SYSC-5807
METHODOLOGICAL ASPECTS OF MODELLING AND SIMULATION

MANUFACTURING FACILITY

(Assignment 1)

Date: October 20, 2003.

Shanta Ramchandani
Student #: 100404100

PART 1 – CONCEPTUAL MODEL

A Manufacturing Facility

The manufacturing facility assembles 3 types of products: P1, P2, and P3. These products consist of one of more components: C1, C2, and C3.
Product P1 contains component C1.
Product P2 contains components C1 and C2.
Product P3 contains components C1 and C3.

Two inspectors, service the components. Inspector 1 works on component C1. Inspector 2 works on components C2 and C3 in random order. The inspectors will never have to wait for components. There is an infinite inventory of them and they are always immediately available.

Three workstations in the facility, W1, W2, and W3, assemble products P1, P2, and P3 respectively.

After the components are serviced, they are passed to the respective workstations. Each workstation has a buffer capacity of 2 components with one buffer for each of the component types needed.

A product can begin being assembled only when components of all types required are available. If all workstation buffers for a specific component are full, the corresponding inspector is considered 'blocked' until there is an opening at which time the inspector can resume servicing and sending components of that type.

Inspector 1 routes components C1 to the buffer with the smallest number of components in waiting. In case of a tie, W1 has the highest priority and W3 has the lowest.
Inspector

Component	State Variables	Type
Warehouse	partType	Integer {1,2}

Prioritizer	bufStatus1
	bufStatus2
	bufStatus2
	partType
	start
		Integer {0,1,2}
		Integer {0,1,2}
		Integer {0,1,2}
		Integer {-1,1}
		Integer {0,1}

Sorter	buf2Status
	buf3Status
	partType
	working
		Integer{0,1,2}
		Integer{0,1,2}
		Integer {0,1,2}
		Integer {0,1}

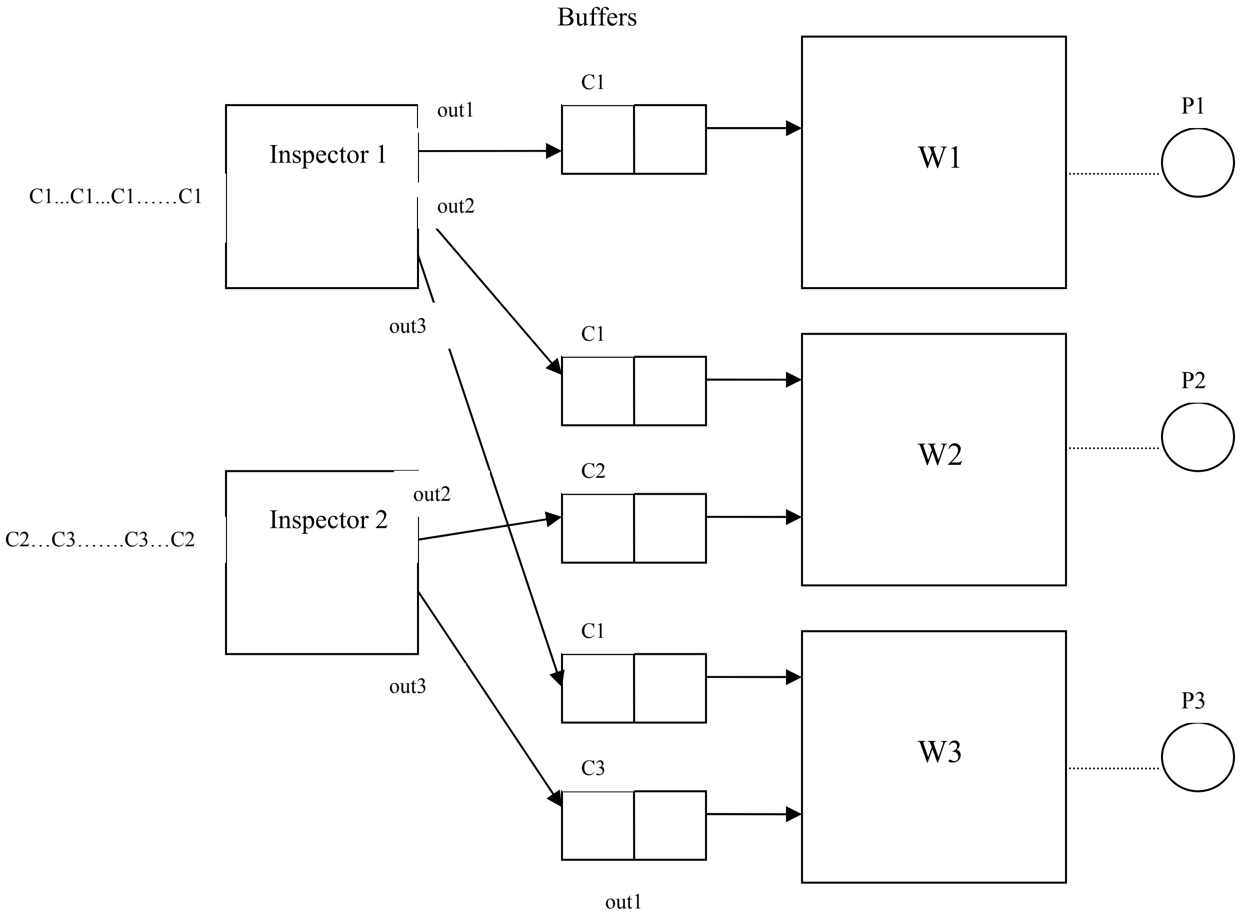
Buffer	elements
	sendElement
		list of Integers{1,2}
		Integer {0,1}

