Carleton University
Department of Systems and Computer Engineering
SYSC 5104: Methodologies for discrete-event modelling and simulation
Fall Semester 2011
Assignment # 1: FSCM DEVS Model Report

Saman Jafartayari
uOttawa ID: 5453154

Cheryl Anne D'souza
Carleton ID: 100771069

Table of Contents

Conceptual Model

Supply Chain Management (SCM) is the key to having a competitive edge in the current global market. It involves effeciently managing the flow of materials, cash, information and services. In order to evalute various advantages and disadvantages of different SCM models, simulation before implementation is key [1]. Thus, a part of SCM Discrete Event Simulation (DEVS) model was was designed to facilitate the simulation of a system of interconnected businesses that are linked together to provide products and services to the end user. This particular part models the manufacturing process of a particular pharmaceutical product. The model includes four major components: administrator, warehouse, and pharmaceutical manufactuing plant (PMP).

Note that the orginial model that was proposed seemed to be too complicated and beyond the scope of the course requirements. Hence, only a part (factory) of the originial model (SCM) was modeled and simulated.

As seen the block diagram in Figure 1, the role of the factory is to transform the raw material it receives from its suppliers to the final product. Therefore, internally the inventory of raw material and final products should be checked. This is done for a number of reasons:

(1) In order to minimize the lead time involved processing and shipping the raw materials from the supplier;
(2) To ensure the factory is producing products at the current demand level.

The administrative duties such as: receiving good from the supplier, attaining the orders from the distributor, placing orders with the supplier and shipping finished products to the distributor; are modeled by the administrator atomic model. It is comprised of seven input ports: RawMaterials2, OrderInfo2, FacWar_out1, FacWar_out2, Busy1, PMP_out and Send. It should be noted that the administrator always tries to maintain a full capacity of finished products.

i. RawMaterials2 is the port through which the supplier transports raw materials to the factory. In order for the supplies to be transferred from the supplier to the factory there exists variable lead times (due to factors such as transportation delay). However, for the purpose of simplicity, we will model this as a constant time delay of 1 day.

ii. OrderInfo2 is a one-way port that originates from the distributor and terminates at the factory. It is used to place orders to replenish the distributor's inventory. Realistically the delay of information flow can vary from case to case. To avoid complexity we set this delay to a constant value of 12 hours.

iii. FacWar_out1 is the port through which the warehouse sends its raw materials to the administration.

iv. FacWar_out2 is the port through which the warehouse sends its finished products to the administration.

v. Busy1 is the port that is Boolean; it is set to true if the powder atomic model (coupled in the factory's pharmaceutical manufacturing plant) is busy and false otherwise.

vi. PMP_out is the port through which the factory's pharmaceutical manufacturing plant (in particular the packing model) communicates with administration.

vii. Send is the port through which the factory's administrator communicates to the warehouse how many raw materials or finished products it needs.

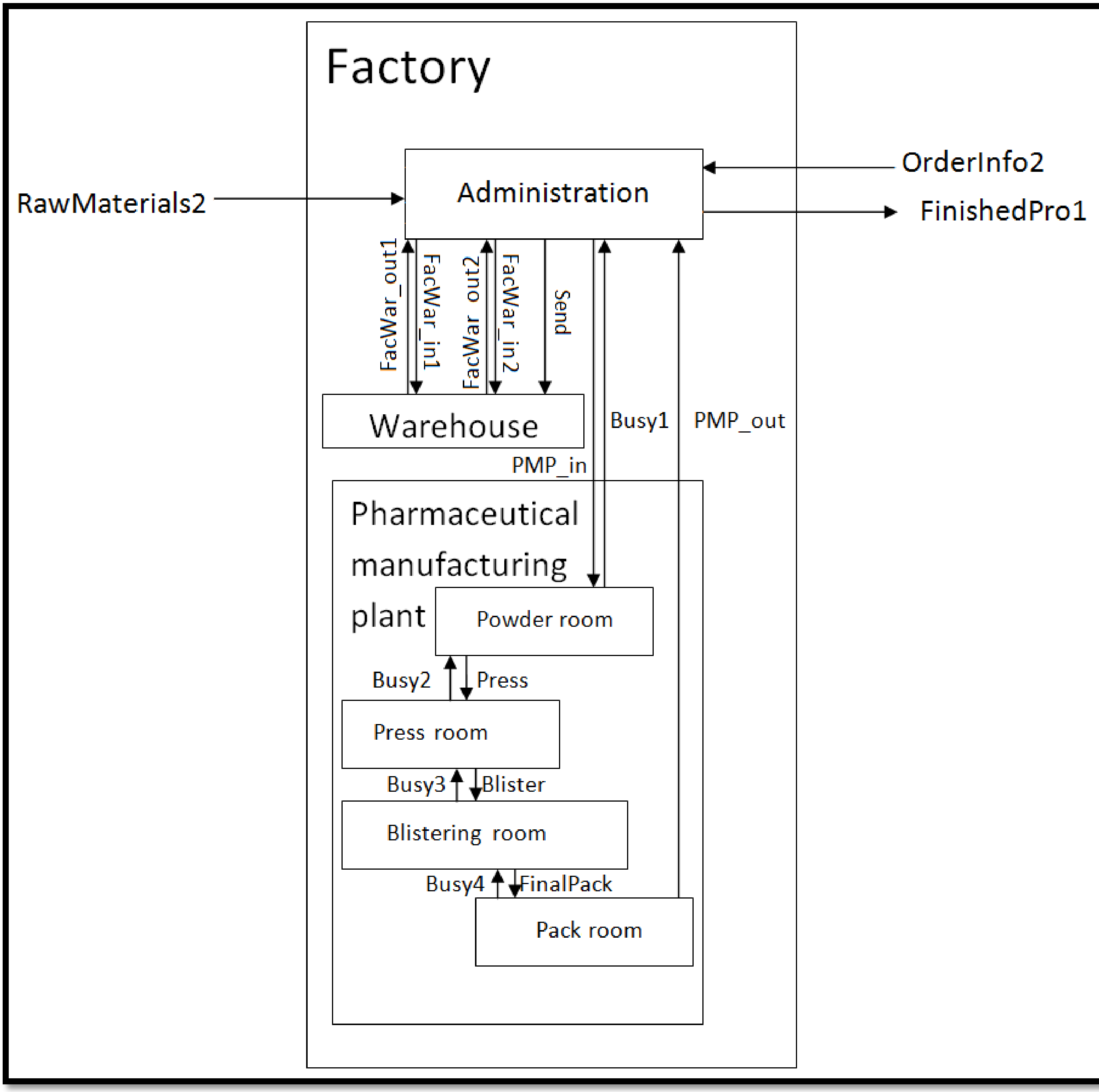

Figure 1: DEVS model block diagram of factory

In addition to input ports, the factory also encompasses four output ports: FinishedPro1, FacWar_in1, FacWar_in2 and PMP_in.

i. FinishedPro1 port is utilized by the factory to ship its finished products to the distributor. The shipment delay for this modeled as a constant value of 1 day.
ii. FacWar_in1 is the port through which the administration transports raw materials to the factory's warehouse.
iii. FacWar_in2 is the port through which the administration transports finished products to the factory's warehouse.
iv. PMP_in is the port through which the administration communicates with the factory's pharmaceutical manufacturing plant (in particular the powder room model).

