

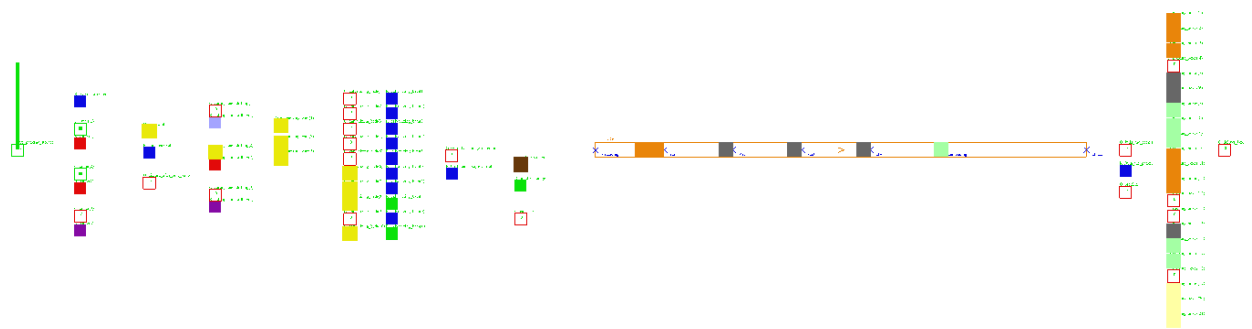
Improving the throughput at Lightning Inc.

Simulation of Production Systems

MPR 271 - 2013

Sebastian Nilsson

Pontus Wikhholm



Abstract

The project improves a street light production so that the throughput of the factory increases. AutoMod was used as simulation software and mainly AutoStat was used to analyse the model. The simulation methodology by Jerry Banks and the theory of constraints are the two main methods used in the project. The result was an increase of the throughput by 33%, reduced WIP and an increase of profit per week by 66%.

Table of Contents

1	Introduction:	4
2	Methodology	4
2.1	Step 1-2. Problem formulation and setting objectives	4
2.2	Step 3-4. Conceptual model building	4
2.3	Step 5-7. Coding, verification and validation	5
2.4	Step 8-10. Experimental design, production runs and analysis	5
2.5	Step 11. Documentation	6
2.6	Step 12. Implementation	6
3	Project realization	7
3.1	Work description (including analysis of bottlenecks)	7
3.2	Experiment design	7
3.3	Analysis of bottlenecks	9
3.4	Analysis of warm-up time	10
4	Project results	10
4.1	Improvement step 1	10
4.2	Improvement step 2	10
4.3	Extra task A: Order point optimization	11
4.4	Extra task B: Energy saving strategy	12
4.5	Extra task C: Middle storage as supermarket	13
5	Discussion	14
5.1	Discussion of the results from improvement step 2	14
5.2	No night shift	15
5.3	Bottleneck indication	15
5.4	Constraints	16
6	Conclusion	16
7	Bibliography	18
9	Appendices	19
9.1	Appendix 1: Model logic code	19
9.3	Appendix 2: Visualization of the final system	38
9.5	Appendix 3: Flow chart	39

1 Introduction:

Lightning Inc has a new production line that does not perform as expected. The purpose of this report is to serve as decision support for future investments to increase throughput and by doing so increasing profit. The investigation will be done through simulations in Automod.

The greatest delimitations in this work are only focusing on increasing profit through increased throughput. Measures to decrease buffers, selling equipment, reducing waste or reducing energy consumption to increase profit is not taken into account. Neither are any sustainability aspects treated, except for the extra task that limits waste and as a consequence results in an environmental gain.

2 Methodology

The project methodology is based on Banks model (Steps in a simulation study, (Banks, 2000)) and will also serve as a plan for experimentation.

2.1 Step 1-2. Problem formulation and setting objectives

The problem formulation is based on the project description from Examination project Simulation of Production Systems, MPR 271 - 2013. The main objective is to increase the throughput. Secondary objectives are to implement pull-flow, power savings on machines and order point optimization.

2.2 Step 3-4. Conceptual model building

To achieve a good overall view of the system and avoid deadlocks while coding we decided to express the base model in a big flow chart (see Appendix 3: Flow chart). This proved to generally improve the quality of our base model since it made it easier to collaboratively code the model simultaneously from different computers.

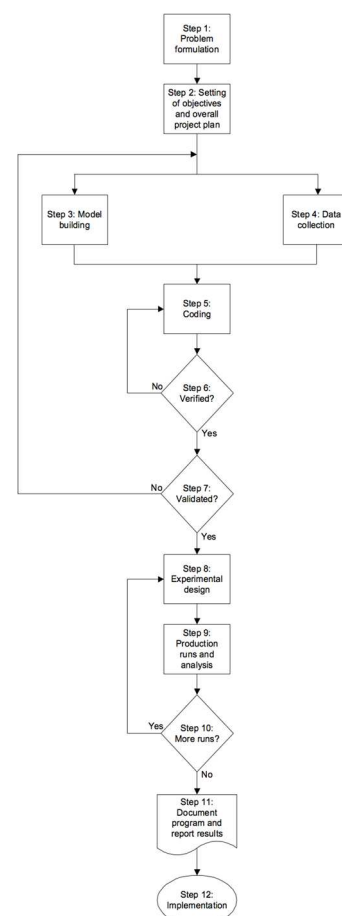


Figure 1: Jerry Bank's methodology (Banks, 2000)

2.3 Step 5-7. Coding, verification and validation

Coding the model based on the conceptual model expressed in flowcharts is a straightforward process. A good approach is to after each new process implementation run the simulation and increase complexity in steps. For example we did not implement breakdown routines until the product flow through the factory was working well without deadlocks. The verification consists of comparing our code to the conceptual model. When satisfactory behaviour is reached a validation of the whole simulation is necessary. This is to make sure the simulated production flow behaves sufficiently close to the project description.

2.4 Step 8-10. Experimental design, production runs and analysis

The ultimate goal of the experimental design was to maximize the throughput of the system. To reach this goal Theory of constraints is used (Goldratt, 1986) which is described below.

1. Apply Theory of constraints
 - a. Identify the system's constraint(s). See Methods for identifying constraints below.
 - b. Decide how to exploit the system's constraint(s). (How to get the most out of the constraint).
 - c. Subordinate other resources to the constraint(s). (Align the whole system or organization to support the decision made above).
 - d. Elevate the system's constraint(s). Make investments to further increase the constraint's capacity.
 - e. If in any of the previous steps a constraint is broken, go back to step a.

This is rather straightforward with the exception of step a. There are many ways of identifying the systems constraint as can be seen in the section below.