Workstation1	product
	partIn
	firstProduct
	numElemBuf
		Integer {1}
		Integer {0,1}
		Integer {0,1}
		Integer {0,1,2}

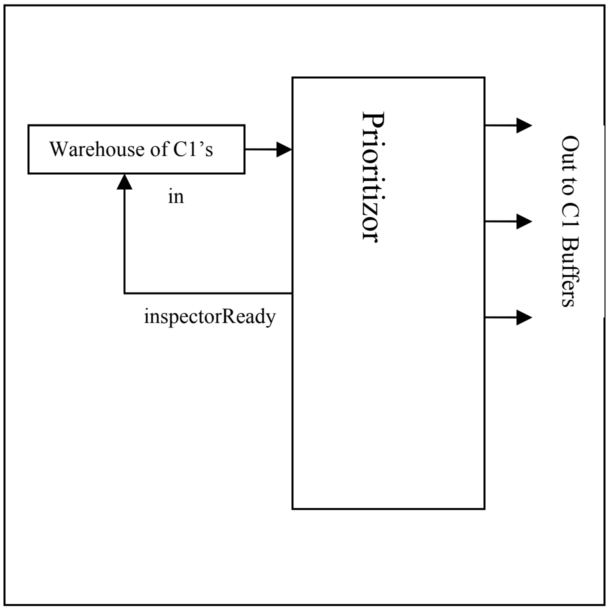
Workstation2	sizeBuffer1 sizeBuffer2	Integer {0,1,2} Integer {0,1,2}

	in1Value	Integer {1,2}
	in2Value	Integer {2,3}
	product	Integer {0,1}
	partsReady	Integer {0,1}
	createProduct	Integer {0,1}