The role of the factory's warehouse is also to act as a storage facility for the factory's raw materials and finished products. It has a maximum carrying capactiy of 80 batches for raw materials (represented as positive integers ranging from 111 to 180) and 20 batches for finished products (represented as positive integers ranging from 181 to 200). In the event of an overflow of either raw materials and/or finished products, the redundant batches will just be discarded without notification. It recieves raw materials and finished products from the admistrator through the ports FacWar_in1 and FacWar_in2, recpectively. In addition, it sends raw materials and finished products as the admistrator requires it through the ports FacWar_out1 and FacWar_out2, respectively. Lastly, the number or products (both raw materials and finished products) that need to be sent to the administrator is communicated through the port Send.

The role of the pharmanceutical manufacturing plant is to manufacture pills (i.e. convert raw materials into a pill that will then be stored in the warehouse and then sent to the distibutor). This is modeled by four atomic models: powder room, press machines, blistering machines and packaging. In order to model all four tasks, each atomic model is associated with a time delay and does not have internal storage.

Since the whole pharmaceutical manufacturing plant has no internal storage, it is important to have a negative feedback at every stage in the plant. These negative feedbacks are the boolean ports: Busy1, Busy2, Busy3, and Busy4.

The powder room model is comprised of two input ports: PMP_in and Busy2; and two output ports: Busy1 and Press. The powder room recieves its raw materials from the administration through PMP_in and then converts it into a powder with appropriate amounts for each pill. Each batch of powder is created with four batches of raw materials. The time delay associated with creating this is a constant of two hours. The boolean variable, Busy2, indicates if the press machines are busy or not. If this variable is true, indicating that the press machines is indeed busy, the powder room must hold its contents before proceeding. Once this varible is false, indicating that the press machines is free, the powder room sends one batch of material to the press machines through the port Press. The port Busy1 is a boolean variable that indicates if the powder room is busy or not. This is to avoid overflow of materials. If the powder room is busy, the administrator must wait until the powder room (indicated by a false transferred through the port Busy1) is free to send more materials to it.

The press machines room is also modeled as an atomic model that has two input ports: Press and Busy3; and two output ports: Busy2 and Blister. In reality, the role of the press machine is to take various powedered raw materials and apply force to shape them into a pill (tablet). For the sake of simplicity this is modeled as a time delay. When the blistering machines are free, indicated by a the boolean variable Busy2 being false, the powder room sends 1 batch of powder to be pressed. Once the press machine recieves the powder it automatically sets Busy2 to be false, thus avoiding any overflow. This is modeled by a constant time delay of four hours. If the boolean variable Busy3 is false, it sends the pressed products to the blistering machines room through the port Blister. If the boolean variable Busy3 is true, it waits until it is false to send the pressed products.

Blistering is the first of two parts to packing the pills/tablets. For simplicity, this process is modeled by an atomic model that functions a time delay. The blistering atomic model has two inputs and two outputs. The two input ports are Blister and Busy4; and the two output ports are Busy3 and FinalPack. Once the blistering room is free (indicated by a setting the variable Busy3 to true), the blistering room can accept inputs from the press machine room. The blistering room recieves its raw materials from the press machine room through the port Blister. Blitering involves a 1:1 ratio of raw materials: finished products. Once the raw materials are recieved it sets Busy3 to false and then introduces a constant six hour time delay, at which point the materials are sent to the packing room (through the port FinalPack) if it is not busy. The boolean variable Busy4 indicates if the packaging room is busy (true) or free (false).

Packaging is the last atomic model in the pharmaceutical manufactuing plant. It is the second part to packaging the final products. This atomic model has three ports, one input port: FinalPack; and two output ports: Busy4 and PMP_out. This input (Busy4), is boolean variable, it is initially set as false. However, once the blistering model moves blistered products to the packing model, the boolean variable, Busy4, automatically sets itself to true. While the the boolean variable, Busy4, is false, the packing model is in passive mode. This occurs infinetly, until it recieves an input through the port, FinalPack. At this point the boolean varibale, Busy4, is set to false and the packing model is in active mode. It then converts a batch of blister pack to one batch of complete finished and packaged product. Upon completion, it sends the finished product to the administration through the port PMP_out. The packaging room instill a time delay of eight hours in the whole process.

Model Specifications

The factory is modeled as a three-level DEVS model with two components that are described as coupled models and six components that are described as atomic models. The DEVS formal specifications [2] for each model is outlined below starting from atomic models and concluding with the factory itself. This also proves to be the appropriate order for implementation and testing.

Atomic models

Administrator

Administrator = <S, X, Y, δ_{ext}, δ_{int}, λ , ta>

Where

InPorts = {"OrderInfo2", "RawMaterials2", "FacWar_out1", "FacWar_out2"}, where $X_{OrderInfo2}$ = {1,2,3,...,18,19,20}, $X_{RawMaterials2}$ = {1,2,3,...,58,59,60}, $X_{FacWarout1}$ = {111, 112, 113, ..., 178, 179, 180}, $X_{FacWarout2}$ = {181,182,183, ... , 197, 198, 200}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = {" FinishedPro1","FacWar_in1", "FacWar_in2", "Send"}, where $Y_{FinishedPro1}$ = {1,2,3,...,18,19,20}, Y_{FacWar_in1} = {111, 112, 113, ..., 178, 179, 180}, Y_{FacWar_in2} = {181,182,183, ... , 197, 198, 200}, Y_{Send} = {100, 102, 103, ..., 198, 199, 200}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of output ports and values

S = {phase, sigma, currentship, current_finished, warehouseFinishRequest, warehouseFinishQuantity, current_raw, rawRecieved, warehouseRaw, rawMaterial, readyToPMP, finishedProduct, readyToShip, powderStat, sent, inProgress, storefinishedProduct, shipToWarehouse, requestRaw, warehouseRawQuantity}

= {"passive", "active"} × R_0^+ × {0,1,2,...,18,19,20} × {0,1,2,...,18,19,20} × {true, false} × {181,182,183,...,198,199,200} × {0,1,2,...,78,79,80} × {0,1,2,...,78,79,80} × {true, false} × {0,1,2,3,4} × {true, false} × {0,1,2,...,18,19,20} × {true, false} ×

{true, false} × {true, false} × {0,1,2,...,18,19,20} × {101,102,103,...,178,179,180} × {true, false} × {true, false} × {104}

