**CARLETON UNIVERSITY**

**Department of System and Computer Engineering**

****

**SYSC 5104**

**Methodologies For Discrete Event Modelling**

**And Simulation**

**Instructor:**

**Professor:** **Gabriel A. Wainer**

**Team members:**

**Xiangyu Chen (101199353)**

**Chang Gao (101152288)**

**Assignment One**

**Part I**

**Step 1: Establishing the DEVS model**

After the team discussion about the assignment, we decided to establish the DEVS model of Oreo production system.

- **Manufacture of Oreo Production System Model (our own model)**

The goal of this model is to simulate the whole procedure of Oreo production, which can make the process more efficient. This activity is to evaluate the capacity in identifying real systems that can be specified as DEVS models.

**Step 2: Description of the problem to be solved**

**- Introduction of the Model**

A manufacturing facility assembles three different types of Oreos, named original Oreo, Oreo with Chocolate, and Oreo with raisins. These products consist of one or more component types. There are three different types of components of these three biscuits, named Oreo, Chocolate, and Raisin. Original Oreo contains 2 Oreo biscuits, Oreo with chocolate contains one Oreo biscuit and chocolate, and Oreo with Raisin contains one Oreo biscuit and raisins.

**- Scratch of the Model**

The scratch of the model can be seen in Figure 1, we can see there are raw materials and manufacturing part. There are 4 levels in the scratch. The top level is this Oreo factory, it consists of three subnets including Raw material, Manufacturing and Package. Each component will be introduced in next part.

图示

描述已自动生成

Fig 1. Scratch of the model

**- Description of the Model and each component in the Scratch**

1. Raw materials model: It includes queue system and process models.

- The function of queue is managing materials which are waiting.

- The process is to make sure multiple materials that can be processed at the same time

1. Manufacturing model: This model explains the process of producing Oreo cookies, such as material classification, Biscuits formation and assemble. It contains four submodels: Inspector, Formation, Bake and Assembly.
2. Inspector model: The inspector model will check the components of Oreo and classify the cream and the biscuits materials. The subnets are M\_check model and classification model respectively.
3. Formation model: The biscuits components after being inspected will be sent here to generate biscuits with pattern. The cream is transported to the workshop directly after classification. The Oreo biscuits materials will formulate a complete biscuit and then they will be transmitted to the bake model.
4. Bake model: When bake model receives biscuits from Formation, it will start to bake. Then, the biscuits will be sent to assembly model.
5. Assembly model: This model assembles Oreo.

The brokencheck model is to check whether the biscuits are complete or not. If the biscuits are complete, they will be sent to next model. Otherwise, they will be sent to output for other uses.

After the components being inspected, they are supposed to being assigned to corresponding workstation. The product can be assembled when Oreo biscuits and cream are available.

1. Package model: Productions will be packaged and outputs are generated.

**Part II**

图示

描述已自动生成

Fig 1. Scratch of the model

**Atomic Models:**

From the scratch model, we could see there are 7 atomic models in the Oreo Manufacture System. In this part, we will illustrate the formalism of each atomic model based on the definition of DEVS. From previous coursework, we know that DEVS formalism structure can be concluded as follows:

DEVS = ( X, Y, S, δext, δint, λ, ta )

Where:

X : external input event set

Y : external output event set

S : state set

δint : internal transition function

δext :external transition function

λ : output function

ta : time advance function

**- Queue model**

The function of queue is managing materials which are waiting.

图示

描述已自动生成

Fig 2. Queue model

X = { in , done}

Y = {out }

S = { queuenum, phase }

queuenum∈N

phase = <active, positive>

δext ( s = queuenum, e, x=in ){

case phase

passive:

queuenum = queuenum+1;

phase = active;

active:

if (queuenum >= 4)

phase = passive;

else

queuenum = queuenum+1;

phase=active;

}

δext ( s=queuenum, x=done) {

case phase

active:

if queuenum < =1

phase==passive;

else

phase==active;

queuenum = queuenum -1;

endif

}

λ(active) = out

ta( passive ) = inf

**- Process model**

The process model will deal with the materials which are in the queue.

**DEVS formalism of Proc Model**

Proc = < S, X, Y, δint , δext , λ, ta >

X = {in}

Y = {out}

S = { B, F }

δint(B) = F

δext(F, in) = B

λ(B) = {out}

ta(B) =15

ta(F) = inf

图示

描述已自动生成

Fig 3. Process model

**- M\_check model**

The M\_Check model will check the components of Oreo.