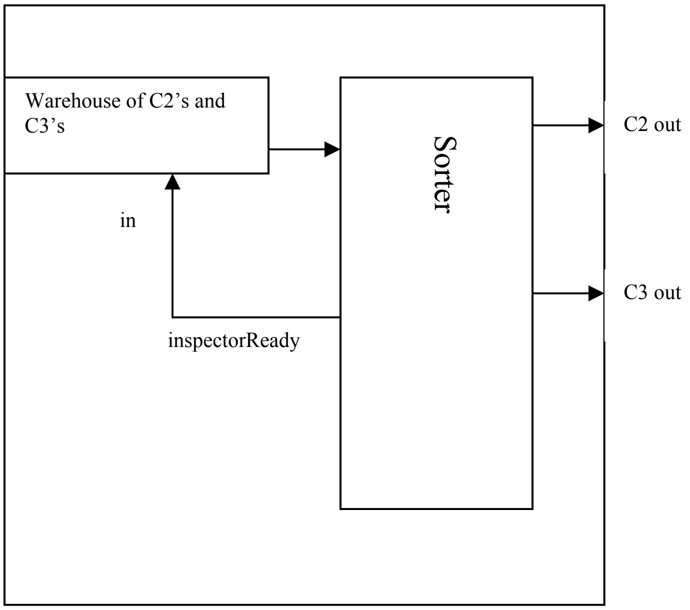
****NOTE: Phase and sigma are implicit for all of the above atomic models.



Inspector 1:



Inspector 2:



Shanta Ramchandani
Student #: 100404100

PART 2 – FORMAL SPECIFICATIONS AND TESTING STRATEGIES

Atomic Model Warehouse

X = {in}
Y = {out}
S = {wait_for_input}
Internal Transition Function:

```
        passivate
```

External Transition Function:

```
    if msg.port() == in {
        partType = random_value between 1 and max_num_parts
        holdIn( active, Time(0,0,0,3) )
    }
```

Output Function:

```
    send partType to port out
```

Testing Strategy:

- Send input to in port and verify that the output is a random value of either 1 or 2 if NUM_PARTS is specified to be 2.
- Send input to in port and verify that the output is of value 1 if NUM_PARTS is specified to be any value except for 2.
- For the above, verify that the outputs occur 3ms after the input occurs.

Atomic Model Prioritizer

X = {in, numBuf1, numBuf2, numBuf3}
Y = {out1, out2, out3, inspectorReady}
S = {{Phase, sigma, bufStatus1, bufStatus2, bufStatus3, partType, start}}
Internal Transition Function:

```
        passivate
```

External Transition Function:

```
    if (msg.port() == numBuf1) {
        buf1Status = msg.value();
        holdIn( active, Time::Zero) ;
    }else if (msg.port() == numBuf2) {
        buf2Status = msg.value();
        holdIn( active, Time::Zero) ;
    }else if (msg.port() == numBuf3) {
        buf3Status = msg.value();
        holdIn( active, Time::Zero) ;
    }else if ((msg.port() == in) && (partType == -1)) {
        partType = msg.value();
        holdIn( active, Time(random ) ) ;
    }
```

Output Function:

```
    if (state() == passive && no_part_received && ((buf1Status < 2.0) || (buf2Status < 2.0) ||
    (buf3Status < 2.0)) || start == 1 ) {
        send output to port inspectorReady;
        start = 0;
```

```
} else {
    if (partType != -1) {
        if ((buf1Status == 2.0) && (buf2Status == 2.0) && (buf3Status == 2.0))          {
            partType = -1
            Do not send anything;
        } else if ( (buf3Status < buf1Status) && (buf3Status < buf2Status) ) {
            send output partType to port out3;
            send output to port inspectorReady;
            partType = -1;
        } else if (buf2Status < buf1Status) {
            send output partType to port out2;
            send output to port inspectorReady;
            partType = -1;
        } else {
            send output partType to port out1;
            send output to port inspectorReady;
            partType = -1;
        }
 }
```

Testing Strategy:

- When a input occurs at port in, verify that the outputs are sent to the correct output based on the prioritization scheme identified in Part1 after a random amount of time.
- When an input occurs at the numBuf1, numBuf2, and/or numBuf3 ports, verify that the buffer value is updated and the bufStatus1, bufStatus2, and/or bufStatus3 are updated respectively.
- Verify that the inspectorReady signal occurs at startup and also with every output.
- Verify that no output occurs when all bufStatus1== bufStatus1== bufStatus1== 2.
- Verify that the numBuf1, numBuf2 and numBuf3 inputs will cause an output only if the Prioritizer is not working on another part. If it is working on another part, the bufStatus1, bufStatus2 and bufStatus3 variables are updated with the input value.