δ $_{ext}$**(phase, sigma, currentship, current_finished, warehouseFinishRequest, warehouseFinishQuantity, current_raw, rawRecieved, warehouseRaw, rawMaterial, readyToPMP, finishedProduct, readyToShip, powderStat, sent, inProgress, storefinishedProduct, shipToWarehouse, e)**{

```
if(msg.port() == OrderInfo2){
        if(msg.value() <=0){
            Display error message! Invalid order!
        }else if((msg.value() >= 1) && (msg.value() <= 20)){
            if (currentship+msg.value()>current_finished){
                Display error message! Can't ask for more than
available!
            }else{
                currentship= currentship+msg.value();
                warehouseFinishRequest=true;
                warehouseFinishQuantity=currentship+180;
                current_finished=current_finished-currentship;
            }
        } else {
            Display error message! No order can be greater than 20
            units, distributor should never request more.
        }
    }

    if (msg.port()== RawMaterials2){
        if (msg.value()<=0){
```

```
                Display Error Message because Supplier should not
                negative or zero input to factory!
        }else if (msg.value()>(80-current_raw)){
            Display Error Message because Supplier should not send more
            raw materials than capacity dictates
        }else if (msg.value()>=0 && msg.value()<=(80-current_raw)){
            current_raw=current_raw+ msg.value();
            rawRecieved= msg.value()+100;
            warehouseRaw=true;
        }
    }

    if (msg.port()== FacWar_out1){
        if (msg.value()!=4){
            Display Error Message because powder room can never
            anything other than an input of 4 raw material units
        }else if(msg.value () ==4){
            rawMaterial=msg.value();
            readyToPMP=true;
        }
    }

    if (msg.port()== FacWar_out2){
        if (msg.value()>20){
            Display Error Message because total capacity for warehouse is
            only 20 so this should never happen
        }else if(msg.value () <=20 && msg.value()>0){
            finishedProduct=msg.value();
            readyToShip=true;
        }else{
```

```
                        Display Error Message because simulation should never reach
this
                }
        }

        if(msg.port()==Busy1){
                if(msg.value()==1){
                        powderStat=true;
                }else if (msg.value()==0){
                        powderStat=false;
                        sent=false;
                }else{
                        cout<<"Error: Port Busy1 should never receive an input other
                        than 0 or 1!"<<endl;
                }
        }

        if(msg.port()==PMP_out){
                if (msg.value()!=1){
                        Display Error Message because PMP_out should only output a
value of 1
                }else if(msg.value()==1){
                        int temp=msg.value();
                        if (current_finished>=20){
                                Display Error Message because Simulation should never
get here!
                        }
                        inProgress=inProgress-1;
                        storefinishedProduct=temp+180;
                        current_finished=current_finished+1;
                        shipToWarehouse=true;
```

```
            }
        }
    }
```

δ_{int}(phase, sigma, current_finished, inProgress, current_raw, powderStat, sent, requestRaw, warehouseRawQuantity){

```
    if(active){
            if((20-current_finished-inProgress)>0 && (20-current_finished-
    inProgress)<=20 && current_raw>=4 && !powderStat && !sent){
                    requestRaw=true;
                    warehouseRawQuantity=4+100;
                    current_raw=current_raw-4;
            }else{
                    passivate();
            }
        } else {
            //this will never happen
            if(passive){
                    Display Error Message!
            }
        }
    }
```

λ (phase, sigma, warehouseFinishRequest, requestRaw, sent, readyToPMP, powderStat, inProgress, readyToShip, shipToWarehouse){

```
        if(warehouseFinishRequest){
                    warehouseFinishRequest=false;
                    send output "warehouseFinishQuantity" to send
            }
            if(requestRaw){
                    requestRaw=false;
                    sent=true;
```

```
                send output "warehouseFinishQuantity" to send
        }
        if(warehouseRaw){
                warehouseRaw=false;
                send output "current_raw" to FacWar_in1
        }
        if(readyToPMP && !powderStat &&
((inProgress+current_finished)<20)){
                readyToPMP=false;
                powderStat=true;
                inProgress=inProgress+1;
                 send output "rawMaterial" to PMP_in
        }
        if(readyToShip){
                readyToShip=false;
                send output "finishedProduct" to FinishedPro1
        }
        if(shipToWarehouse){
                shipToWarehouse=false;
                send output "storefinishedProduct" to FacWar_in2
        }
 }
ta("passive") = ∞;
ta("active") = (0,0,1,0);
```

Testing Strategy

1. Verify the effects of overflow – if the shipment will incur overflow, the shipment must not be accepted and an error message should be displayed.
2. Verify the effects non-positive - error message should be displayed.
3. Verify the current_finished is always at max capacity, if it is not request raw materials from administrator and send to PMP_in.
4. Verify the raw_current is decreased by finished_units when there is an input to FacWar_out1.
5. Verify the finished_current is decreased by finished_units when there is an input to FacWar_out2.
6. Verify the raw_current is increased by raw_units when there is an output from FacWar_out1.
7. Verify the finished_current is increased by finished_units when there is an input from FacWar_out2.

Warehouse

Warehouse = <S, X, Y, δ_{ext}, δ_{int}, λ, ta>

Where

InPorts = {“FacWar_in1”, “FacWar_in2”, “Send”}, where X_{FacWar_in1} = {111, 112, 113, ..., 178, 179, 180}, X_{FacWar_in2} = {181,182,183, ... , 197, 198, 200}, Y_{Send} = {100, 102, 103, ..., 198, 199, 200}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = {“FacWar_out1”, “FacWar_out2”}, where $Y_{FacWarout1}$ = {111, 112, 113, ..., 178, 179, 180}, $Y_{FacWarout2}$ = {181,182,183, ... , 197, 198, 200}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of Output ports and values

S = {phase, sigma, currentRaw, rawMaterialCapacity, currentFinished, finishedProductCapacity,

rawTobesent, sendRaw, finishedTobesent, sendFinished}

= {“passive”, “active”} × R_0^+ × {101, 112, 113, ..., 178, 179, 180} × {180}× {181,182,183, ... , 197, 198, 200} × {200} × {1, 2, 3,...,78, 79, 80} × {true,false} × {1, 2, 3,...,18, 19, 20} × {true,false}