2.4.1 Methods for identifying constraints

Four methods are considered (S. Chick, 2003):

1. **Utilization method.** Highest utilization indicates a constraint (also called bottleneck). Greatest disadvantage is the uncertainty of the method and it lacks the ability to identify secondary bottlenecks.
2. **Waiting time method.** High waiting times in queue and length of queue before a resource indicate the resource being a bottleneck. Also empty buffers after a resource indicate being a bottleneck. Greatest disadvantage is that the queues need to be

infinite (which they are not in our model) to fully gain the capabilities of this method.

3. **Shifting bottleneck method.** A resource is either in active state (limiting flow) or passive state (not limiting flow). The resources with the longest uninterrupted active period indicate being a bottleneck. This method is superior to the other mentioned methods but will not be used because of limitations in the Automod software (see discussion in 5.3 Bottleneck indication).
4. **Average active duration method** (Roser, 2001). A resource is either in active or passive state just as in the shifting bottleneck method. The resources with the highest active percentage indicate being a bottleneck. This method is very similar to the utilization method but extends to include setup time and downtime in the active state compared to the utilization method that only focuses on the time the machine is processing.
5. **Experimental Bottleneck Detection** is the most time consuming but also the most accurate method. It is simply to run simulations for different scenarios and compare the throughput. The scenario that gives the most improved result corresponds to being a (previous) bottleneck.

In this project a combination of method one, four and five is used. For the experimental bottleneck detection we use Autostat optimization method that will vary multiple variables to find an optimal throughput. Each investment is represented as a variable and no permanent changes in the code are made to implement an investment. The Autostat optimization function will present a couple of solutions where the throughput is maximized and it is up to us to chose one of them. The other bottleneck identification methods are used to check if the answer from Autostat is reasonable or not.

2.5 Step 11. Documentation

Documentation mainly consists of comparisons of investments and their corresponding result and recommendations for future investments; in essence, this report.

2.6 Step 12. Implementation

Implementation is out of the scope of this project but we have presented our results and proposed solutions in presentation form.

3 Project realization

3.1 Work description (including analysis of bottlenecks)

3.1.1 Base model

The base model is a simulation model representing the factory's current behaviour. Before any improvements of the production system were made the throughput of the factory was 780 products per week. This answers well to the current demand of 600 lights per week. The graphic representation of the system can be seen in Figure 9 and Figure 10 (in 9.3 Appendix 2: Visualization of the final system). The products are coloured according to paint, dry-status and galvanize to aid debugging. They are also given different lengths in the beginning of the production. Discussions about assumptions are in 5 Discussion.

3.1.2 Improvements triggered by variables

Instead of testing investments by commenting out different sections of code, variables that toggle investments are created. If for example our variable `V_fast_drying_paint_investment` is set to 1 the code block that implements fast drying paint will be used instead of the original code for drying time. The reason for doing this is to make use of Autostat optimize function (also see 2.4.1 Methods for identifying constraints).

3.2 Experiment design

Experimentation on how to increase the throughput of the model was done in two steps. The first step maximize the profit while keeping WIP low with an investment of maximum 100 000 €. In the second step the whole budget of 300 000 € was used and the aim was to increase throughput as much as possible.

3.2.1 Step 1

The optimization run gives the results of the 30 highest profits per week depending on which improvements that are used. All of the 30 runs have a result of between 13000 and 18550 in profit per week. The sets of investments vary slightly between the 30 highest throughputs but most investments are either used or not used in the top 30. This is a list of the best investment setup for step 1.

1. Buy two extra fixtures to use at the assembly line.
2. Start using fast drying paint.
3. Improve the cutting and drilling equipment to increase the MTTF for that workstation.
4. Train the operators to reduce the MTTR for the cutting equipment.

5. Train the operators to reduce the MTTR for the cutting & drilling equipment.
6. Increase the speed of the conveyors in the assembly area.
7. Train the middle storage operator to inspect painting to reduce transport time for rework.
8. Invest in the fans that dry the galvanize and paint.
9. Train a worker on the assembly line station one to decrease his/her cycle time.

These investments increased the profit from the base models 11505 € to 18550 € per week. The cost for these investments are in total 92 000 €.

3.2.2 Step 2

The top 30 runs of step 2 have a result of between 1050 and 1080 in throughput per week. The sets of investments vary slightly between the 30 highest throughputs but most investments are either used or not used in the top 30. The investments that proved to be most important to maximize the output was:

1. Increase quality in cutters to eliminate the burr removal process.
2. Buy four extra fixtures to use at the assembly line.
3. Start using fast drying paint.
4. Train the operators to reduce the MTTR for the cutting equipment.
5. Train the operators to reduce the MTTR for the cutting & drilling equipment.
6. Train the operators to reduce the MTTR for the galvanizing baths.
7. Buy bigger packing boxes to replace the smaller ones.
8. Increase the speed of the conveyors in the assembly area.
9. Train the packing operator within painting to reduce transport time for painting rework.
10. Improve the cutting and drilling equipment to reduce cycle times.
11. Buy new tools to reduce the tool change time in the cutting process.
12. Improve the chemicals used in the galvanizing process to reduce the cycle times.
13. Invest in the fans that dry the galvanize and paint.
14. Train the operator working in the middle storage to reduce rework time.
15. Implement barcode usage in the package process to reduce packing time.
16. Train the workers on the assembly line station one and four to decrease their cycle times

These investments increased the throughput from the base models 780 products per week up to 1040 products per week. The cost for these investments are in total 282 000 €.

3.3 Analysis of bottlenecks

3.3.1 Multiple runs to get reliable results

Because of the stochastic nature of modelling resource uptimes, attribute settings, cycle times and so on, multiple runs are necessary to find averages. We also found a large standard deviation (see Table 1), which makes running multiple runs even more necessary.

P_packing tot	Average	5212.7
	Std. Dev.	132.97
	Minimum	4948
	Maximum	5495
	Median	5312
	# of Runs	20

Table 1: Total products in the packaging process after 25 runs

3.3.2 The base models weaknesses

The diagram below in Figure 2 (left) shows the throughput of the system after a set of improvements. This test is made with the 9 improvements that are used in step 2 in order to identify where the bottleneck is located in the base model. By definition the bottleneck is a constraint that prevents the system from achieving its goal and if the goal is high throughput the bottleneck is the resource that benefits the system the most when being improved, (Goldratt, 1986). All improvements are applied to the system separately. The three highest throughputs are done before the middle storage and the three lowest results are in or after the middle storage. This indicates that the bottleneck was located before the middle storage in the original system. It also indicates that the cutting and drilling machine is the bottleneck since the third investment is increasing the uptime of the machine.

If the same test is run with the difference of having profit as goal instead of the factory throughput (Figure 2 (right)), then the bottleneck may change as well, which is confirmed by the graph below. In fact the improvement that increased the throughput the least proved to be the most impactful factor in increasing the profit.