Atomic Model Sorter

X = {in, numBuf1, numBuf2}
Y = {out2, out3, inspectorReady}
S = {{Phase, sigma, buf2Status, buf3Status, partType, working}}
Internal Transition Function:

```
working = 0;   //stop working
passivate;
```

External Transition Function:

```
if (msg.port() == numBuf2) {
    buf2Status = msg.value();
    if (working == 0) holdIn( active, Time::Zero) ;
}else if (msg.port() == numBuf3) {
    buf3Status = msg.value();
    if (working == 0) holdIn( active, Time::Zero) ;
}else if (msg.port() == in) {
    partType = msg.value();
    working = 1;
    holdIn( active, Time( random ) ) ;
}
```

Output Function:

```
if (working == 0 && partType == 0 && ((buf2Status < 2.0) || (buf3Status < 2.0)) ) {
    send output to port inspectorReady
} else {
    if (partType != 0 && working == 1) {
        if (partType == 1 && buf2Status < 2) {
            send partType to port out2;
            send output to port inspectorReady;
            partType = 0;
        } else if (partType == 2 && buf3Status < 2) {
            send partType to port out2;
            send output to port inspectorReady;
            partType = 0;
        }
    }
}
```

Testing Strategy:

- Verify that no ouput occurs if either of bufStatus2 or bufStatus3 == 2.
- Verify that the input part is sent out its corresponding output after a random amount of time.
- Verify that the inspectorReady output occurs at startup and with every output.
- Verify that the numBuf2 and numBuf3 inputs will cause an output only if the Sorter is not working on another part. If it is working on another part, the bufStatus2 and bufStatus3 variables are updated with the input value.

Atomic Model Buffer

X = {in, done}
Y = {out, numElemOutput}
S = {{Phase, sigma, elements, sendElement}}
Internal Transition Function:

```
        passivate
```

External Transition Function:

```
if( msg.port() == in ){
    if( elements.size() < BUFFER_MAX_SIZE ) {
        elements.push_back( msg.value() ) ;}
if( msg.port() == done ){
        if( elements.size() > 0 ) {
        sendElement = 1;}
holdIn( active, Time::Zero );           // Call output function to update numElemOutput
```

Output Function:

```
if (sendElement == 1) {
    sendElement = 0;
    send buffer element to port out;
    remove the element from the buffer;
}
send number of elements in the buffer to port numElemOutput
```

Testing Strategy:

- Verify that the buffer will only hold a maximum of 2 elements. All other elements that enter the buffer at this point are discarded.
- Verify that the buffer sends an element out of the 'out' port when it receives a signal from its 'done' port.
- Verify that the buffer sends the number of elements in the buffer to the numElemOutput port after it services a 'done' signal and after it receives an input from 'in'.

Atomic Model Workstation1

X = {in, numBuf}
Y = {out, ready}
S = {{Phase, sigma, product, partIn, firstProduct, numElemBuf}}
Internal Transition Function:

```
        passivate
```

External Transition Function:

```
    if (msg.port() == in) {
        partIn = 1;
        holdIn( active, Time(random));
    }
    if (msg.port() == numBuf) {
        numElemBuf = msg.value();
        holdIn(active, Time::Zero);
    }
```

Output Function:

```
    if (partIn == 1) {
        send product to port out;
        partIn = 0;
    }
    if (numElemBuf > 0) {
        send output to port ready;
    }
```

Testing Strategy:

- When receiving an input from port 'in' verify that the workstation sends out a product '1' after a random amount of time and at the same time, an output to the 'ready' port.
- Verify that the workstation sends an output to the 'ready' port when numElemBuf variable is > 0.
- Verify that the workstation does nothing when the numElemBuf variable is 0.