δ_{ext}(phase, sigma, currentRaw, rawMaterialCapacity, currentFinished, finishedProductCapacity, rawTobesent, sendRaw, finishedTobesent, sendFinished, e, send){

```
if(msg.port() == FacWar_in1){
        if((msg.value() > 100)&&(msg.value() <= 180)){
                int temp = msg.value() + currentRaw-100;
                //Warehouse overflow; discard surplus
                if(temp > rawMaterialCapacity){
                        temp = rawMaterialCapacity;
                        Display error message – overflow!
                }
                currentRaw=temp;
        }else{  //if the input is out of range
```

```
				Display error message – input out of range!
			}
		}
		if(msg.port() == FacWar_in2){
			if((msg.value() > 180)&&(msg.value() <= 200)){
				int temp = msg.value() + currentFinished-180;
				if(temp > finishedProductCapacity){
					temp = finishedProductCapacity;
					Display error message – finished products overflow!
				}
				currentFinished=temp;
			}else{  //if the input is out of range
				Display error message – input out of range!
			}
		}
		if(msg.port() == Send){
			if(msg.value()>=101 && msg.value()<=180){
				rawTobesent=msg.value()-100;
				if (currentRaw>=rawTobesent && this->state() == passive){
					sendRaw=true;
				}else{
					Display error message – not enough raw units in
warehouse!
				}
			}else if (msg.value()>=181 && msg.value()<=200){
				finishedTobesent=msg.value()-180;
				if (currentFinished>=finishedTobesent && this->state() ==
passive){
							sendFinished=true;
				}else{
					Display error message – not enough finished units in
```

```
warehouse!
                    }
            }else{
                    Display error message – invalid input from administrator!
            }
        }
    }
```

}

δ $_{\textbf{int}}$**(phase, sigma)**{

```
    if(this->state() == active){
                passive();
    }else {
            Display error meassage - Simulation should never reach this point
    }
```

}

λ **(phase, sigma, sendRaw, currentRaw, rawTobesent, sendFinished, currentFinished, finishedTobesent)**{

```
    if(sendRaw){
            currentRaw=currentRaw-rawTobesent;
            sendRaw=false;
            send output " rawTobesent " to FacWar_out1
    }
    if(sendFinished){
            currentFinished=currentFinished-finishedTobesent;
            sendFinished=false;
            send output " finishedTobesent " to FacWar_out
    }
```

}

ta(“passive”) = ∞;

ta(“active”) = 0;

Testing Strategy

1. Verify the effects of overflow – all surplus units should be discarded without warning.
2. Verify the effects non-positive - error message should be displayed.
3. Verify that no output occurs if either trans_busy is true or pause is true.
4. Verify the raw_current_units is decreased by raw_units when there is an output to FacWar_out1.
5. Verify the finished_current_units is decreased by finished_units when there is an output to FacWar_out2.
6. Verify the raw_current_units is increased by raw_units when there is an input from FacWar_out1.
7. Verify the finished_current_units is increased by finished_units when there is an input from FacWar_out2.

Powder Room

PowderRoom = <X, Y, S, δ_{ext}, δ_{int}, λ, ta>

Where

InPorts = {"PMP_in", "Busy2"}, where X_{PMP_in} = {1,2,3,4}, X_{Busy2} = {true, false}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = {"Busy1", "Press"}, where Y_{Busy1} = {true,false}, Y_{Press} = {1}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of Output ports and values

S = {phase, sigma, raw, pressStat, powder, powderStat}

= {"passive", "active"}×R_0^+×{1,2,3,4}×{ true, false }×{true, false}×{true, false}

δ_{ext} (phase, sigma, raw, powderStat, pressStat, e){

```
if(msg.port() == PMP_in){
        if(msg.value() == 4 && !powderStat){
                raw=msg.value();
                powderStat = true; // change to powder
                holdIn(active, 0);  // change to powder
        } else {
                Display error message administrator should only send an input
of 4.
        }
}

if(msg.port() == Busy2){
        if(msg.value() == 1){
                pressStat = true;
                holdIn(active, 0);
        }else if(msg.value() == 0){
                pressStat = false;
                holdIn(active, 0);
        }else{
                 Display error message! Simulation should never get here;
        }
}
```

}

δ_{int} (phase, sigma, raw, powder,powderStat){

```
if(this->state() == active){
        if (powderStat && !powder && (raw==4)){
```

```
                powder=true;
                holdIn(active, powderRoom_time);
            }else{
                passivate();
            }
    } else {
            //this will never happen
            if(this->state() == passive){
                Display error message – simulation should never reach this
            }
    }
}
```

λ ("active", sigma, powder, powderStat, pressStat){

```
    if(!pressStat && powder){
            powder=false;
            powderStat=false;
            pressStat = true;
            send output 1 to Press
    }
    if(powderStat){
            send output 1 to Busy1
    }
    if(!powderStat){
            send output 0 to Busy1
    }
}
```

ta("passive") = ∞

ta("active") = {powderRoom_time, 0}

Testing Strategy

1. Verify that the input received through PMP_in port is checked to have the value of 4.
2. Verify that the values received through the port Busy2 is either a zero or one.
3. Verify that the message sent through the port Busy1 is either 0 or 1.
4. Validate that the port Busy1 sends a value of 1 when the powder room starts the process of converting raw materials to powder.
5. Verify that the port Busy1 sends a value of 0 the moment the powder room is free.
6. Ensure that processing time of the powder room is 2 hours.
7. Verify that the port Press sends a value of 1 once the powder room has converted the raw materials to powder and that the press room is free.

Press Room

PressRoom = <X, Y, S, δ_{ext}, δ_{int}, λ , ta>

Where

InPorts = {“Press”, “Busy3”}, where X_{Press} = {1}, X_{Busy3} = {true, false}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = {“Busy2”, “Blister”}, where Y_{Busy2} = {true,false}, $Y_{Blister}$ = {1}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of Output ports and values

S = {phase, sigma, press, pressStat, powder, blisterStat}

= {“passive”, “active”}×R_0^+×{ true, false }×{true, false}×{0,1}×{true, false}

δ_{ext}(phase, sigma, powder, blisterStat, preeStat, e){

```
if(msg.port() == Press){
        if(msg.value() == 1){
                powder=msg.value();
                pressStat = true; // change to press
                holdIn(active, 0);  // change to press
        } else {
                Display error message the powder room should only send an
input of 1.

        }
}

if(msg.port() == Busy3){
        if(msg.value() == 1){
                blisterStat = true;
                holdIn(active, 0);
        }else if(msg.value() == 0){
                blisterStat = false;
                holdIn(active, 0);
        }else{
                Display error message! Simulation should never get here;
        }
}
```

}

δ_{int}(phase, sigma, powder, press, pressStat){

```
if(this->state() == active){
        if (pressStat && !press && (powder==1)){
                press = true;
```

```
                holdIn(active, pressRoom_time);  // change to press
        }else{
                passivate();
        }
    } else {
        if(this->state() == passive){
                Display error message, simulation should never reach this.
        }
    }
```

}

λ ("active", sigma, press, blisterStat, pressStat){

```
    if(!blisterStat && press){
        press=false;
        pressStat=false;
        blisterStat = true;
        send output 1 to Blister;
    }
    if(pressStat){
        send output 1 to Busy2;
    }
    if(!pressStat){
        send output 0 to Busy2;
    }
```

}

ta("passive") = ∞

ta("active") = {pressRoom_time, 0}

Testing Strategy

1. Verify that the input received through Press port is checked to have the value of 1.
2. Verify that the values received through the port Busy3 is either a zero or one.
3. Verify that the message sent through the port Busy2 is either 0 or 1.
4. Validate that the port Busy2 sends a value of 1 when the press room starts the process of converting powder to pill/tablet.
5. Verify that the port Busy2 sends a value of 0 the moment the press room is free.
6. Ensure that processing time of the press room is 4 hours.
7. Verify that the port Blister sends a value of 1 once the press room has converted the powder to pill/tablet and that the blister room is free.