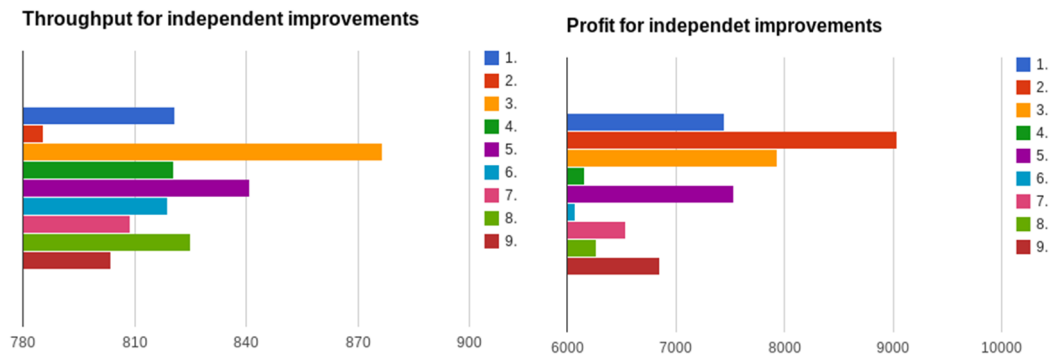


Figure 2: Bottleneck identification for throughput (left) and profit (right)

3.4 Analysis of warm-up time

The simulation is always started with no parts in any of the queues and with the resources in passive state. Loads entering in such a state will flow through the system at faster rate than loads entering during more represented times. The effect of this initial bias is eliminated by not starting data collection until the simulation reaches a steady state. Autostat warmup graph was used to find the steady state, which can be seen in Figure 3. From this graph we decided to use a warmup period of 30 hours.

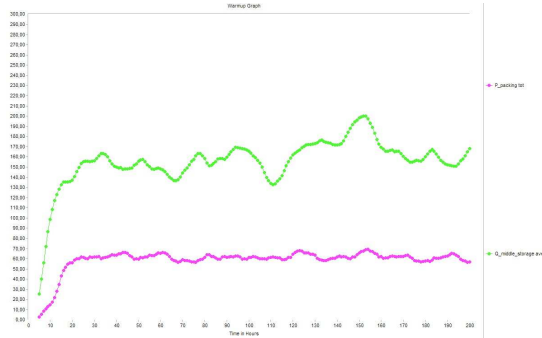


Figure 3: Warmup graph from Autostat

4 Project results

4.1 Improvement step 1

The set of improvements in step 1 (see experimental design) increases the profit per week by 60 % and decreases the amount of products in the middle storage by 47 % (see Figure 4). The decrease of average amount of products in the middle storage depends mostly on more improvements being done after the middle storage than before the middle storage. It also comes to play that several improvements have been made directly on the middle storage that decrease its cycle time.

4.2 Improvement step 2

The second step of improvements results in 1040 products per week in throughput, (up from the original 780), meaning a 33% performance increase which is high compared to the result from step 1. It is also interesting that the profit has increased by so little from step 1 to step 2. The profit can be expressed in profit per product as well to give a different picture of the profit. For step 1 it is 23 € per product and after step 2 it has decreased to 18 € per product.

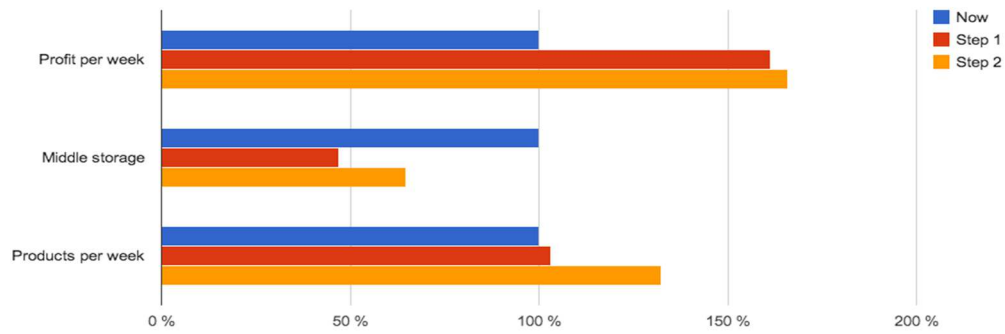


Figure 4: Results from improvement steps 1 and 2

4.3 Extra task A: Order point optimization

4.3.1 Implementation

As soon as the first storage buffer reaches a specific low limit then the system should order as many packages as would fill the buffer. The truck will arrive in a while but may not fill the buffer since packages could have been removed somewhere between order and delivery. This assumption was made to avoid having the truck arriving but not being able to load all packages. An assumption is also made that only one truck can be ordered at a time. Otherwise it would for example be possible to send one more truck as soon as one package leaves the first buffer. This would probably be costly as well.

4.3.2 Tests and result

To test the performance we did 30 runs (each with 20 replications) on the base model where the first one is with the order point strategy off and the other 29 is with varying order point from 0 to 29. At 0 an order will be sent when the buffer is completely empty and 29 as soon as 1 package leaves the first buffer. The result can be seen in Figure 5 (for buffer capacity 30) and compared to the run without order point strategy in Table x.

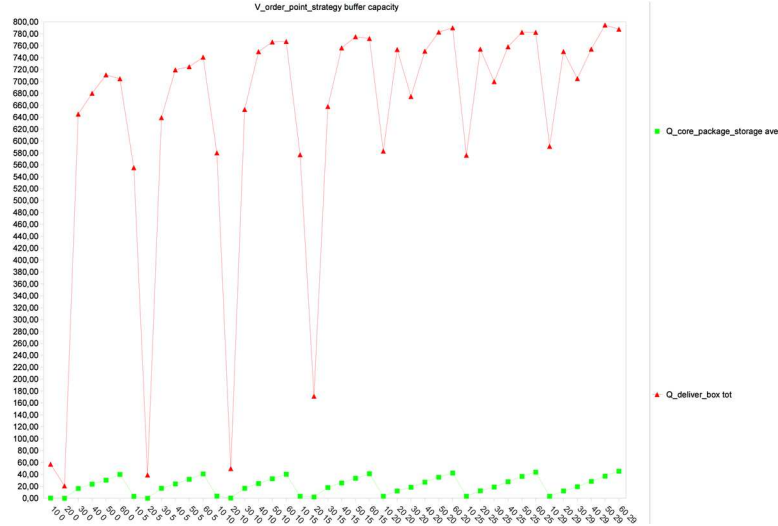


Figure 5: Order point strategy simulation

Q_deliver_box tot	Average	784,6
-------------------	---------	-------

Table 2: Without order point strategy

A trend clearly shows that ordering as soon as possible is better. Another test was run where the buffer capacity was varying from 10 to 60 at order strategies ranging from 0 to 29 generating the graph in Figure 5. The result from these runs indicates that the system performs worse for buffer capacities under 29. The result from the last simulation can be used to optimize the cost of acquiring goods. A compromise between how often to order and the cost of a high capacity buffer would be evaluated.