Atomic Model Workstation2

X = {input1, input2, numBuf1, numBuf2}
Y = {out, ready}
S = {{Phase, sigma, sizeBuffer1, sizeBuffer2, in1Value, in2Value, product, partsReady, createProduct}}
Internal Transition Function:

```
        passivate
```

External Transition Function:

```
    if (msg.port() == input1) {
        if(msg.value() != 1) product = msg.value();
        in1Value = 1;
        if (in1Value != 0 && in2Value != 0) createProduct = 1;
        holdIn( active, Time( static_cast< float >( fabs( distribution().get() ) ) ) ) ;
    }else if (msg.port() == input2) {
        if(msg.value() != 1) product = msg.value();
        in2Value = 1;
        if (in1Value != 0 && in2Value != 0) createProduct = 1;
        holdIn( active, Time( static_cast< float >( fabs( distribution().get() ) ) ) ) ;

    }else if (msg.port() == numBuf1) {
        sizeBuffer1 = msg.value();
        if ((sizeBuffer1 > 0) && (sizeBuffer2 > 0)) {
            partsReady = 1;
            if (state() == passive) {
                holdIn( active, Time( static_cast< float >( fabs( distribution().get() ) ) ) ) ;
            }
        }
    }else if (msg.port() == numBuf2) {
        sizeBuffer2 = msg.value();
        if ((sizeBuffer1 > 0) && (sizeBuffer2 > 0)) {
            partsReady = 1;
            if (state() == passive) {
                holdIn( active, Time( static_cast< float >( fabs( distribution().get() ) ) ) ) ;
            }
        }
    }
```

Output Function:

```
    if (createProduct == 1) {
        sendOutput( msg.time(), out, product ) ;
        createProduct = 0;
    }

    if ((createProduct == 0)&&(partsReady == 1)) {
        sendOutput( msg.time(), ready, 1) ;
        partsReady = 0;
    }
```

Testing Strategy:

- if sizeBuffer1 and sizeBuffer2 > 0 : When receiving an input from port ‘input1’ and/or ‘input2’, verify that the workstation sends out a product ‘2’ or ‘3’ (based on the value of the input) after a random amount of time and at the same time, an output to the ‘ready’ port.
- Verify that the workstation sends an output to the ‘ready’ port when sizeBuffer1 and sizeBuffer2 variables are both > 0 and the Workstation21 is not working.
- Verify that the workstation does nothing when one of the sizeBuffer1 or sizeBuffer2 values are 0.
- Verify that the Workstation2 receives an input at ‘input1’ and ‘input2’ when and output is sent out of the ‘ready’ port.

Coupled Model Inspector1

Inspector1 = <X,Y,{Warehouse, Prioritizer}, EIC, EOC, IC, SELECT>

X = {numBuf1, numBuf2, numBuf3}
Y = {out1, out2, out3}
EIC = {(Inspector1.numBuf1, Prioritizer.numBuf1), (Inspector1.numBuf2, Prioritizer.numBuf2), (Inspector1.numBuf3, Prioritizer.numBuf3)}
EOC = {(Prioritizer.out1, out1), (Prioritizer.out2, out2), (Prioritizer.out3, out3)}
IC = {(Prioritizer.inspectorReady, Warehouse.in), (Warehouse.out, Prioritizer.in)}
SELECT = ({Warehouse, Prioritizer}) = Prioritizer

Testing Strategy:

- Verify that an ouput of product '1' is sent at random amounts of time.
- Verify that the model starts on it's own without any inputs.

Coupled Model Inspector2

Inspector2 = <X,Y,{Warehouse, Sorter}, EIC, EOC, IC, SELECT>

X = {numBuf2, numBuf3}
Y = {out2, out3}
EIC = {(Inspector2.numBuf2, Sorter.numBuf2), (Inspector2.numBuf3, Sorter.numBuf3)}
EOC = { (Sorter.out2, out2), (Sorter.out3, out3)}
IC = {(Sorter.inspectorReady, Warehouse.in), (Warehouse.out, Sorter.in)}
SELECT = ({Warehouse, Sorter}) = Sorter

Testing Strategy:

- Verify that product '2' is sent out of port out2 and product '3' is sent out of port out3 at random amounts of time.
- Verify that the model starts on it's own without any inputs.

