Methodology for Discrete-Event Modelling and Simulation

LaundryCard Money Depositer

Assignment 1

**Name: Revathy Ilamvazhudhi**

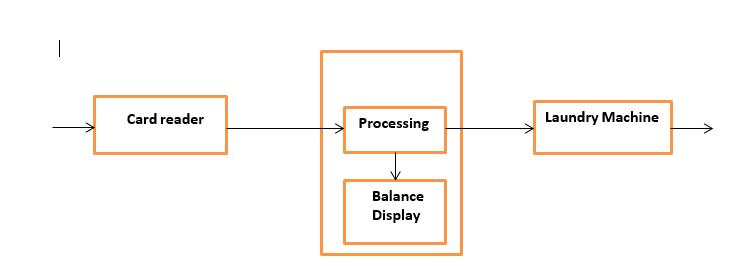
**Student No: 8483553**

**Part 1**

1. **Description of the System**

This system provides service of helping customers residing in an apartment recharge their Laundry card by themselves. If a resident inserts the “Laundry card” and if there is money, it enters the Laundry controller. The current balance display will be shown on the machine and the machine is ready to start its process.

1. **Model structure.**



**Fig 1: LaundryCard Model**

1. **A brief description for each component.**

The atomic models are described as follows:

Card Reader: Read the information such as the card no, name and the current balance from the LaundryCard and forward this information to the Processor.

Laundry Machine: After receiving the confirm signal, the machine would be ready to start its process.

Balance Display: Display the initial and the final balance of the Laundry Card.

Processing: Gets the input from card reader and checks if there is balance in it.

# **Part 2**

# **Detailed structure**

The Laundry card machine consists of four atomic models and one coupled model. The atomic models include card reader, Processing, source, Balance display, Laundry machine. The coupled model is Laundry Controller. The following is the detailed structure of the Laundry card Machine.

# **Fig 2: Detailed model of Laundrycard Model**

# **Formal specification for each atomic model and coupled model**

* 1. *Formal specification for CardReader*

X={card ∈ <N, R>};

Y={card\_info ∈ N};

S={phase, sigma, card\_info<card.id, card.balance>, preparation\_time};

δext(s,e,x)

{

case phase

passive:

sigma=preparation\_time;

get card\_info;

phase = busy;

busy:

continue;

}

δint(s,e)

{

case phase

busy: passivate;

passivate: /\*never happens\*/

}

λ(s)

{

Send card\_info to the port out

}

* 1. *Formal specification for BalanceDisplay*

X={balance ∈ <N, R>};

Y={balance\_display ∈ <N, R>};

S={phase, sigma, balance, preparation\_time};

δext(s,e,x)

{

case phase

passive:

sigma=preparation\_time;

save balance;

phase = busy;

busy:

continue;

}

δint(s,e)

{

case phase

busy: passivate;

passivate: /\*never happens\*/

}

λ(s)

{

Send balance\_display to the port out

}

* 1. *Formal specification for Processing*

X = {CardReader\_in∈ < N, R >, CardReader\_in∈ < N, R >, LaundryControl\_done∈ < N > };

Y = { message\_out ∈ < N, R >};

S = { Phase, sigma, Processing, message, preparationTime }

δext ( s, e, x )

{

case port

message\_in /\* this message includes the message from Processing,

CardReader\_in \*/

push\_back(message\_queue);

sigma = preparationTime;

phase = busy;

done:

pop\_front ( Processing );

if (Processing!= 0);

phase = busy;

sigma = preparationTime;

}

δint ( s, e )

{

case phase

busy: passivate;

passive: /\*never happens\*/

}

λ (s)

{

send message to the port message\_out;

}

* 1. *Formal specification for LaundryControl*

X = { Processing\_in ∈ < N, R >};

Y = { BalanceDisplay\_out∈ < N, R >, LaundryControl\_done ∈ < N > };

S = { Phase, sigma, balance, card\_info, preparationTime }

δext ( s, e, x )

{

message\_in:

if (message.value <10 ) /\* this message is from Cardreader\*/

save balance;

sigma = preparationTime;

phase = busy;

elseif (message.value >=10) /\* this message is from CardReader\*/

save the card\_info;

=card\_info.balance+Cardreader.balance;

sigma = preparationTime;

phase = busy;

}

δint ( s, e )

{

case phase

busy: passivate

passivate: /\* never happens\*/

}

λ (s)

{

send balance to the port balance\_out;

send done to the port done;

}

* 1. *Formal specification for LaundryMachine*

LaundryControl = < X, Y, { Processing, LaundryControl }, EIC, EOC, IC, SELECT >

X = { card\_in }

Y = { balance\_out}

EIC = { ( Cardreader.out, money\_in), (CardReader.out, card\_in)}

EOC = { (balance\_out, BalanceDisplay.in)}

IC = { (money\_in, Processing.in), (card\_in, Processing.in, (LaundryControl.out, balance\_out), (LaundryControl.out , Processing.done), (Processing.out, LaundryControl.in)}

SELECT : ({Processing, LaundryControl}) = Processing

# **Part 3**

# **Simulation Results**

Before explaining the simulation of input and output, the simulation are predefined with a list of values that were entered.

**card\_in:** The value is always 1 which represents a card has been inserted into the Card Reader.

Output for CardReader:

00:00:22:000 outcr 1

00:00:32:000 outcr 1

00:00:42:000 outcr 1

00:00:52:000 outcr 1

00:01:02:000 outcr 1

00:01:07:000 outcr 1

Output for the Top model: LaundryOperation:

00:00:16:000 display 1

00:00:36:000 display 1

00:00:56:000 display 1