Blister Room

BlisterRoom = <X, Y, S, δ_{ext}, δ_{int}, λ , ta>

Where

InPorts = {“Blister”, “Busy4”}, where $X_{Blister}$ = {1}, X_{Busy4} = {true, false}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = {“Busy3”, “Pack”}, where Y_{Busy3} = {true,false}, Y_{Pack} = {1}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of Output ports and values

S = {phase, sigma, press, packStat, blister, blisterStat}

= {“passive”, “active”}×R_0^+×{0,1}×{true, false}×{ true, false }×{true, false}

δ_{ext}(phase, sigma, press, packStat, blisterStat, e){

```
if(msg.port() == Blister){
        if(msg.value() == 1){
                press=msg.value();
                blisterStat = true; // change to blister
                holdIn(active, 0);  // change to blister
        } else {
                Display error message the press room only send an input of 1
        }
}

if(msg.port() == Busy4){
        if(msg.value() == 1){
                packStat = true;
                holdIn(active, 0);
        }else if(msg.value() == 0){
                packStat = false;
                holdIn(active, 0);
        }else{
                Display error message! Simulation should never get here;
        }
}
```

}

δ_{int}(phase, sigma, blister, press,blisterStat){

```
if(this->state() == active){
        if (blisterStat && !blister && (press==1)){
                blister = true;
                holdIn(active, blisterRoom_time);  // change to blister
        }else{
```

```
            passivate();
        }
    } else {

        if(this->state() == passive){
            This will never happen
        }
    }
```

}

λ ("active", sigma, blister, blisterStat, packStat){

```
    if(!packStat && blister){
        blister=false;
        blisterStat=false;
        send output 1 to FinalPack;
        packStat = true;
    }
    if(blisterStat){
        send output 1 to Busy3;
    }
    if(!blisterStat){
        send output 0 to Busy3;
    }
```

}

ta("passive") = ∞

ta("active") = {BlisterRoom_time, 0}

Testing Strategy

1. Verify that the input received through Blister port is checked to have the value of 1.
2. Verify that the values received through the port Busy4 is either a zero or one.
3. Verify that the message sent through the port Busy3 is either 0 or 1.
4. Validate that the port Busy3 sends a value of 1 when the blister room starts the process of blistering the pill/tablet that were received from the press room.
5. Verify that the port Busy3 sends a value of 0 the moment the press room is free.
6. Ensure that processing time of the blister room is 6 hours.
7. Verify that the port FinalPack sends a value of 1 once the blister room has blistered the pill/tablet and that the pack room is free.

Pack Room

PackRoom = <X, Y, S, δ_{ext}, δ_{int}, λ , ta>

Where

InPorts = {“Final_Pack”, “Busy3”}, where X_{Final_Pack} = {1}, X_{Busy4} = {true, false}

X = {(p,v)|p∈InPorts, v∈X_p} is the set of input ports and values

OutPorts = { “PMP_out”}, where Y_{PMP_out} = {1}

Y = {(p,v)|p∈OutPorts, v∈Y_p } is the set of Output ports and values

S = {phase, sigma, blister, packStat, pack}

= {“passive”, “active”}×R_0^+×{0,1}×{true, false}×{true, false}

```
δext(phase, sigma, blister,  packStat, e){
    if(msg.port() == FinalPack){
        if(msg.value() == 1){
            blister=msg.value();
            packStat = true; // change to pack
            holdIn(active, 0);  // change to pack
        } else {
            Display error message blister room should only send an input of 1.
        }
    }
}
δint(phase, sigma, pack,packStat, blister){
    if(this->state() == active){
        if (packStat && !pack && (blister==1)){
            pack = true;
            holdIn(active, packRoom_time);  // change to pack
        }else{
            passivate();
        }
    }else{
```

```
            // Simulation should never reach this
        }
    }
    λ ("active", sigma, pack, packStat){
        if(pack){
            pack=false;
            packStat=false;
            send output 1 to PMP_out;
        }
        if(!packStat){
            send output 0 to Busy4
        }
        if(packStat){
            send output 1 to Busy4
        }
    }
    ta("passive") = ∞
    ta("active") = {packRoom_time, 0}
```

Testing Strategy

1. Verify that the input received through FinalPack port is checked to have the value of 1.
2. Verify that the message sent through the port Busy4 is either 0 or 1.
3. Validate that the port Busy4 sends a value of 1 when the pack room starts the process of packing the blister packs.
4. Verify that the port Busy4 sends a value of 0 the moment the pack room is free.
5. Ensure that processing time of the pack room is 8 hours.
6. Verify that the port PMP_out sends a value of 1 once the pack room has packaged the blister packs.

Coupled models

Pharmaceutical Manufacturing Plant (PMP)

$$\mathbf{PMP} = < X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, Select>$$

Where

InPorts = {“PMP_in”}, where $X_{PMP\_in} = \{1, 2, 3, 4\}$

X = {(“ PMP_in” , v) | v$\in X_{PMP_in}$}

OutPorts = {“PMP_out”, “Busy1”}, where $Y_{PMP\_out} = \{1\}$, $Y_{Busy1} = \{-1, 1\}$

Y = {(p,v)|p$\in$OutPorts, v$\in Y_p$ }

D = {PowderRoom, PressRoom, BlisterRoom, PackRoom}

$\mathbf{M}_{PMP} = \{\mathbf{M}_{PowderRoom}, \mathbf{M}_{PressRoom}, \mathbf{M}_{BlisterRoom}, \mathbf{M}_{PackRoom}\}$

EIC = {(PMP, “PMP_in”), (PowderRoom, “PMP_in”)}

EOC = {((PowderRoom, “Busy1”), (PMP, “Busy1”)),

((PackRoom, “PMP_out”), (PMP, “PMP_out”))}

IC = {((PowderRoom, “Press”), (PressRoom, “Press”)),

((PressRoom, “Busy2”), (PowderRoom, “Busy2”)),

((PressRoom, “Blister”), (BlisterRoom, “Blister”))

((BlisterRoom, “Busy3”), (PressRoom, “Busy3”))

((BlisterRoom, “FinalPack”), (PackRoom, “FinalPack”))

((PackRoom, “Busy4”), (BlisterRoom, “Busy4”))}

Select: ({PackRoom, PressRoom}) = PackRoom

({PackRoom, BlisterRoom}) = PackRoom

({PackRoom, PowderRoom}) = PackRoom

({BlisterRoom, PressRoom}) = BlisterRoom

({BlisterRoom, PowderRoom}) = BlisterRoom

({PressRoom, PowderRoom}) = PressRoom

Testing Strategy

1. Verify that when a value of 1 is passed to PMP through the port PMP_in there is an output through the port PMP_out after the pre-defined time of 20 hours.
2. Verify that when a value of 1 is passed to PMP through the port PMP_in there is an output through the port BUSY1 simultaneously.

