round result which should be ‘0’ or ‘1’. Accordingly, one of the output is the current number of audiences who still want to watch the game ( they may leave if they are angry!) and the audience’s behavior(i.e., 0:angry, 1:excited, 2:normal).

The following is an example of the output file *.ev*. In this output file,20 audiences will come first. Then the game will begin. With game going on, audiences will be excited at time **01:03:51:00, 01:09:51:00** and get angry at 02:04:51:00. Ten audiences will leave at 02:04:51:00.

01:01:51:00 in 1

01:02:51:00 in 1

**01:03:51:00 in 1**

01:04:51:00 in 0

01:05:51:00 in 1

01:06:51:00 in 1

**01:09:51:00 in 1**

01:10:51:00 in 0

01:14:51:00 in 0

01:15:51:00 in 0

01:16:51:00 in 0

01:19:51:00 in 0

02:20:51:00 in 0

02:01:51:00 in 0

02:02:51:00 in 0

02:03:51:00 in 0

02:04:51:00 in 0

02:05:51:00 in 1

02:06:51:00 in 1

The following is an example of the output file *.out*. As it shows, the result is expected.

00:00:01:000 audienceout 30

01:01:51:000 audienceout 30

01:01:51:000 out 2

01:02:51:000 audienceout 30

01:02:51:000 out 2

01:03:51:000 audienceout 30

**01:03:51:000 out 1**

01:04:51:000 audienceout 30

01:04:51:000 out 2

01:05:51:000 audienceout 30

01:05:51:000 out 2

01:06:51:000 audienceout 30

01:06:51:000 out 2

01:09:51:000 audienceout 30

**01:09:51:000 out 1**

01:10:51:000 audienceout 30

01:10:51:000 out 2

01:14:51:000 audienceout 30

01:14:51:000 out 2

01:15:51:000 audienceout 30

01:15:51:000 out 2

01:16:51:000 audienceout 30

01:16:51:000 out 2

01:19:51:000 audienceout 30

01:19:51:000 out 2

02:00:51:000 audienceout 30

02:00:51:000 out 2

02:01:51:000 audienceout 30

02:01:51:000 out 2

02:02:51:000 audienceout 30

02:02:51:000 out 2

02:03:51:000 audienceout 30

02:03:51:000 out 2

02:04:51:000 audienceout 20

02:04:51:000 out 0

02:05:51:000 audienceout 20

02:05:51:000 out 2

02:06:51:000 audienceout 20

02:06:51:000 out 2

**Atomic Model** **basketballcourt:**

The only input of basketballcourt is a 6 or 7 bit number which contains many information.Take number n=5005350 as an example: n/10000 is the total round number; (the fourth number from the right \*20)/100

is the opportunity to lose one attacking round;(the third number from the right \*20-the first number from the right\*10)/100 is the ratio of successful shot;( the second number from the right\*4)/100 is the opportunity to gain another attacking round. The ta() is a random number from 0 second to 48 seconds.

Several *.ev* files will be set to test as follow:

Example 1:

00:00:00:00 totalRoundNumIn 5000350

Result:

……

04:23:41:271 scoreout 0

04:23:56:674 scoreout 0

04:24:33:441 scoreout 0

04:24:46:109 scoreout 0

**04:24:46:109 totalout 0**

This is expected because (the fourth number from the right \*20)/100=0, which means PG misses all attack opportunities.

Example 2:

00:00:00:00 totalRoundNumIn 5005550

Result:

……

04:23:25:207 scoreout 1

04:23:41:271 scoreout 1

04:23:56:674 scoreout 1

04:24:33:441 scoreout 1

**04:24:33:441 totalout 500**

This is expected because: 1>(the fourth number from the right \*20)/100=100. The PG will never make a mistake in this case. 2>(the third number from the right \*20-the first number from the right\*10)/100=100.

The SF ‘s successful shot ratio is 100%.