图示

描述已自动生成

Fig 4. M\_Check model

**DEVS formalism of M\_Check Model**

M\_check = < S, X, Y, δint , δext , λ, ta >

X ={M\_in}

Y ={M\_out, not\_M}

S = {check, passive}

δext(M\_in, passive)= check

δint(check)=passive

λ(check)

{

send material from *M\_in* to the port *M\_out* 95%

send others to the port *not\_M* 5%

}

ta(check)= 2

ta(passive)= inf

**- Classification model**

Dividing the materials into cream and biscuits.

图示

描述已自动生成

Fig 5. Classification model

**DEVS formalism of Classification Model**

Classification = < S, X, Y, δint , δext , λ, ta >

X={in}

Y={biscuit, cream }

S={active, passive}

δext(in, passive)= active

δint(active)=passive

λ(active)

{

send material from *in* to the port *biscuit*  50%

send material from *in* to the port *cream* 50%

}

ta(passive)=inf

ta(active)=3

**- Formation model**

The biscuits components after being inspected will be sent here to generate biscuits with pattern.

**图示

描述已自动生成**

Fig 6. Formation model

**DEVS formalism of Formation Model**

Formation = < S, X, Y, δint , δext , λ, ta >

X={in}

Y={out}

S={active, passive}

δext(in, passive)= active

δint(active)=passive

λ(active)=out

ta(passive)=inf

ta(active)=5

**- Bake model**

When bake model receives biscuits(4 biscuits)from Formation, it will start to bake. Then, at least four baked biscuits will be sent to assembly model.

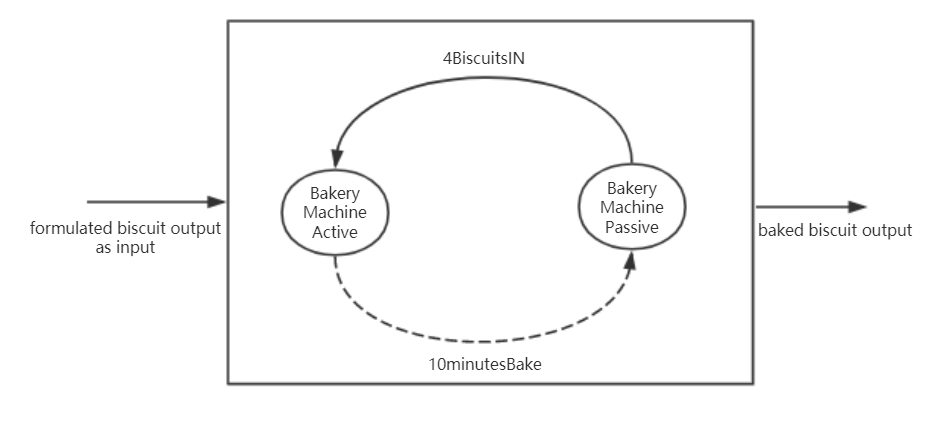


Fig 7. Bake model

**DEVS formalism of Bake Model**

X = { in }

Y ={out }

S = {active, passive}

δext(in, passive)= active

δint(active)=passive

λ(active)= out

ta(passive)=inf

ta(active)=10

- Assembly model

The brokencheck model is to check whether the biscuits are complete or not. If the biscuits are complete, they will be sent to next model. Otherwise, they will be sent to output for other uses.

**- Assembly model**

The brokencheck model is to check whether the biscuits are complete or not. If the biscuits are complete, they will be sent to next model. Otherwise, they will be sent to output for other uses.