4.4 Extra task B: Energy saving strategy

4.4.1 Implementation

A state set with the two states `active_state` and `passive_state` is introduced. As soon as either resource `R_cutter(x)` or `R_cutting_and_drilling` is starved or blocked it is put into power save mode. When the resource is requested a check will be done to see if the resource is in power save mode and if so, start it. See Code 1. At the end of the run the consumed energy is calculated in kWh (see Equation 1) and printed to the console.

```

// start the machine if it is in power save mode
if V_cutting_and_drilling_power_save_strategy = 1
    and R_cutter(A_qcutter index) state = power_save_active then begin
        wait for 10 sec
        set R_cutter(A_qcutter index) state to power_save_inactive
    end
end

```

Code 1: Power save code implementation

$$\begin{aligned}
 \text{energy_used} = & \text{time_in_hours} \cdot (\text{power_save_consumption} \cdot \sum_{i=1}^{2+\text{extra_machine}} \text{power_save_percentage}_i \\
 & + \text{regular_consumption} \cdot \sum_{i=1}^{2+\text{extra_machine}} \text{power_save_inactive_percentage}_i)
 \end{aligned}$$

Equation 1: Energy consumption calculation

4.4.2 Result

The performance of the overall system decreased when the energy saving strategy was activated. Any improved throughput would have been unrealistic since we run the machines less. By converting the saved energy into actual saved energy cost the result could be used to find out if it would be profitable to run the machines in power save or not. In this case it is highly unlikely that this power save function is profitable since the price for electricity in Sweden is approximately 0.2 sek/kWh (SCB, 2012)

	Without power save	With power save
Packages delivered	778,2	758
Cutters (kWh)	1040	677
Cutting and drilling (kWh)	1300	856
Cutters saved (%)	-	35%
Cutting and drilling saved (%)	-	34%

Table 3: Comparison table for energy saving strategy

4.5 Extra task C: Middle storage as supermarket

4.5.1 Implementation

A conceptual model for the implementation of task C can be seen in Figure 6. A procedure will each day generate a number of orders based on the distributions from previous year's orders. An array of 3 order lists will make sure that the loads will not continue when they reach the middle storage unless an order exist that has the same length as the length of the load. When an

order is received in middle storage it will continue manufacturing according to the current orders. Every order is split into a variable array of 21, (3 product types, 3 lengths and 3 colours except for one product type). When an order is received by a load it will remove the order and send a new order to produce a core of the same length as the current length. If the cutters do not have any orders they will simply produce cores of different length from a random distribution.

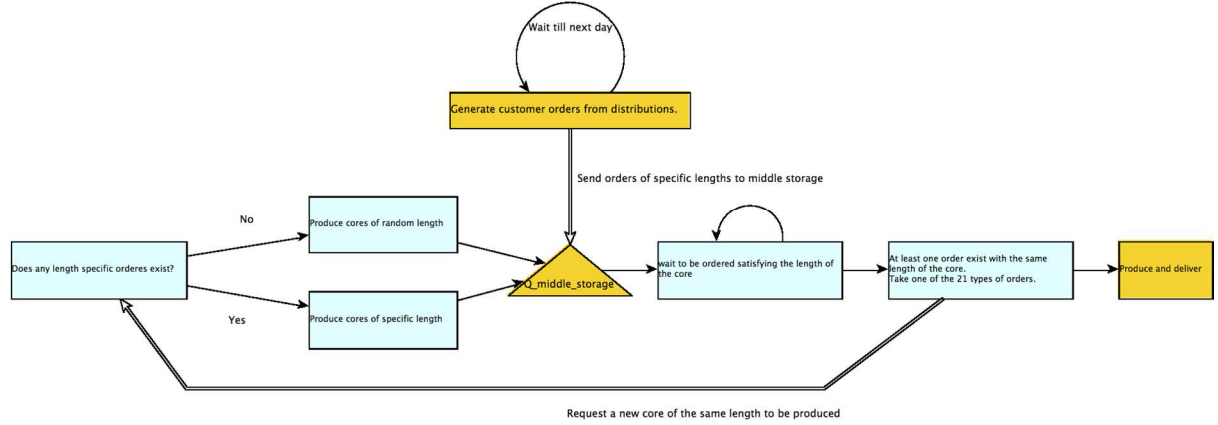


Figure 6: Flow chart

4.5.2 Result

Each day we are supposed to manufacture and deliver an average of 215 packages. With our best optimized strategy, (investment step 2), we produce 1040 products per week or 208 products per day with the factory running full time. The average number of products per week that the order demands is 215 per day. Satisfying 98% of the orders to be delivered with the truck the next day therefore become difficult.

During an 185 hours run with 8 hours warm-up, 55% of the orders requested from the market has been produced. This is rather far from satisfying the demand of 98% even though we are using the investment set that maximizes the throughput of lights from the factory.

5 Discussion

5.1 Discussion of the results from improvement step 2

If the profit per product continues to decrease when Lightning Inc. increase their production volume it will eventually become unprofitable to increase the production volume. This outcome is highly unlikely though since the costs will decrease rather than increase when increasing production volume. One big difference between step one and two is that step two has more WIP in the system which means that they have more products that the company has paid for but not received any income from.

The methodology that is used in this project is mostly obtained from Proceedings of the Winter Simulation Conference and from the Theory of constraints (Goldratt, 1986). The theory

of constraints is useful in almost any kind of production simulation project. In the factory there is a big storage in the middle of the production flow. This middle storage is highly useful as an indicator of where the systems biggest constraint is located. If the middle storage is empty the factory's bottleneck is located in the first part of the system, meaning the later part of the system performs better. If the middle storage later is full it means that the bottleneck has moved to the later part of the system. This kind of improvement process and constraint identification is what the Theory of constraints is about. In five methods of detecting a systems bottleneck are discussed. These methods are all useful in specific cases but in this project method five, four and one were used. Perhaps the best tool for us was however the optimisation function in Autostat. This provided an iterative way of finding the best solution depending on what parameter that was desired to maximize.

5.2 No night shift

We decided to model only 8 hours per day instead of modelling 24 hours with 8 hour work day and 1 hour lunch. The most important consequence of this approach is the increased cycle time in middle storage. If we had modelled 24 hours some of the cores would have been able to dry over the night. Time is a resource in any project and the only reason for simplifying and only modelling 8 hours was to save time, but looking back we are not sure if the approach brought sufficient saved time. For someone else starting this project we would recommend modelling 24 hours and introducing a state for each rescore when the factory is closed (to make sure that this time is not included in the utilization analysis).