Example 3:

00:00:00:00 totalRoundNumIn 5004442

Result:

……

04:36:27:376 scoreout 0

04:36:35:049 scoreout 0

04:37:03:069 scoreout 0

04:37:45:071 scoreout 0

**04:37:45:071 totalout 259**

That is an expected result.

**Atomic Model** **Score:**

The only input of Score is the totalScoreIn which is an integer.While receiving the input, it will output the result. A simple example is as below:

.ev:

00:20:00:00 totalScoreIn 60

.out:

00:20:00:000 totalscoreout 60

**Coupled Model Court:**

The input is the same as atomic model basketballcourt. The output is current attacking round result and total score. A simple example is as below:

.ev:

00:00:00:00 totalRoundNumIn 5004442

.out:

……

04:36:27:376 scoreout 0

04:36:35:049 scoreout 0

04:37:03:069 scoreout 0

04:37:45:071 scoreout 1

04:37:52:573 scoreout 0

04:37:52:573 totalscoreout 243

**Coupled Model BasketballPark:**

The inputs are workerIn, totalRoundNumIn, audienceIn.

The outputs are audienceOut moneyOut out totalScoreOut.

3 examples will be set to test:

Example 1:

.ev:

00:00:00:00 workerIn 12

02:00:00:00 totalRoundNumIn 5005500

00:20:00:00 audienceIn 520

.out:

……

02:00:23:707 audienceout 500

02:00:23:707 out 2

02:00:28:600 audienceout 500

02:00:28:600 out 2

02:00:29:612 audienceout 500

02:00:29:612 out 1

……

06:23:56:674 audienceout 500

06:23:56:674 out 1

06:24:33:441 audienceout 500

06:24:33:441 out 1

06:24:33:441 totalscoreout 500

In this case we can see that the host team win all attacking round , so the audience are excited almost all the time except for the first two round (‘Excited’, if the host team wins 3 scores consecutively in their 3 attacking rounds). The result is expected.

Example 2:

.ev:

00:00:00:00 workerIn 12

02:00:00:00 totalRoundNumIn 5000500

00:20:00:00 audienceIn 520

.out:

……

02:00:23:707 out 2

02:00:28:600 audienceout 500

02:00:28:600 out 2

02:00:29:612 audienceout 500

02:00:29:612 out 2

02:00:54:462 audienceout 500

02:00:54:462 out 2

02:00:58:843 audienceout 500

02:00:58:843 out 2

02:01:26:699 audienceout 500

02:01:26:699 out 2

02:02:11:096 audienceout 500

02:02:11:096 out 2

02:02:50:821 audienceout 500

02:02:50:821 out 2

02:03:25:529 audienceout 500

02:03:25:529 out 2

02:03:45:463 audienceout 490

02:03:45:463 out 0

……

02:31:43:910 audienceout 20

02:31:43:910 out 0

02:31:59:035 audienceout 10

02:31:59:035 out 0

02:32:34:378 audienceout 0

……

06:24:33:441 audienceout 0

06:24:33:441 out 0

06:24:33:441 totalscoreout 0

In this case, the host team miss all attacking round, so audiences become angry after the 10th round.

Before the game is over, all audiences are gone. The result is expected.

Example 3:

.ev:

00:00:00:00 workerIn 12

02:00:00:00 totalRoundNumIn 5004442

00:20:00:00 audienceIn 520

.out:

……

06:36:04:320 audienceout 500

06:36:04:320 out 2

06:36:23:054 audienceout 500

06:36:23:054 out 2

06:36:23:054 totalscoreout 262

In this case, the host team is a good one. The players win 262 rounds of total 500 and their audience are all wanting to watch them play and sometimes they are excited.

**Summary:**

Based on the basketball park’s simple behavior, different models are created accordingly. With different data chosen to test the model, the simulator works as expected. In conclusion, the M&S of 3v3 basketball park satisfies the V&V requirement.