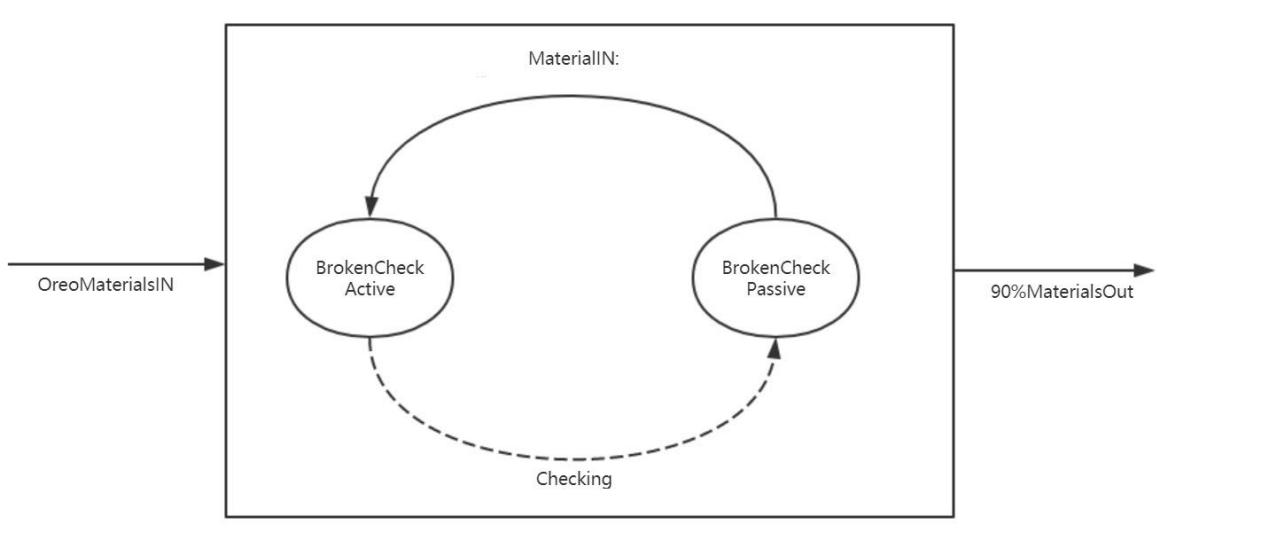


Fig 9. BrokenCheck model

**DEVS formalism of BrokenCheck Model**

X ={in}

Y ={out }

S = {check, passive}

δext(in, passive)= check

δint(check)=passive

λ(check)

{

send material from *in* to the port *out*  90%

send nothing to port *broken 10%*

}

ta(check)= 2

ta(passive)= inf

**DEVS formalism of Workshop Model**

X={in1, in2 }

Y={out}

S={active, passive}

δext(in1, in2, passive)= active

δint(active)=passive

λ(active)=out

ta(passive)=inf

ta(active)=15

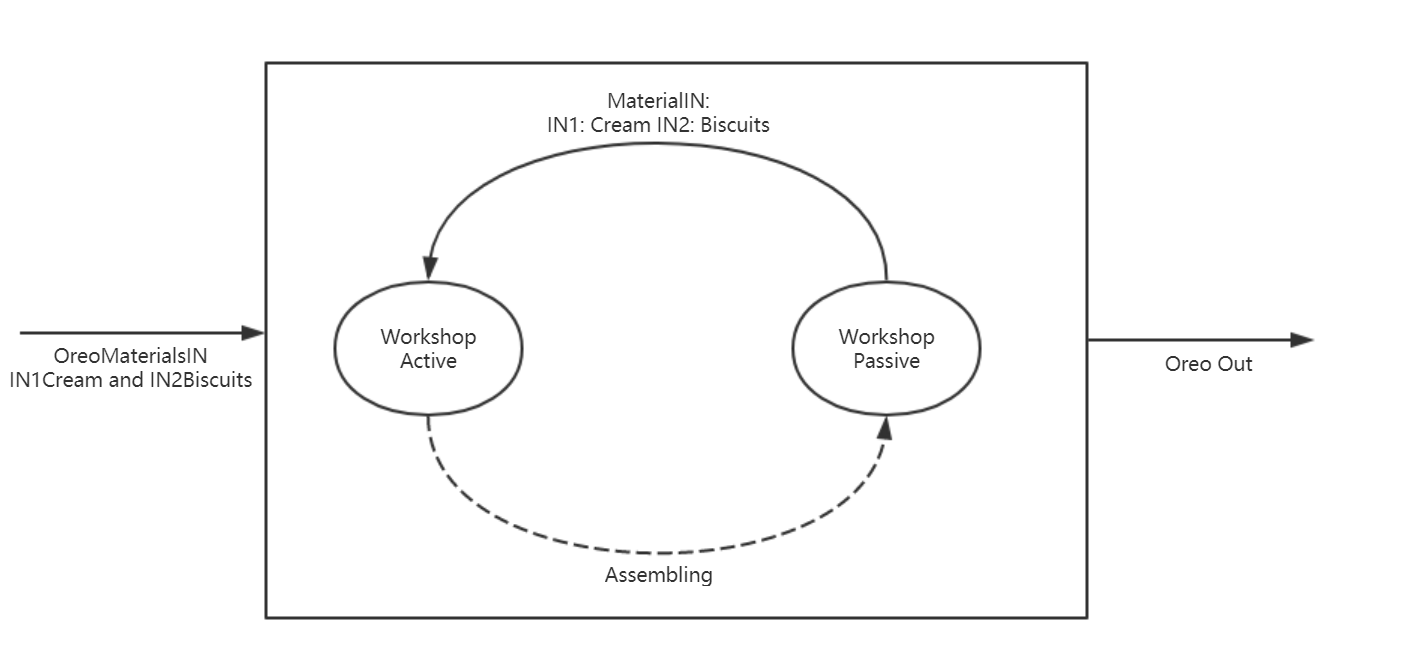
****

Fig 10. Workshop model

**- Package model**

Productions will be packaged and outputs are generated.