OVERALL TESTING STRATEGY

- Each of the atomic models were tested individually according to their corresponding testing strategies.
- Each of the coupled models were tested individually.
- Workstation1 was tested with a Buffer
- Workstation2 was tested with 2 Buffers.
- Inspector1 was tested with a Buffer and Workstation1
- Inspector2 was tested with two Buffers and Workstation2
- Complete model was tested under the following conditions:
    - Study1: Workstation1 has a mean of 10. Workstation2 and 3 have a mean of 1000.
    - Study2: Workstation1 and 3 have a mean of 1000 and Workstation2 has a mean of 10.
    - Study3: Workstation1 and 2 have a mean of 1000 and Workstation3 has a mean of 10.
    - Study4: Workstation 1 has a mean of 20 and Workstation2 and 3 have a mean of 10.
    - Study5: All Workstations have a mean of 10.
    - Study6: Inspector1 has a mean production rate of 50 seconds and Inspector2 has a mean production rate of 2 seconds.

PART3 – RESULTS

(see the following pages for numerical results
and .ma files for the distribution of the inspecting and working time)

Study 1:

The results of this test show that if the rate of production of Workstation1 is significantly greater than the rate of production of Workstation2 and Workstation3 then Workstation1 will produce a significantly greater number of products. Since production time of Inspector1 and Inspector2 is very small compared to all of the Workstations production time, there will always be elements in the buffers and the Workstations should never have to wait for parts to create their products.

Study2 & Study3:

Similar results show here as in Study 1. If one single Workstation has a significantly greater rate of production, its product will be produced at a significantly greater rate.

Study 4:

The results of this test show that even though the rate of product production by Workstation1 is only double that of Workstation2 and Workstation3, it produces a much greater than double the amount of products. This is largely because the buffer that feeds into Workstation1 from Inspector1 has a higher priority than the buffers feeding into Workstation2 and Workstation3 from Inspector1. Therefore, Workstation1 should never be waiting for a part to begin production.

Study 5:

The results here show that when all the Workstations have the same production rate, they produce approximately the same number of products. This is again due to the fact that Inspector1 and Inspector2 are producing parts at a speed where the Workstations will always have parts to begin production.

Study 6:

Inspector1 has a slower production rate than Inspector2. This causes Workstation2 and Workstation3 to slow down production since the buffers containing parts from Inspector2 will be full causing Inspector2 to stop producing until the Workstation2 or Workstation3 receive a part from the buffers coming from Inspector1. And since the buffer feeding Workstation1 from Inspector1 has a higher priority than the buffers feeding Workstation2 and Workstation3, Workstation1 will be more productive.

The above studies show that the rate of production is highly dependant on the inspecting time in Inpector1's Prioritizer and obviously the rate at which the Workstations can produce the products. Using a simulating tool such as CD++, will allow you to experiment with various inspecting time and production rates to achieve an optimal production rate for all products.