5.3 Bottleneck indication

We had an idea about using shifting bottleneck indication to find the current bottlenecks. Automod support most of the common bottleneck identification methods but not the shifting bottleneck method. Implementing Average active duration method (Roser, Nakano, and Tanaka 2001) is straight forward by introducing active and passive states. The resources with the highest percentage being active indicate being the bottlenecks. The shifting bottleneck identification method relies on being able to answer duration a resource is active without interruption which makes implementation depend on memory of previous states and not only the current state. It would be difficult, but not impossible, to implement in Automod which is visualized in Figure x.

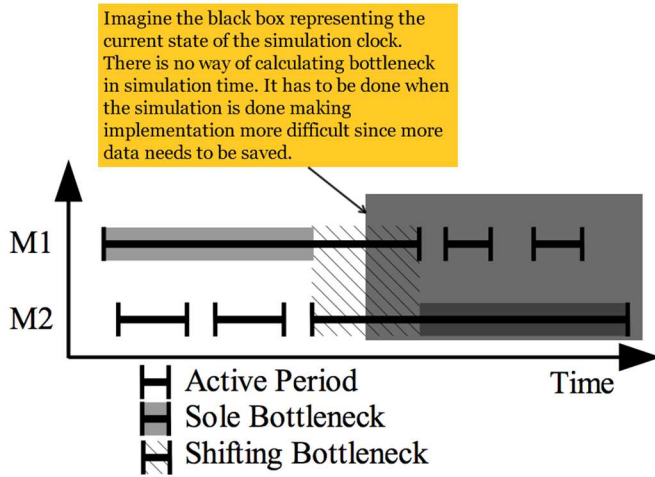


Figure 7: The difficulties with implementing shifting bottleneck detection. Original image from (Roser, 2001).

Shifting bottleneck method is considered superior and that is why we spent time sketching on a solution to implement it, (Roser, 2001). The most promising idea was to plot machine states against time with an offset on the y axis, (see Figure 8). That plot would then be used as support when implementing an algorithm to calculate duration of sole bottleneck and shifting bottleneck for each resource. It is a shame that Automod only supports the very basic bottleneck identifiers out of the box and not the shifting bottleneck method.

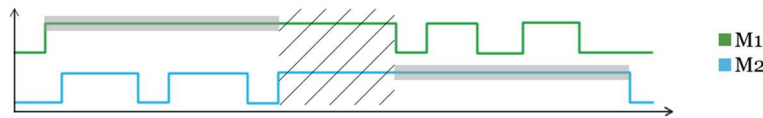


Figure 8: Sketch of how to implement shifting bottleneck with Automod basic graph capabilities

5.4 Constraints

Our main focus throughout the project was to increase the throughput and not necessarily on improving the profit. If this were an industry project it would make more sense to increase the profit since this often is more important to companies. If profit optimization was the goal more focus would have been on finding optimal buffers sizes.

6 Conclusion

One conclusion that can be taken from this project is that there are many different ways to find a production systems bottleneck. Every time a bottleneck is eliminated a new one will take place in a different place. Improvements on any other parts of a serial system than the constraint are practically wasted since it will not bring the system closer to the goal. Another conclusion is that finding bottlenecks in Automod is a tedious task and without the availability of inbuilt modern tools like Shifting bottleneck identification. The result from the simulation was an

increase of the throughput by 33%, reduced WIP and increase of profit per week by 66% which proves simulation can be used to improve a production system, assuming the model closely resembles the reality.

7 Bibliography

- Banks, C. N. (2000). *Discrete Event System Simulation*, 3rd ed.
- Goldratt, E. M. (1986). *The goal: a process of ongoing improvement*.
- Roser, C. N. (2001). A Practical Bottleneck Detection Method. In *Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 949-953, Arlington, Virginia, USA: Institute of Electrical and Electronics Engineers. .
- S. Chick, P. J. (2003). Comparison of bottleneck detection methods for AGV systems.
- SCB. (2012). *Prisutveckling på el och naturgas samt leverantörsbyten, första kvartalet 2012*. Retrieved 12 2013 from scb.se: http://www.scb.se/Statistik/EN/EN0304/2012K01/EN0304_2012K01_SM_EN24SM1202.pdf