**DEVS formalism of Package Model**

X = {O-IN }

Y = {out, done}

S = { Passive, Active }

δext (Passive,O-IN) = Active

δint (Active) = Passive

λ(Active) = {out,done}

ta(Active)=10

ta(Passive) = inf

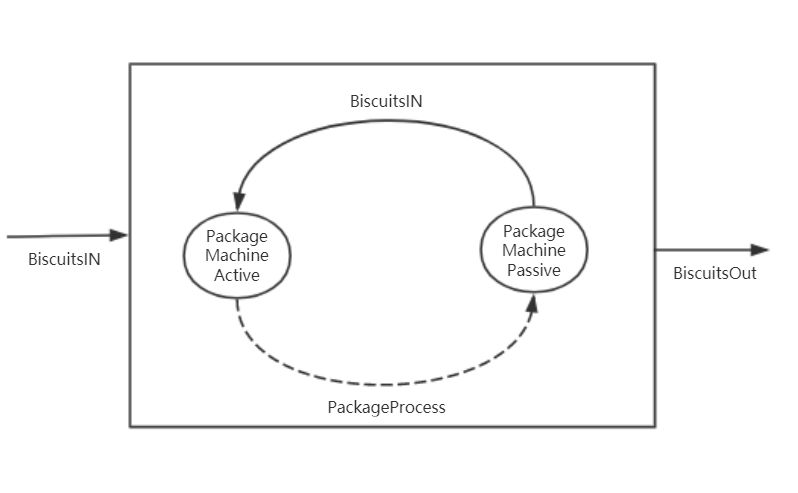


Fig 5. Package model

**Coupled Models:**

**- RawMaterial Model**

X = {In, done}

Y = {Out}

D = {Queue, Process}

EIC = {(RawMaterial.In, Queue.in), (RawMaterial.done, Queue.done) }

EOC = {(Process.out, RawMaterial.Out)}

IC = {(Queue.out, Process.In) }

**- Inspector Model**

X = {in }

Y = {biscuit, cream}

D = {M\_check, Classification}

EIC = {(Inspector.in, M\_check.M\_in) }

EOC = {(Classification.biscuit, Inspector.biscuit), (Classification.cream, Inspector.cream)}

IC = {(M\_check.M\_out, Classification.in)}

**- Assembly Model**

X = {biscuit, cream }

Y = {oreo}

D = {BrokenCheck, Workshop }

EIC = {(Assembly.biscuit, BrokenCheck.in) , (Assembly.cream, Workshop.in2) }

EOC = {(Workshop.out, Assembly.Oreo) }

IC = {(BrokenCheck.out,Workshop.in1) }

**- Manufacturing Model**

X = {in }

Y = { oreo }

D = { Inspector, Formation, Bake, Assembly }

EIC = {(Manufacturing.in, Inspector.in) }

EOC = {(Assembly.oreo, Manufacturing.oreo) }

IC = {(Inspector.biscuit, Formation.in), (Formation.out, Bake.in),

(Bake.out, Assembly.biscuit), (Inspector.cream, Assembly.cream) }

1. **OreoFactory**

X = {in }

Y = {out }

D = {RawMaterial, Manufacturing, Package }

EIC = {(OreoFactory.in, RawMaterial.in) }

EOC = {(Package.out, OreoFactory.out) }

IC = {(RawMaterial.out, Manufacturing.in), (Manufacturing.oreo, Package.O-IN) (Package.done, RawMaterial.done)}

**Experimental strategies:**

After the establishment of the models, we will use the testing cases to verify the atomic models and coupled models. The testing result and procedure are listed below.

**Part III**

For each atomic models, we have illustrated in details in Part one and Part two. In part three, we are going to build the atomic models using the Cadmium. We are expected to finish these steps to ensure the accuracy of the model:

1. Build corresponding experiments for each atomic model. In this way, we will make sure the correctness of atomic models.
2. Test the coupled models based on previous strategy.
3. Build the top model of the Oreo simulation and conduct integration testing.
4. Run the original model that we built, show the reaction of each model.
5. Write the report of the results and make analysis of the execution.

**- Queue model**

The queue model test input is shown here, also you can check the queue\_input\_test.txt in input\_data file. There are six input value are used to test the bake model.

**- The input data:**

00:00:10:00 IN

00:01:00:00 IN

00:02:10:00 IN

00:03:20:00 IN

00:04:00:00 IN

00:05:00:00 DONE

**- Explanation:**

The queue system is designed to manage the materials that are waiting in line. When the system received “IN”, it means there is materials coming. The number of queue will increase, and will be sent to next model after 5 seconds. The maximum of queue\_number is 4. When the system received “DONE”, the number of queue will decrease.