Factory

Factory = < X, Y, D, $\{M_d \mid d \in D\}$, EIC, EOC, IC, Select>

Where

InPorts = {"RawMaterials2", "OrderInfo2"}, where $X_{RawMaterials2} = \{1, 2, 3, \ldots, 78, 79, 80\}$, $X_{OrderInfo2} = \{1, 2, 3, \ldots, 18, 19, 20\}$

X = $\{(p, v) \mid v \in X_{PMP\_in}\}$

OutPorts = { "FinishedPro1"}, where $Y_{FinishedPro1} = \{-1, 1\}$

Y = $\{(p,v) \mid p \in OutPorts, v \in Y_p\}$

D = {FactoryWarehouse, FactoryAdministrator, PMP}

$\mathbf{M}_{Factory} = \{\mathbf{M}_{FactoryWarehouse}, \mathbf{M}_{FactoryAdministrator}, \mathbf{M}_{PMP}\}$

EIC = {(Factory, "RawMaterials2"), (FactoryAdministrator, "RawMaterials2")

((Factory, "OrderInfo2"), (FactoryAdministrator, "OrderInfo2"))}

EOC = {((FactoryAdministrator, "FinishedPro1"), (Factory, "FinishedPro1"))}

IC = {((FactoryAdministrator, "Send"), (FactoryWarehouse, "Send")),

((FactoryAdministrator, "FacWar_in1"), (FactoryWarehouse, "FacWar_in1")),

((FactoryAdministrator, "FacWar_in2"), (FactoryWarehouse, "FacWar_in2")),

((FactoryWarehouse, "FacWar_out1"), (FactoryAdministrator, "FacWar_out1")),

((FactoryWarehouse, "FacWar_out2"), (FactoryAdministrator, "FacWar_out2")),

((PMP, "PMP_out"), (FactoryAdministrator, "PMP_out"))

((PMP, "Busy1"), (FactoryAdministrator, "Busy1"))

({FactoryAdministrator, “PMP_in”), (PMP, “PMP_in”))}

Select: ({PMP, FactoryAdministrator }) = PMP

({PMP, FactoryWarehouse }) = PMP

({FactoryAdministrator, FactoryWarehouse }) = FactoryAdministrator

Testing Strategy

1. Each atomic model (administrator, warehouse, powder room, press room, blister room and pack room) was tested and verified individually according to the test strategies defined above.
2. The coupled model (PMP) was also tested and verified separately.
3. The following test condition were used to test and verify the complete factory model:

    1. Send an input of value 10 through the port OrderInfo2 at 00:00:01:01 and send an input of 20 through the port RawMaterials2 at 00:00:02:01

    2. Send an input of value 22 through the port OrderInfo2 at 00:00:01:01 and send an input of 19 through the port OrderInfo2 at 00:00:02:01

    3. Send an input of value 50 through the port RawMaterials2 at 00:00:01:01 and send an input of 20 through the port RawMaterials2 at 00:00:02:01

    Note: All the above test conditions the number of finsihed products in the warehouse initally was set to 19 and the number of raw materials in the warehouse was set to 50. In addition, initially the pharmaceutical plant (PMP) was not processing anything in any of its components.

Testing and Simulation Analysis

Both the atomic and coupled models were all implemented and tested using the toolkit CD++. The atomic models (administrator, warehouse, powder room, press room, blister room and pack room) were implemented and tested independantly before there were incorporated into the coupled models. The PMP coupled model was also implemented and tested seperately before the it was integrated into the factory coupled model. Although numerous tests were performed to ensure all models were behaved as desired, only a selected three test cases are outlined below.

Note that to better understand the tests cases, the block diagram was modified as seen in Figure 2. This not only ensures that the outputs of the factory coupled model are outputted, but also shows some processing of the second level models (administrator, warehouse and PMP). In addition for all the test conditions outlined below, the number of finsihed products in the warehouse initally was set to 19 and the number of raw materials in the warehouse was set to 50. Furthermore, initially the pharmaceutical plant (PMP) was not processing anything in any of its components.