9 Appendices

9.1 Appendix 1: Model logic code

begin model initialization function

```
// init variables

//storage variables
set V_objects_in_storage to 0
set V_painting_repair to 0
set V_painting_refill to 0

// order strategy parameters (extra task 1)
//set V_order_point_strategy to 0 // extra task 1
//set Q_core_package_storage capacity to 30
//set V_order_point to 15 // order (Q_core_package_storage remaining space) packages
when stock reaches this value
//set V_order_point_max_size to 15 // max packages in each delivery

// power saving for cutting and drilling (extra task 2)
//set V_cutting_and_drilling_power_save_strategy to 1

// order loads by forecast from market (extra task 3)
set V_Ordering_task_C to 0

if V_work_station_decrease_cycle_time_investment_step = 1 then
    set V_work_station_decrease_cycle_time_investment(1) to 1
else if V_work_station_decrease_cycle_time_investment_step = 2 then begin
    set V_work_station_decrease_cycle_time_investment(4) to 1
    set V_work_station_decrease_cycle_time_investment(1) to 1
end
else if V_work_station_decrease_cycle_time_investment_step = 3 then begin
    set V_work_station_decrease_cycle_time_investment(2) to 1
    set V_work_station_decrease_cycle_time_investment(4) to 1
    set V_work_station_decrease_cycle_time_investment(1) to 1
end
else if V_work_station_decrease_cycle_time_investment_step = 4 then begin
    set V_work_station_decrease_cycle_time_investment(1) to 2
    set V_work_station_decrease_cycle_time_investment(2) to 1
    set V_work_station_decrease_cycle_time_investment(4) to 1
end
else if V_work_station_decrease_cycle_time_investment_step = 5 then begin
    set V_work_station_decrease_cycle_time_investment(4) to 2
    set V_work_station_decrease_cycle_time_investment(1) to 2
    set V_work_station_decrease_cycle_time_investment(2) to 1
end
else if V_work_station_decrease_cycle_time_investment_step = 6 then begin
    set V_work_station_decrease_cycle_time_investment(3) to 1
    set V_work_station_decrease_cycle_time_investment(4) to 2
    set V_work_station_decrease_cycle_time_investment(1) to 2
    set V_work_station_decrease_cycle_time_investment(2) to 1
end
else if V_work_station_decrease_cycle_time_investment_step = 7 then begin
    set V_work_station_decrease_cycle_time_investment(3) to 1
    set V_work_station_decrease_cycle_time_investment(4) to 2
    set V_work_station_decrease_cycle_time_investment(1) to 2
    set V_work_station_decrease_cycle_time_investment(2) to 2
end
else if V_work_station_decrease_cycle_time_investment_step = 8 then begin
    set V_work_station_decrease_cycle_time_investment(3) to 2
    set V_work_station_decrease_cycle_time_investment(4) to 2
    set V_work_station_decrease_cycle_time_investment(1) to 2
    set V_work_station_decrease_cycle_time_investment(2) to 2
end

set convassembly.sec26 velocity to
0.4*pow(1.2,V_increase_speed_on_conveyor_investment) m per sec

//set V_extra_fixtures_investment to 0
if V_extra_fixtures_investment > 0 then
    set V_free_fixtures to V_free_fixtures + V_extra_fixtures_investment
```

```

//set capacity of Q_packing_boxes
if V_increase_packing_boxes_size_investment = 0 then begin
    set Q_packing_boxes(1) capacity to 5
    set Q_packing_boxes(2) capacity to 5
    set Q_packing_boxes(3) capacity to 5
    set Q_packing_boxes(4) capacity to 5
    set Q_packing_boxes(5) capacity to 5
    set Q_packing_boxes(6) capacity to 5
    set Q_packing_boxes(7) capacity to 5
    set Q_packing_boxes(8) capacity to 5
    set Q_packing_boxes(9) capacity to 5
    set Q_packing_boxes(10) capacity to 5
    set Q_packing_boxes(11) capacity to 5
    set Q_packing_boxes(12) capacity to 5
    set Q_packing_boxes(13) capacity to 5
    set Q_packing_boxes(14) capacity to 5
    set Q_packing_boxes(15) capacity to 5
    set Q_packing_boxes(16) capacity to 5
    set Q_packing_boxes(17) capacity to 5
    set Q_packing_boxes(18) capacity to 5
    set Q_packing_boxes(19) capacity to 5
    set Q_packing_boxes(20) capacity to 5
    set Q_packing_boxes(21) capacity to 5
end
else begin
    set Q_packing_boxes(1) capacity to 10
    set Q_packing_boxes(2) capacity to 10
    set Q_packing_boxes(3) capacity to 10
    set Q_packing_boxes(4) capacity to 10
    set Q_packing_boxes(5) capacity to 10
    set Q_packing_boxes(6) capacity to 10
    set Q_packing_boxes(7) capacity to 10
    set Q_packing_boxes(8) capacity to 10
    set Q_packing_boxes(9) capacity to 10
    set Q_packing_boxes(10) capacity to 10
    set Q_packing_boxes(11) capacity to 10
    set Q_packing_boxes(12) capacity to 10
    set Q_packing_boxes(13) capacity to 10
    set Q_packing_boxes(14) capacity to 10
    set Q_packing_boxes(15) capacity to 10
    set Q_packing_boxes(16) capacity to 10
    set Q_packing_boxes(17) capacity to 10
    set Q_packing_boxes(18) capacity to 10
    set Q_packing_boxes(19) capacity to 10
    set Q_packing_boxes(20) capacity to 10
    set Q_packing_boxes(21) capacity to 10
end

// mess with assembly workers
if V_synchronize_assembly_line_improvement = 0 then begin
    set V_convassembly_products_before_getting_material(1) to 40
    set V_convassembly_products_before_getting_material(2) to 8
    set V_convassembly_products_before_getting_material(3) to 3
    set V_convassembly_products_before_getting_material(4) to 20
end

// best one so far:
/*
set V_eliminate_burr_removal_investment to 1
set V_extra_fictures_investment to 22
set V_fast_drying_paint_investment to 1
set V_increase_speed_on_conveyor_investment to 0
set V_reduce_cutting_and_drilling_cycle_times_investment to 2
set V_reduce_cutting_cycle_times_investment to 2
set V_reduce_cutting_tool_time_change_investment to 2
set V_reduce_galvanizing_cycle_times_investment to 1
set V_work_station_decrease_cycle_time_investment(1) to 0
set V_work_station_decrease_cycle_time_investment(2) to 0
set V_work_station_decrease_cycle_time_investment(3) to 0
set V_work_station_decrease_cycle_time_investment(4) to 0

*/

/* investments */
increment C_investments by 4000*V_increase_speed_on_conveyor_investment
increment C_investments by 4000*V_extra_fictures_investment
increment C_investments by 30000*V_eliminate_burr_removal_investment

```

```

increment C_investments by 25000*V_fast_drying_paint_investment
increment C_investments by 20000*V_reduce_galvanizing_cycle_times_investment
increment C_investments by 15000*V_reduce_cutting_and_drilling_cycle_times_investment
increment C_investments by 20000*V_reduce_cutting_cycle_times_investment
increment C_investments by 25000*V_reduce_cutting_tool_time_change_investment
increment C_investments by 10000*V_increase_MTTF_cutters_investment
increment C_investments by 13000*V_increase_MTTR_cutters_investment
increment C_investments by 10000*V_increase_MTTF_cutting_and_drilling_investment
increment C_investments by 6000*V_increase_MTTR_cutting_and_drilling_investment
increment C_investments by 10000*V_increase_MTTF_galvanizing_investment
increment C_investments by 6000*V_increase_MTTR_galvanizing_investment
increment C_investments by 14000*V_reduce_painting_cycle_time_investment
increment C_investments by 6000*V_increase_painting_refill_capacity_investment
increment C_investments by 5000*V_painting_done_by_Julia_investment
increment C_investments by 5000*V_painting_inspection_done_by_Gert_investment
increment C_investments by 60000*V_extra_cutter_investment
increment C_investments by 40000*V_extra_cutting_and_drilling_investment
increment C_investments by 50000*V_automatic_painting_station_investment
increment C_investments by 20000*V_reduce_middlestorage_rework_investment
increment C_investments by 6000*V_reduce_middlestorage_cycletime_investment
increment C_investments by 10000*V_reduce_packing_cycletime_investment
increment C_investments by 5000*V_increase_packing_boxes_size_investment
increment C_investments by 15000*V_work_station_decrease_cycle_time_investment_step

// create loads
if V_autostat = 0 or (C_investments value > 200000 and C_investments value < 300000)
then begin
    create 1 load of type L_dummy to P_core_package_delivery
    create 1 load of type L_galvanizing_dummy to P_Downtime_galvanizing
    create 3 loads of type L_cutting_dummy to P_Downtime_cutting
    create 3 loads of type L_cutting_dummy to P_Downtime_cutting_and_drilling
    create 1 load of type L_conveyor_dummy to P_Downtime_conveyor
    create 10 load of type L_personell_dummy to P_Downtime_personell
    create 4 loads of type L_conveyor_work_stations_get_material_dummy to
P_conveyor_work_stations_get_material_dummy
    if V_Ordering_task_C = 1 then
        create 1 load of type L_ordering_task_C to P_Ordering_task_C
    end
else begin
    // if we break the investment limit, then set profit to 0
    set C_profit_per_week to 0
end
return 0
end

begin model snap function
    if Q_dummy value = 0 then
        return true

    //Machining cost for Cutting machines 100 € per hour processing
    increment C_cost by 100*ac/3600*(R_cutter(1) utilization)
    increment C_cost by 100*ac/3600*(R_cutter(2) utilization)

    //Machining cost for Cutting/Drilling machines 120 € per hour processing
    increment C_cost by 120*ac/3600*(R_cutting_and_drilling(1) utilization)
    increment C_cost by 120*ac/3600*(R_cutting_and_drilling(2) utilization)

    // Cost for Assembly line and Equipment 100 € per total hour
    increment C_cost by 100*ac/3600

    //Each operator costs 40 € per hour
    increment C_cost by 40*(9-V_eliminate_burr_removal_investment)*ac/3600

    //The maintenance personal cost 50 € per hour
    increment C_cost by 50*ac/3600

    // Running cost for the facility is 20000 € month
    increment C_cost by 20000/4/40*ac/3600

    // variable cost
    increment C_cost by C_variable_costs value

    set C_profit_per_week to (C_income value - C_cost value)/(ac/3600)*40

    print "C_cost: " C_cost value to message