The output results are listed below, they are the same as what we expected. This illustrates the queue model is correct.

**- The output results:**

00:00:10:000

State for model input\_reader\_con is next time: 00:00:50:000

State for model queue1 is totalNum: 1

00:00:15:000

State for model input\_reader\_con is next time: 00:00:50:000

State for model queue1 is totalNum: 1

00:01:00:000

State for model input\_reader\_con is next time: 00:01:10:000

State for model queue1 is totalNum: 2

00:01:05:000

State for model input\_reader\_con is next time: 00:01:10:000

State for model queue1 is totalNum: 2

00:02:10:000

State for model input\_reader\_con is next time: 00:01:10:000

State for model queue1 is totalNum: 3

00:02:15:000

State for model input\_reader\_con is next time: 00:01:10:000

State for model queue1 is totalNum: 3

00:03:20:000

State for model input\_reader\_con is next time: 00:00:40:000

State for model queue1 is totalNum: 4

00:03:25:000

State for model input\_reader\_con is next time: 00:00:40:000

State for model queue1 is totalNum: 4

00:04:00:000

State for model input\_reader\_con is next time: 00:01:00:000

State for model queue1 is totalNum: 4

00:05:00:000

State for model input\_reader\_con is next time: inf

State for model queue1 is totalNum: 3

**- Proc model**

The proc model test input is shown here, also you can check the proc\_input\_test.txt in input\_data file. There are four input value are used to test the proc model.

**- The input data:**

00:00:10:00 IN

00:00:20:00 IN

00:02:10:00 IN

00:03:20:00 IN

**- Explanation:**

The process system is designed to process the materials which are ingredients of Oreo. When the system received “IN”, the model will divide the materials into two parts, and will be sent to next model after 5 seconds. Therefore, the system received “IN” at 00:00:10:00, the totalNum is 2, and it becomes 0 at 00:00:15:00.

The output results are listed below, they are the same as what we expected. This illustrates the queue model is correct.

**- The output results:**

**图片包含 文本

描述已自动生成**

**- Material\_check model**

The Material\_check model test input is shown here, also you can check the mcheck\_input\_test.txt in input\_data file. This model is established to check whether the materials are in the ingredients of Oreo. 95% percent of probability that materials will be sent to next model. The unqualified materials will be sent for other use.

**- The input data:**

00:00:10:00 2

00:00:20:00 10

00:00:40:00 2

- **Explanation:**

At 00:00:10:00, the model received materials. It found that materials are part of Oreo, so that the “Materials” was 2, and the “check\_materials\_out” set 1. At 00:20:00, 10 materials came in, the “Materials” turned to 10.

**- The output result:**

**图片包含 表格

描述已自动生成**

**- Classification model**

The classification model test input is shown here, also you can check the classify\_input\_test.txt in input\_data file. The classification model can help us to classify the materials. The raw materials including the cream and biscuits will be separated. The cream will be sent to the workshop directly. The biscuits materials will be transmitted to the formation model.

**- The input data:**

00:00:10:00 2

00:02:20:00 4

00:04:00:00 6

00:04:20:00 8

00:06:30:00 10

00:08:10:00 15

00:08:50:00 20

**- Explanation:**

The model had 2 materials at 00:10:00, and then it divided into two parts, one for biscuits, another one for cream. After 10 seconds, materials were sent to different models, their numbers became 0.

**- Output results:**

**报纸上的文字

描述已自动生成**

**- Formation model**

The formation model test input is shown here, also you can check the formation\_input\_test.txt in input\_data file. The formation model can is designed to generate the biscuit with logo in specific shape. Then the biscuits will be sent to bake. The cream will be sent to the workshop directly. After the materials arrive in the formation model, the machine will be active. Otherwise, the machine status will be passive with infinite period. The active machine need 10 minutes to generate the biscuits with logo. Hence, we could check the output time of each input to test the accuracy of the model.

**- The input data:**

00:00:10:00 2

00:00:12:00 4

00:04:00:00 6

00:04:20:00 8

00:06:30:00 10

**- Explanation**

Formation model got 2 biscuits at 00:10:00. After 2 seconds, there was another input. Because the time advance is 10 seconds, the model did not receive the second input. 10 seconds later, 2 biscuits were send to next model. It became 0.

**- Output results**

文本

描述已自动生成

**- Bake model**

The bake model test input is shown here, also you can check the bake\_input\_test.txt in input\_data file. There are seven input value are used to test the bake model.