Shanta Ramchandani
Student #: 100404100

Average	Stdev
9787.817	1083.449

0	10	172	**10172**	partout1	
0	20	256	**20256**	partout1	10084
0	28	960	**28960**	partout1	8704
0	39	186	**39186**	partout1	10226
0	48	912	**48912**	partout1	9726
0	58	294	**58294**	partout1	9382
1	8	131	**68131**	partout1	9837
1	16	630	**76630**	partout1	8499
1	27	166	**87166**	partout1	10536
1	36	698	**96698**	partout1	9532
1	45	351	**105351**	partout1	8653
1	56	480	**116480**	partout1	11129
2	6	36	**126036**	partout1	9556
2	17	613	**137613**	partout1	11577
2	24	916	**144916**	partout1	7303
2	36	136	**156136**	partout1	11220
2	46	456	**166456**	partout1	10320
2	55	859	**175859**	partout1	9403
3	6	472	**186472**	partout1	10613
3	15	471	**195471**	partout1	8999
3	25	470	**205470**	partout1	9999
3	36	808	**216808**	partout1	11338
3	44	335	**224335**	partout1	7527
3	53	914	**233914**	partout1	9579
4	3	318	**243318**	partout1	9404
4	13	638	**253638**	partout1	10320
4	24	914	**264914**	partout1	11276
4	35	870	**275870**	partout1	10956
4	45	808	**285808**	partout1	9938
4	54	318	**294318**	partout1	8510
5	4	16	**304016**	partout1	9698
5	14	989	**314989**	partout1	10973
5	23	557	**323557**	partout1	8568
5	31	840	**331840**	partout1	8283
5	42	552	**342552**	partout1	10712
5	53	259	**353259**	partout1	10707
6	1	775	**361775**	partout1	8516
6	10	93	**370093**	partout1	8318
6	21	348	**381348**	partout1	11255
6	31	964	**391964**	partout1	10616
6	40	320	**400320**	partout1	8356
6	49	619	**409619**	partout1	9299
6	59	286	**419286**	partout1	9667
7	9	138	**429138**	partout1	9852
7	19	813	**439813**	partout1	10675
7	32	431	**452431**	partout1	12618
7	43	22	**463022**	partout1	10591
7	53	651	**473651**	partout1	10629
8	4	63	**484063**	partout1	10412

8	15	84	**495084**	partout1	11021
8	23	874	**503874**	partout1	8790
8	33	233	**513233**	partout1	9359
8	41	99	**521099**	partout1	7866
8	50	779	**530779**	partout1	9680
8	59	688	**539688**	partout1	8909
9	8	606	**548606**	partout1	8918
9	18	324	**558324**	partout1	9718
9	27	889	**567889**	partout1	9565
9	38	307	**578307**	partout1	10418
9	48	569	**588569**	partout1	10262
9	57	441	**597441**	partout1	8872

				Average	Stdev
				9976	353.5534
26	174	26174	partout2		
36	400	36400	partout2	10226	
46	126	46126	partout2	9726	

					Average	Stdev
					9976	353.5534
0	28	434	**28434**	partout3		
0	38	660	**38660**	partout3	10226	
0	48	386	**48386**	partout3	9726	

0	21	172	**21172**	partout1			
0	40	898	**40898**	partout1	19726		
0	59	551	**59551**	partout1	18653		
1	20	680	**80680**	partout1	21129		
1	40	236	**100236**	partout1	19556		
2	1	813	**121813**	partout1	21577		
2	19	116	**139116**	partout1	17303		
2	40	336	**160336**	partout1	21220		
3	0	656	**180656**	partout1	20320		
3	20	59	**200059**	partout1	19403		
3	40	672	**220672**	partout1	20613		
3	59	671	**239671**	partout1	18999		
4	19	670	**259670**	partout1	19999		
4	41	8	**281008**	partout1	21338		
4	58	535	**298535**	partout1	17527		
5	18	114	**318114**	partout1	19579		
5	37	518	**337518**	partout1	19404		
5	57	838	**357838**	partout1	20320		
6	19	114	**379114**	partout1	21276		
6	40	70	**400070**	partout1	20956		
7	0	8	**420008**	partout1	19938		
7	18	518	**438518**	partout1	18510		
7	38	216	**458216**	partout1	19698		
7	59	189	**479189**	partout1	20973		
8	17	757	**497757**	partout1	18568		
8	36	40	**516040**	partout1	18283		
8	56	752	**536752**	partout1	20712		
9	17	459	**557459**	partout1	20707		
9	35	975	**575975**	partout1	18516	Average	Stdev
9	54	293	**594293**	partout1	18318	19762.79	1198.167
0	26	174	**26174**	partout2			
0	36	286	**36286**	partout2	10112		
0	46	822	**46822**	partout2	10536	10324	299.8133
0	29	127	**29127**	partout3			
0	37	626	**37626**	partout3	8499		
0	47	158	**47158**	partout3	9532	9015.5	730.4413