```

```

print "C_variable_cost: " C_variable_costs value to message
print "C income:" C_income value to message
print "C investments:" C_investments value to message
print "Packages shipped: " Q_deliver_box total to message
if (ac/3600)*40*5*pow(2, V_increase_packing_boxes_size_investment) != 0 then
    print "Products shipped per week: " (Q_deliver_box total)/(ac/3600)*40*5*pow(2,
V_increase_packing_boxes_size_investment) to message
    print "Profit after " (ac/3600) " hours: " (C_income value - C_cost value) to message
    print "Profit per week: " C_profit_per_week value to message
    if Q_deliver_box total != 0 then
        print "Profit per product: " (C_income value - C_cost value)/Q_deliver_box
total to message /// Q_deliver_box value to message
        if V_cutting_and_drilling_power_save_strategy = 1 then begin
            set C_r_cutters_energy to (R_cutter(1) power_save_active average+R_cutter(2)
power_save_active average +V_extra_cutter_investment*R_cutter(3) power_save_active
average)*2*(ac/3600)
            +(R_cutter(1) power_save_inactive average+R_cutter(2)
power_save_inactive average +V_extra_cutter_investment*R_cutter(3) power_save_inactive
average)*1*(ac/3600)
            set C_r_cutting_and_drilling_energy to (R_cutting_and_drilling(1)
power_save_active average+R_cutting_and_drilling(2) power_save_active average
+V_extra_cutting_and_drilling_investment*R_cutting_and_drilling(3) power_save_active
average)*2.5*(ac/3600)
            +(R_cutting_and_drilling(1) power_save_inactive
average+R_cutting_and_drilling(2) power_save_inactive average
+V_extra_cutting_and_drilling_investment*R_cutting_and_drilling(3) power_save_inactive
average)*0.7*(ac/3600)
            print "R_cutter(1) % " R_cutter(1) power_save_active average to message
            print "R_cutter(2) % " R_cutter(2) power_save_active average to message
            print "R_cutter(3) % " R_cutter(3) power_save_active average to message
            print "R_cutting_and_drilling(1) % " R_cutting_and_drilling(1)
power_save_active average to message
            print "R_cutting_and_drilling(2) % " R_cutting_and_drilling(2)
power_save_active average to message
            print "R_cutting_and_drilling(3) % " R_cutting_and_drilling(3)
power_save_active average to message
        end
    else begin
        set C_r_cutters_energy to (2+V_extra_cutter_investment)*2*(ac/3600)
        set C_r_cutting_and_drilling_energy to
(2+V_extra_cutting_and_drilling_investment)*2.5*(ac/3600)
        end
        print "R_cutters energy " C_r_cutters_energy value " kWh" to message
        print "R_cutting_and_drilling energy " C_r_cutting_and_drilling_energy value " kWh" to
message
    return true
end

begin P_core_package_delivery arriving procedure
    move into Q_dummy

    print pow(9,5) this load to message
    // new order strategy
    if V_order_point_strategy = 1 then begin
        // assume 10 packages are already in stock
        create 10 load of type L_core_package to P_core_package_storage
        while 1=1 do begin
            /*
            an order of raw material should be placed when there are X cores left
in stock.
            From the time the order is placed and the order is received it takes
between 3 to 9 hours,
            and the most likely value is 7 hours.
            find a good order point (X) using simulation
            Expected service level of raw material for the cutters to be 98%
            */

            // if remaining space is less or equal than order point then a new
order should be placed
            if Q_core_package_storage current loads <= V_order_point then begin
                // how many packages should we order?
                set V_packages_to_order to Q_core_package_storage remaining
space

                // order and deliver

```

```

        wait for triangular 3,7,9 hr
        create V_packages_to_order load of type L_core_package to
P_core_package_storage
    end
    else begin
        /* if the stock is enough, wait for OL_cutting which is called
when stock is decreased.
        this will simply trigger a new loop */
        wait to be ordered on OL_cutting
    end
end
end

// traditional order strategy
else begin
    while 1=1 do begin
        if Q_core_package_storage remaining space < 30 then
            wait to be ordered on OL_cutting
            create 30 load of type L_core_package to P_core_package_storage
            wait for 8 hr
        end
    end
end

begin P_core_package_storage arriving procedure
    if V_Ordering_task_C = 1 then begin
        if V_middle_storage_request(1) > 0 then begin
            set A_core_length to 5
            decrement V_middle_storage_request(1) by 1
        end
        else if V_middle_storage_request(2) > 0 then begin
            set A_core_length to 6
            decrement V_middle_storage_request(2) by 1
        end
        else if V_middle_storage_request(3) > 0 then begin
            set A_core_length to 7
            decrement V_middle_storage_request(3) by 1
        end
        else
            set A_core_length to nextof(5,6,7)
        end
    else
        set A_core_length to nextof(5,6,7)

    if A_core_length = 5 then
        scale this load to z 5 ft
    else if A_core_length = 6 then
        scale this load to z 6 ft
    else
        scale this load to z 7 ft

    move into Q_core_package_storage
    send to P_core_cutting
end

begin P_core_cutting arriving procedure
    if V_extra_cutter_investment = 1 then begin
        choose a queue from among Q_cutter(1), Q_cutter(2), Q_cutter(3) whose current
loads is minimum
        save choice as A_qcutter
    end
    else begin
        choose a queue from among Q_cutter(1), Q_cutter(2) whose current loads is
minimum
        save choice as A_qcutter
    end
    move into A_qcutter
    if Q_core_package_storage remaining space >= 30 then
        order 1 load from OL_cutting to continue
    set R_cutter(A_qcutter index) state to active_state // bottleneck analysis

    if V_r_cutter_cuts(A_qcutter index) >= 100 then begin
        use R_Sven_the_cutter for uniform 5, 2 min
        set V_r_cutter_cuts(A_qcutter index) to 0
    end
end