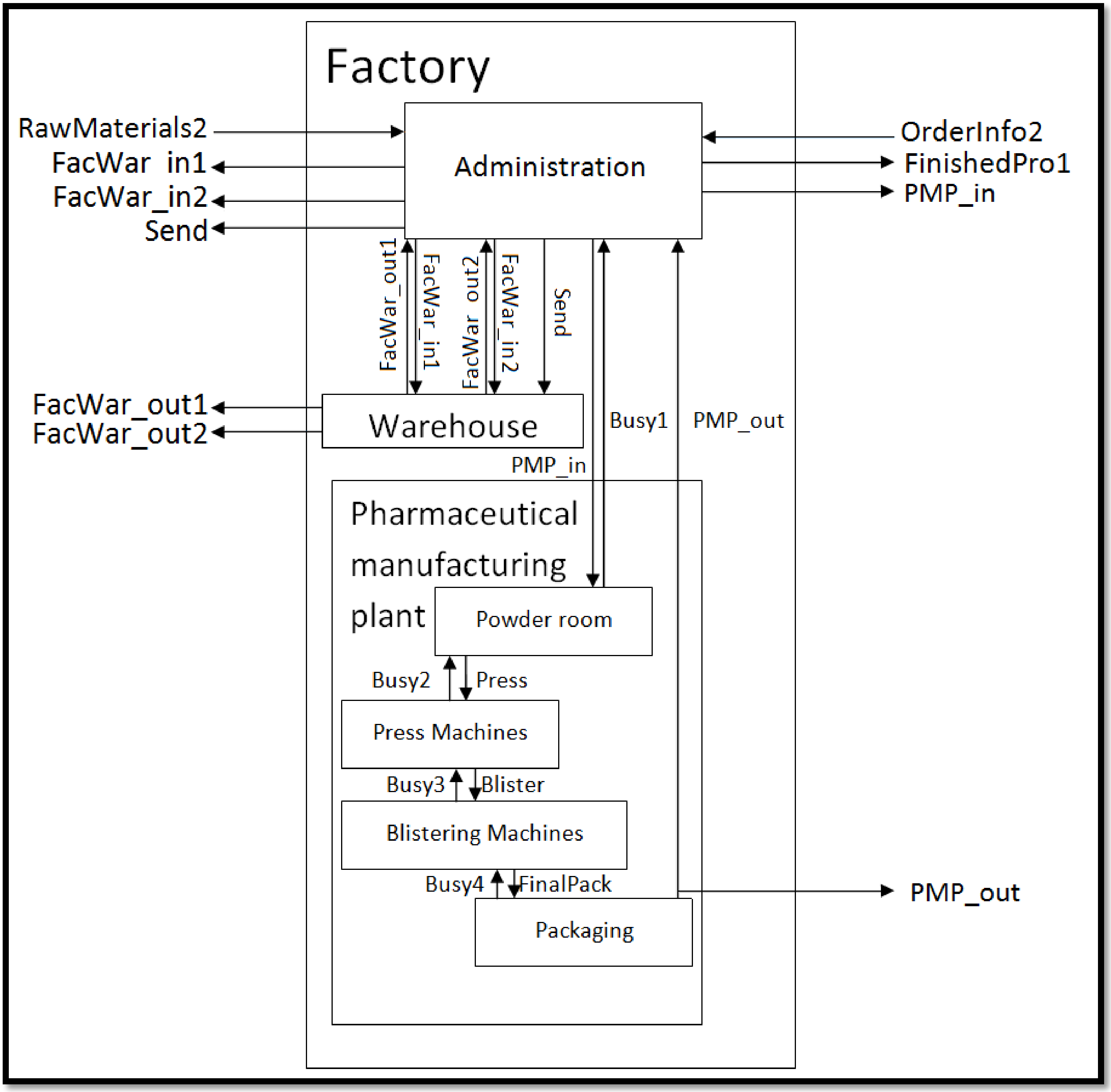

Figure 2: Modified block diagram of Factory model for testing purposes

Test Case 1: Valid Inputs

Table 1: Input (event file) and output (out file) displayed in time sequence for case 1 - valid inputs

Input	Output
	00:00:00:000 send 104 00:00:00:000 facwar_out1 4 00:00:01:000 pmp_in 4
00:00:01:01 OrderInfo2 10 00:00:02:01 RawMaterials2 20	
	00:00:03:001 send 190 00:00:03:001 facwar_in1 120 00:00:03:001 facwar_out2 10 00:00:04:001 finishedpro1 10 02:00:02:000 send 104 02:00:02:000 facwar_out1 4 02:00:03:000 pmp_in 4 06:00:02:000 send 104 06:00:02:000 facwar_out1 4 06:00:03:000 pmp_in 4 12:00:02:000 send 104 12:00:02:000 facwar_out1 4 12:00:03:000 pmp_in 4 20:00:01:000 pmp_out 1 20:00:02:000 facwar_in2 181 20:00:02:000 send 104 20:00:02:000 facwar_out1 4 20:00:03:000 pmp_in 4 28:00:01:000 pmp_out 1 28:00:02:000 facwar_in2 181 28:00:02:000 send 104 28:00:02:000 facwar_out1 4 28:00:03:000 pmp_in 4 36:00:01:000 pmp_out 1 36:00:02:000 facwar_in2 181 36:00:02:000 send 104 36:00:02:000 facwar_out1 4 36:00:03:000 pmp_in 4 44:00:01:000 pmp_out 1 44:00:02:000 facwar_in2 181 44:00:02:000 send 104 44:00:02:000 facwar_out1 4 44:00:03:000 pmp_in 4 52:00:01:000 pmp_out 1 52:00:02:000 facwar_in2 181

Table 2: Continuation of Table 1

Input	Output
	52:00:02:000 send 104 52:00:02:000 facwar_out1 4 52:00:03:000 pmp_in 4 60:00:01:000 pmp_out 1 60:00:02:000 facwar_in2 181 60:00:02:000 send 104 60:00:02:000 facwar_out1 4 60:00:03:000 pmp_in 4 68:00:01:000 pmp_out 1 68:00:02:000 facwar_in2 181 68:00:02:000 send 104 68:00:02:000 facwar_out1 4 68:00:03:000 pmp_in 4 76:00:01:000 pmp_out 1 76:00:02:000 facwar_in2 181 84:00:01:000 pmp_out 1 84:00:02:000 facwar_in2 181 92:00:01:000 pmp_out 1 92:00:02:000 facwar_in2 181

The above input and output was run for 99:00:00:000. This simulation shows that for 'proper' inputs (i.e. the order placed for finished products from the distributor is between 0 and 20, and is less than the current number of products that is held in the factory's warehouse. In addition the number of raw materials recieved from the supplier is less than or equal to the available amount of space in the factory's warehouse for raw materials, and the input is between the values of 0 and 80) there were no error messgaes displayed, and the simulation ran till the specified time.

This simulation also shows that factory is trying to replenish it's finished products stock from the beignning (00:00:00:000) since the amount of finished products stored in it's warehouse was not at full capacity. In addition the input and output times for PMP were verified to be correct. Lastly, the time at which various values were outputed from the administrator and warehouse thorugh the ports facwar_in1, facwar_in2, facwar_out1, facwar_out2 all occured at desiered times. Thus, it is verified that for correct inputs, the entire factory model behaves as specified.

CD++ Console View

Note that there were no errors recieved.

```
C:\eclipse\workspace\Factory>cd /D "C:\eclipse\workspace\Factory"

C:\eclipse\workspace\Factory>"C:/eclipse/workspace/Factory/simu.exe"  -m"Factory.MA" -e"FactoryCase1.ev" -
o"FactoryCase1OUT.out" -l"FactoryCase1LOG.log" -t"99:00:00:000"
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 Factory.MA
Loading events from FactoryCase1.ev
Message log: FactoryCase1LOG.log
Output to: FactoryCase1OUT.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/ta88.0
Printing parser information = OFF
DEVS-Graphs debug level: 0 (No debug info. Use -g parameter to specify debug level)

Starting simulation. Stop at time: 99:00:00:000
00:00:01:001 / orderinfo2 /     10.00000
00:00:02:001 / rawmaterials2 /     20.00000
Simulation ended!
```

Test Case 2: Invalid and Valid values for OrderInfo2

Table 3: Input (event file) and output (out file) displayed in time sequence for case 2