0	11	172	**11172**	partout1	
0	22	931	**22931**	partout1	11759
0	32	313	**32313**	partout1	9382
0	41	845	**41845**	partout1	9532
0	51	401	**51401**	partout1	9556
1	2	978	**62978**	partout1	11577
1	10	281	**70281**	partout1	7303
1	21	501	**81501**	partout1	11220
1	31	821	**91821**	partout1	10320
1	41	224	**101224**	partout1	9403
1	51	837	**111837**	partout1	10613
2	0	836	**120836**	partout1	8999
2	10	835	**130835**	partout1	9999
2	22	173	**142173**	partout1	11338
2	29	700	**149700**	partout1	7527
2	39	279	**159279**	partout1	9579
2	48	683	**168683**	partout1	9404
2	59	3	**179003**	partout1	10320
3	10	279	**190279**	partout1	11276
3	21	235	**201235**	partout1	10956
3	31	173	**211173**	partout1	9938
3	39	683	**219683**	partout1	8510
3	49	381	**229381**	partout1	9698
4	0	354	**240354**	partout1	10973
4	8	922	**248922**	partout1	8568
4	17	205	**257205**	partout1	8283
4	27	917	**267917**	partout1	10712
4	38	624	**278624**	partout1	10707
4	47	140	**287140**	partout1	8516
4	55	458	**295458**	partout1	8318
5	6	713	**306713**	partout1	11255
5	17	329	**317329**	partout1	10616
5	25	685	**325685**	partout1	8356
5	34	984	**334984**	partout1	9299
5	44	651	**344651**	partout1	9667
5	54	503	**354503**	partout1	9852
6	5	178	**365178**	partout1	10675
6	17	796	**377796**	partout1	12618
6	28	387	**388387**	partout1	10591
6	39	16	**399016**	partout1	10629
6	49	428	**409428**	partout1	10412
7	0	449	**420449**	partout1	11021
7	9	239	**429239**	partout1	8790
7	18	598	**438598**	partout1	9359
7	26	464	**446464**	partout1	7866
7	36	144	**456144**	partout1	9680
7	45	53	**465053**	partout1	8909
7	53	971	**473971**	partout1	8918
8	3	689	**483689**	partout1	9718
8	13	254	**493254**	partout1	9565
8	23	672	**503672**	partout1	10418
8	33	934	**513934**	partout1	10262

8	42	806	**522806**	partout1	8872		
8	52	713	**532713**	partout1	9907		
9	1	42	**541042**	partout1	8329		
9	12	231	**552231**	partout1	11189		
9	22	381	**562381**	partout1	10150		
9	31	241	**571241**	partout1	8860		
9	41	641	**581641**	partout1	10400	Average	Stdev
9	51	258	**591258**	partout1	9617	9860	1025.205
0	26	60	**26060**	partout2			
0	36	374	**36374**	partout2	10314		
0	45	27	**45027**	partout2	8653	9483.5	1174.504
0	28	627	**28627**	partout3			
0	39	163	**39163**	partout3	10536		
0	50	292	**50292**	partout3	11129	10832.5	419.3143

manfac_study6

						Average	Stdev
						50518.1	1479.603
1	0	839	**60839**	partout1			
1	51	94	**111094**	partout1	50255		
2	39	881	**159881**	partout1	48787		
3	29	226	**209226**	partout1	49345		
4	22	531	**262531**	partout1	53305		
5	13	698	**313698**	partout1	51167		
6	1	854	**361854**	partout1	48156		
6	52	798	**412798**	partout1	50944		
7	44	169	**464169**	partout1	51371		
8	35	450	**515450**	partout1	51281		
9	26	20	**566020**	partout1	50570		