**- The input data:**

00:00:10:00 2

00:00:12:00 4

00:04:00:00 6

00:04:20:00 8

00:06:30:00 10

**- Explanation:** The bake machine will be active after the biscuits transmitted from the formulation part. Otherwise, the machine will be passive with infinite time. When the machine status is active, then the ta(active) of the bake model is 10 minutes. Hence, if the bake model is correct, we will see the time will increase 10 minutes to each input.

The output results are listed below, they are the same as what we expected. This illustrates the bake model is correct.

**- The output results:**

文本

描述已自动生成

**- Results:** From the results, we can see that the input biscuits will be baked 10 minutes, then they became output of the bake model. This shows the bake model is correct.

**- BrokenCheck model**

The brokencheck model is designed to check the qualification of the baked biscuits. If the biscuits are complete and qualified, then it will be assembled to Oreo. 90% of baked biscuits will be transmitted to next model. The input data are listed here, the file name is brokencheck\_input\_test.txt

**- Input data:**

00:00:10:00 2

00:02:10:00 4

00:04:00:00 3

00:04:20:00 8

00:06:30:00 10

**- Explanation:**

The brokencheck is dedicated to check whether the baked biscuits are qualified. If the biscuits are not complete or damaged by accident, the biscuits will not be transmitted to the next model. The bake machine will be active after the biscuits transmitted from the formulation part. Otherwise, the machine will be passive with infinite time. When the machine status is active, then the ta(active) of the bake model is 2 minutes. Hence, if the brokencheck model is correct, we will see the time will increase 2 minutes to each input. The output of the brokencheck model are listed below

**- The output results:**

图片包含 表格

描述已自动生成

**- Results:**

From the results, we can see that the input biscuits will be checked 2 minutes, then they became output of the brokencheck model. The result shows the brokenceheck model is correct.

**- Workshop model**

The workshop model is established to assemble the biscuits and cream to generate an Oreo. If the biscuits and cream both arrive, the workshop will become active and generate an Oreo. After the process, a complete Oreo will become the output of the workshop atomic model. The input data are listed here, the file names are workshop\_biscuit\_input\_test.txt and workshop\_cream\_input\_test.txt.