Input	Output
	00:00:00:000 send 104
	00:00:00:000 facwar_out1 4
	00:00:01:000 pmp_in 4
00:00:01:01 OrderInfo2 22	
00:00:02:01 OrderInfo2 19	
	00:00:03:001 send 199
	00:00:03:001 facwar_out2 19
	00:00:04:001 finishedpro1 19
	02:00:02:000 send 104
	02:00:02:000 facwar_out1 4
	02:00:03:000 pmp_in 4
	06:00:02:000 send 104
	06:00:02:000 facwar_out1 4
	06:00:03:000 pmp_in 4
	12:00:02:000 send 104
	12:00:02:000 facwar_out1 4
	12:00:03:000 pmp_in 4
	20:00:01:000 pmp_out 1
	20:00:02:000 facwar_in2 181
	20:00:02:000 send 104
	20:00:02:000 facwar_out1 4
	20:00:03:000 pmp_in 4
	28:00:01:000 pmp_out 1
	28:00:02:000 facwar_in2 181
	28:00:02:000 send 104
	28:00:02:000 facwar_out1 4
	28:00:03:000 pmp_in 4
	36:00:01:000 pmp_out 1
	36:00:02:000 facwar_in2 181
	36:00:02:000 send 104
	36:00:02:000 facwar_out1 4
	36:00:03:000 pmp_in 4
	44:00:01:000 pmp_out 1
	44:00:02:000 facwar_in2 181
	44:00:02:000 send 104
	44:00:02:000 facwar_out1 4
	44:00:03:000 pmp_in 4
	52:00:01:000 pmp_out 1
	52:00:02:000 facwar_in2 181
	52:00:02:000 send 104

Table 4: Continuation of Table 3

Input	Output
	52:00:02:000 facwar_out1 4 52:00:03:000 pmp_in 4 60:00:01:000 pmp_out 1 60:00:02:000 facwar_in2 181 60:00:02:000 send 104 60:00:02:000 facwar_out1 4 60:00:03:000 pmp_in 4 68:00:01:000 pmp_out 1 68:00:02:000 facwar_in2 181 68:00:02:000 send 104 68:00:02:000 facwar_out1 4 68:00:03:000 pmp_in 4 76:00:01:000 pmp_out 1 76:00:02:000 facwar_in2 181 76:00:02:000 send 104 76:00:02:000 facwar_out1 4 76:00:03:000 pmp_in 4 84:00:01:000 pmp_out 1 84:00:02:000 facwar_in2 181 92:00:01:000 pmp_out 1 92:00:02:000 facwar_in2 181

The above input and output was run for 99:00:00:000. This simulation shows how the factory model handles error. Note thate the first input was an improper input – i.e. the order placed by the distributor was more than 20 finished products. This is invalid because the model specifications dictates that no order exceeding 20 can be placed. However, the second order shows that for a proper input the simulation runs for specifed amount of time.

This simulation also shows that factory is trying to replenish it's finished products stock from the beignning (00:00:00:000) since the amount of finished products stored in it's warehouse was not at full capacity. In addition the input and output times for PMP were verified to be correct. Lastly, the time at which various values were outputed from the administrator and warehouse thorugh the ports facwar_in1, facwar_in2, facwar_out1, facwar_out2 all occured at desiered times. Thus, it is verified that for correct inputs, the entire factory model behaves as specified.

CD++ Console View

Note that an error was received (highlighted in red below) – "Error: Factory Administrator asked to send more than 20 finished products at a time!". This shows that the factory model does handle error as specified by above.

```
C:\eclipse\workspace\Factory>cd /D "C:\eclipse\workspace\Factory"

C:\eclipse\workspace\Factory>"C:/eclipse/workspace/Factory/simu.exe"  -m"Factory.MA" -
e"FactoryCase2.ev" -o"FactoryCase2OUT.out" -l"FactoryCase2LOG.log" -t"99:00:00:000"
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 Factory.MA
Loading events from FactoryCase2.ev
Message log: FactoryCase2LOG.log
Output to: FactoryCase2OUT.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/ta90.0
Printing parser information = OFF
DEVS-Graphs debug level: 0 (No debug info. Use -g parameter to specify debug level)

Starting simulation. Stop at time: 99:00:00:000
00:00:01:001 / orderinfo2 /     22.00000
00:00:02:001 / orderinfo2 /     19.00000
Error: Factory Administrator asked to send more than 20 finished products at a time!
Simulation ended!
```

Test Case 3: Invalid and Valid values for RawMaterials2

Table 5: Input (event file) and output (out file) displayed in time sequence for case 3

Input	Output
	00:00:00:000 send 104 00:00:00:000 facwar_out1 4 00:00:01:000 pmp_in 4
00:00:01:01 RawMaterials2 50 00:00:02:01 RawMaterials2 20	
	00:00:03:001 facwar_in1 120 20:00:01:000 pmp_out 1 20:00:02:000 facwar_in2 181

The above input and output was run for 99:00:00:000. This simulation shows how the factory model handles error. Note thate the first input was an improper input – i.e. the amount of raw materials recieved by the factory from the supplier exceeded the factory's warehouse capacity. This is invalid because the model specifications dictates that the factory will not accept a shipment of raw materials unless it has the capacity to store the entire shipment. However, the second order shows that for a proper input the simulation runs for specifed amount of time.

This simulation also shows that factory is trying to replenish it's finished products stock from the beignning (00:00:00:000) since the amount of finished products stored in it's warehouse was not at full capacity. In addition the input and output times for PMP were verified to be correct. Lastly, the time at which various values were outputed from the administrator and warehouse thorugh the ports facwar_in1, facwar_in2, facwar_out1, facwar_out2 all occured at desiered times. Thus, it is verified that for correct inputs, the entire factory model behaves as specified. In addition unlike the previous two test cases, the factory administrator only sent one batch of products to be manfucatured. This is because no orders from the distributor were placed and the current number of finsihed products held in the warehouse is 19 units.

CD++ Console View

Note that an error was received (highlighted in red below) – "Error: Supplier should not send more raw materials than capacity dictates!". This shows that the factory model does handle error as specified by above.

```
C:\eclipse\workspace\Factory>cd /D "C:\eclipse\workspace\Factory"

C:\eclipse\workspace\Factory>"C:/eclipse/workspace/Factory/simu.exe"  -m"Factory.MA" -
e"FactoryCase3.ev" -o"FactoryCase3OUT.out" -l"FactoryCase3LOG.log" -t"99:00:00:000"
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 Factory.MA
Loading events from FactoryCase3.ev
Message log: FactoryCase3LOG.log
Output to: FactoryCase3OUT.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/t128c.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: 99:00:00:000
00:00:01:001 / rawmaterials2 /     50.00000
00:00:02:001 / rawmaterials2 /     20.00000
Error: Supplier should not send more raw materials than capacity dictates!
Simulation ended!
```

Bibliography

[1] James A. Tompkins, "The Challenge of Warehousing," in *The Warehouse Management Handbook*, James A. Tompkins and Jerry D. Smith, Eds. Raleigh, North Carolina, United States of America: Tompkins Press, 1998, ch. 1, p. 2.

[2] Gabriel A. Wainer, "Introduction to the DEVS Modeling and Simulation Formalism," in *Discrete-event modeling and simulation: a practitioner's approach*. Boca Raton, United States of America: CRC Press, 2009, ch. 2, pp. 35-54.