**- Input data-Biscuits:**

00:01:10 2

00:22:30 4

00:43:45 3

01:06:52 4

01:23:25 5

01:42:35 2

01:52:55 1

**- Input data-Cream:**

00:00:10 2

00:20:30 4

00:45:45 3

01:05:52 4

01:21:25 5

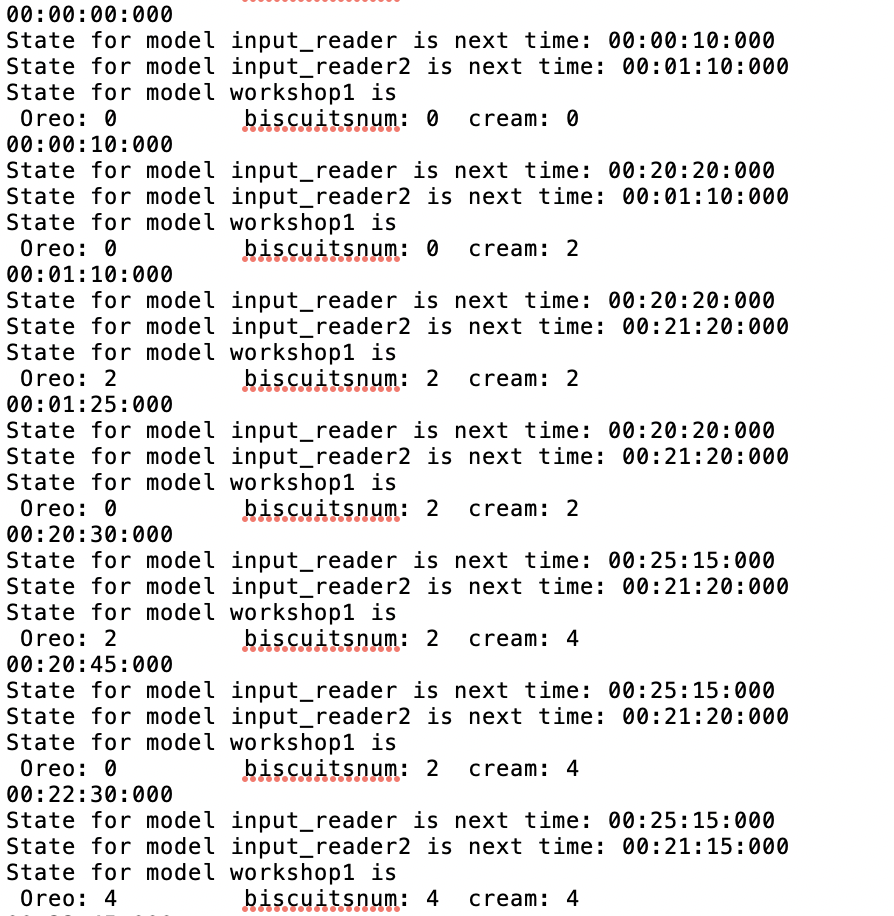
01:41:35 2

01:51:55 1

**- Explanation:**

The cream is transmitted to the workshop after classify model, the biscuits are transmitted from the brokencheck model. The processing time (time advance) is 15 minutes. Hence, if the workshop model is correct, we will see the time will increase 15 minutes to each input. The output of the workshop model are listed below.

**- The output results:**

****

**- Results:**

From the results, we can see that 2 biscuits and cream can formulate a complete Oreo. Meanwhile, the cream arrives earlier than the biscuits since cream is transmitted directly to the Workshop after classification. Therefore, the input time is a little bit different of these two materials. When the biscuits and cream matched, the biscuits will be transmitted to the package model. The test result shows the workshop model is correct.

**- Package model**

The package model test input is shown here, also you can check the package\_input\_test.txt in input\_data file. There are seven input value are used to test the package model.

**- Input data**

00:00:10 1

00:20:30 2

00:45:45 3

01:05:52 4

01:21:25 5

01:41:35 6

01:51:55 7

**- Explanation:**

After the Oreo is made by previous process, the complete Oreo will be sent to package model. The package machine will be active after the completed biscuits transmitted to the package model. Otherwise, the machine will be passive with infinite time. When the machine status is active, then the ta(active) of the package model is 10 minutes. Hence, if the package model is correct, we will see the output time will increase 10 minutes to each corresponding input. The output test result is shown below:

**- Output data:**

**文本

描述已自动生成**

**- Results:**

From the results, we can see that the package time is 10 minutes and all completed Oreos are packaged successfully, so the package model is correct.

**- Top\_model Results Analysis**

The Top model test input is shown here, also you can check the maintop\_input\_test.txt in input\_data file.

**- Input data:**

00:00:10:00 IN

00:00:40:00 IN

00:01:10:00 IN

**- Output data:**

00:00:00:000

**//\*\*\*Initial state, all sub\_states are 0.**

State for model input\_reader\_con is next time: 00:00:10:000

State for model queue1 is totalNum: 0

State for model process1 is totalNum: 0 sendingNum:0

State for model mcheck1 is Materials: 0 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: 0 cream state: 0

State for model formation1 is foramtionNum: 0 formationTimes:0

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 0

State for model package1 is productout: 0 product num: 0

00:00:10:000

**//\*\*\*There is an input “IN”, so the number of queue model becomes 1.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 1

State for model process1 is totalNum: 0 sendingNum:0

State for model mcheck1 is Materials: 0 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: 0 cream state: 0

State for model formation1 is foramtionNum: 0 formationTimes:0

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 0

State for model package1 is productout: 0 product num: 0

00:00:15:000

**//\*\*\*Queue model passes the materials to Process model.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 1

State for model process1 is totalNum: 2 sendingNum:1

State for model mcheck1 is Materials: 0 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: 0 cream state: 0

State for model formation1 is foramtionNum: 0 formationTimes:0

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 0

State for model package1 is productout: 0 product num: 0

00:00:20:000

**//\*\*\*Process model passes the materials to Material check model.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 1

State for model process1 is totalNum: 0 sendingNum:1

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 1

State for model classify1 is biscuit state: 0 cream state: 0

State for model formation1 is foramtionNum: 0 formationTimes:0

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 0

State for model package1 is productout: 0 product num: 0

00:00:30:000

**//\*\*\* Material check model passes the materials to classification model, which will divide materials into biscuits and cream**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 1

State for model process1 is totalNum: 0 sendingNum:1

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 1 cream state: Cream 1

State for model formation1 is foramtionNum: 0 formationTimes:0

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 0

State for model package1 is productout: 0 product num: 0

00:00:40:000

**//\*\*\* The second input comes, the number of queue becomes 2.**

**//\*\*\* classification model sends biscuits to formation model and cream to workshop model.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 2

State for model process1 is totalNum: 0 sendingNum:1

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 0 cream state: Cream 0

State for model formation1 is foramtionNum: 1 formationTimes:1

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 1

State for model package1 is productout: 0 product num: 0

00:00:50:000

**//\*\*\* Formation model sends biscuits to bake model.**

**//\*\*\* Process model got the second input and sends results to material check model.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 2

State for model process1 is totalNum: 0 sendingNum:2

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 1

State for model classify1 is biscuit state: Biscuits 0 cream state: Cream 0

State for model formation1 is foramtionNum: 0 formationTimes:1

State for model bake1 is Bake: 1 & transmitting: 1

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 1

State for model package1 is productout: 0 product num: 0

00:01:00:010

**//\*\*\* bake model sends biscuits to broken check model.**

**//\*\*\*the second input will repeat the process described above.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 2

State for model process1 is totalNum: 0 sendingNum:2

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 1 cream state: Cream 1

State for model formation1 is foramtionNum: 0 formationTimes:1

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 1 Broken: 0 & check\_brocken\_out: 1

State for model workshop1 is Oreo: 0 biscuitsnum: 0 cream: 1

State for model package1 is productout: 0 product num: 0

00:01:05:010

**//\*\*\*Broken check model sends the unbroken biscuits to workshop model.**

State for model input\_reader\_con is next time: 00:00:30:000

State for model queue1 is totalNum: 2

State for model process1 is totalNum: 0 sendingNum:2

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 1 cream state: Cream 1

State for model formation1 is foramtionNum: 0 formationTimes:1

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 1 biscuitsnum: 1 cream: 1

State for model package1 is productout: 0 product num: 0

00:01:10:000

**//\*\*\* The third input comes which will repeat the above process.**

**//\*\*\* Workshop model have assembled Oreo and ready to send to package model.**

State for model input\_reader\_con is next time: inf

State for model queue1 is totalNum: 3

State for model process1 is totalNum: 0 sendingNum:2

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 0 cream state: Cream 0

State for model formation1 is foramtionNum: 1 formationTimes:2

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 1 biscuitsnum: 1 cream: 1

State for model package1 is productout: 0 product num: 0

00:01:25:000

**//\*\*\* The Oreos are sent to package model.**

State for model input\_reader\_con is next time: inf

State for model queue1 is totalNum: 3

State for model process1 is totalNum: 0 sendingNum:3

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 1

State for model classify1 is biscuit state: Biscuits 0 cream state: Cream 0

State for model formation1 is foramtionNum: 0 formationTimes:2

State for model bake1 is Bake: 1 & transmitting: 1

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 1 cream: 1

State for model package1 is productout: 1 product num: 1

00:01:35:000

**//\*\*\*The package model sends the products out and the whole process is done, so it also sends the “DONE” message to queue model.**

**//\*\*\*The number of queue number decreases.**

State for model input\_reader\_con is next time: inf

State for model queue1 is totalNum: 2

State for model process1 is totalNum: 0 sendingNum:3

State for model mcheck1 is Materials: 2 Others: 0 & check\_materils\_out: 0

State for model classify1 is biscuit state: Biscuits 1 cream state: Cream 1

State for model formation1 is foramtionNum: 0 formationTimes:2

State for model bake1 is Bake: 0 & transmitting: 0

State for model brokencheck1 is Biscuit: 0 Broken: 0 & check\_brocken\_out: 0

State for model workshop1 is Oreo: 0 biscuitsnum: 1 cream: 1

State for model package1 is productout: 0 product num: 1

**Acknowledgement**

During the assignment one, our team(ChangGao and XiangyuChen) designed the Oreo Manufacture model ourselves. From the atomic models to the coupled models, lots of efforts are made by ourselves. We had a great time working and studying with each other. I’d say that the whole process is full of challenge since Xiangyu and I have 12 hours jet lag but our communications works well, and we finish the original modelling by ourselves in an efficient way. It’s unforgettable and meaningful for both of us. For assignment One, the workload are assigned in the most reasonable and fair way for both of us.

**Workload:**

**Chang Gao**

- Write the proposal of the Oreo Manufacture Model

- Conclude and Write the Final report of the assignment

- Finish the coding of Bake model; BrokenCheck Model; Workshop Model; Package Model and Coupled Model of Assembly

- Write the ReadMe.Md and txt version

- Write the XML of Oreo Manufacture Model

- Draw the corresponding flow chart of assigned model

**Xiangyu Chen**

- Finish the coding of Queue Model; Proc Model; M\_Check Model; Classification and Formation Model; The Coupled Model of Raw Material Processing and the Top Model

- Debug the whole System and check the model

- Generate the Makefile of the model

- Write the ReadMe.Md of corresponding models

- Draw the corresponding flow chart of assigned model

As you can see, we learn how to cooperate with each other and have a great team work during the process. Meanwhile, we did make efforts in coding and the report presentation. We learned how to design the DEVS model and understood the atomic model structures and how the Cadmium work. The assignment 1 is hard with challenge, but we gained more than knowledge from it. Thanks for your reading and we are looking forward to your suggestions.