```

```

        if V_cutter_previous_length(A_qcutter index) <> A_core_length then begin
            // lengthts unequal, we need setup
            use R_Sven_the_cutter for (n 40, 6)*pow(0.5,
V_reduce_cutting_tool_time_change_investment) sec
        end

        set A_cuts to 0
        while A_cuts < 10 do begin
            // start the machine if it is in power save mode
            if V_cutting_and_drilling_power_save_strategy = 1
                and R_cutter(A_qcutter index) state = power_save_active then begin
                wait for 10 sec
                set R_cutter(A_qcutter index) state to power_save_inactive
            end
            use R_cutter(A_qcutter index) for 82*pow(0.8,
V_reduce_cutting_cycle_times_investment) sec
            inc A_cuts by 1
            inc V_r_cutter_cuts(A_qcutter index) by 1

            if Q_burr_wait remaining space < 1 then begin
                //print "Q_burr_wait full. Wait for OL_burr_removal" this load to
message
                if V_cutting_and_drilling_power_save_strategy = 1
                    set R_cutter(A_qcutter index) state to power_save_active
                else
                    set R_cutter(A_qcutter index) state to passive_state //
bottleneck analysis
                wait to be ordered on OL_burr_removal
            end
            //print "Q_burr_wait not full. create a new load and send to P_burr" this load
to message
            clone 1 loads to P_burr new load type L_core
        end

        if Q_core_package_storage current loads > 0 then begin
            if V_cutting_and_drilling_power_save_strategy = 1 then
                set R_cutter(A_qcutter index) state to power_save_inactive
            end
            else begin
                if V_cutting_and_drilling_power_save_strategy = 1 then
                    set R_cutter(A_qcutter index) state to power_save_active
                else
                    set R_cutter(A_qcutter index) state to passive_state
                end
            end
            send to die
        end
    end

begin P_DownTime_cutting arriving procedure
    move into Q_dummy
    set A_cutting_dummy_index = nextof(1,2,3)

    // we do not need the third dummy if extra cutter is inactive
    if V_extra_cutter_investment = 0 and A_cutting_dummy_index = 3
        send to die
    while 1=1 do begin
        wait for e 4*pow(0.8, V_increase_MTF_cutters_investment) hr
        set R_cutter(A_cutting_dummy_index) state to active_state // bottleneck
analysis
        take down R_cutter(A_cutting_dummy_index)
        use R_Britta_the_problem_solver for (triangular 12, 30, 35 min)*pow(0.8,
V_increase_MTTR_cutters_investment)
        bring up R_cutter(A_cutting_dummy_index)
    end
end

begin P_burr arriving procedure
    move into Q_burr_wait
    if V_eliminate_burr_removal_investment = 0 then
        use R_burr_removal for w 1.5, 70 sec
        send to oneof(97:P_cutting_and_drilling,3:P_discard_after_burr_removal)
    end
end

begin P_cutting_and_drilling arriving procedure

    // chose a cutting and drilling machine
    if V_extra_cutting_and_drilling_investment = 1 then begin

```

```

        choose a queue from among Q_cutting_and_drilling(1), Q_cutting_and_drilling(2),
        Q_cutting_and_drilling(3) whose current loads is minimum
        save choice as A_qcutting_and_drilling_wait
    end
    else begin
        choose a queue from among Q_cutting_and_drilling(1), Q_cutting_and_drilling(2)
        whose current loads is minimum
        save choice as A_qcutting_and_drilling_wait
    end

    if Q_cutting_and_drilling(A_qcutting_and_drilling_wait index) remaining space = 0
        wait to be ordered on OL_cutting_and_drilling(A_qcutting_and_drilling_wait
index)

        // use the operator to load the machine
        use R_Bosse_the_cutter_and_driller for w 2, 40 sec
        move into Q_cutting_and_drilling(A_qcutting_and_drilling_wait index)
        // Q_burr_wait now has a spot free
        order 1 load from OL_burr_removal to continue

        // setup
        if V_cutting_and_drilling_previous_length(A_qcutting_and_drilling_wait index) <>
A_core_length then begin
            use R_Bosse_the_cutter_and_driller for u 50,10 sec
        end

        // start the machine if it is in power save mode
        if V_cutting_and_drilling_power_save_strategy = 1
            and R_cutting_and_drilling(A_qcutting_and_drilling_wait index) state =
power_save_active then begin
            wait for 10 sec
            set R_cutting_and_drilling(A_qcutting_and_drilling_wait index) state to
power_save_inactive
        end
        // use the machine
        use R_cutting_and_drilling(A_qcutting_and_drilling_wait index) for 111*pow(0.9,
V_reduce_cutting_and_drilling_cycle_times_investment) sec
        send to P_galvanizing
    end

begin P_DownTime_cutting_and_drilling arriving procedure
    move into Q_dummy
    set A_cutting_and_drilling_dummy_index = nextof(1,2,3)

    // we do not need the 3rd if extra cutting_and_drilling is inactive
    if V_extra_cutting_and_drilling_investment = 0 and A_cutting_and_drilling_dummy_index
= 3 then
        send to die
        while 1=1 do begin
            wait for e 3*pow(0.8, V_increase_MTF_cutting_and_drilling_investment) hr
            take down R_cutting_and_drilling(A_cutting_and_drilling_dummy_index)
            use R_Britta_the_problem_solver for (triangular 18, 25, 55)*pow(0.8,
V_increase_MTR_cutting_and_drilling_investment) min
            bring up R_cutting_and_drilling(A_cutting_and_drilling_dummy_index)
        end
    end

begin P_discard_after_burr_removal arriving procedure
    //print "Discard this " this load to message
    move into Q_discard after burr_removal
    order 1 load from OL_burr_removal to continue
end

begin P_galvanizing arriving procedure
    if A_core_length = 5 then
        set A_qgalvanizing_wait_index to 1
    else if A_core_length = 6 then
        set A_qgalvanizing_wait_index to 2
    else
        set A_qgalvanizing_wait_index to 3

    if Q_galvanizing_wait(A_qgalvanizing_wait_index) remaining space = 0 then begin
        // blocked so activate power save for R cutting_and_drilling
        set R_cutting_and_drilling(A_qcutting_and_drilling_wait index) state to
power_save_active
        wait to be ordered on OL_galvanizing_wait(A_qgalvanizing_wait_index)
    end
end

```

```

end
move into Q_galvanizing_wait(A_qgalvanizing_wait_index)

// cutting and drilling machine now is free
order 1 load from OL_cutting_and_drilling(A_qcutting_and_drilling_wait_index) to
continue

// check if Q_cutting_and_drilling(index) is starved and if so activate power save
mode
if Q_cutting_and_drilling(A_qcutting_and_drilling_wait_index) remaining space = 1 then
    set R_cutting_and_drilling(A_qcutting_and_drilling_wait_index) state to
    power_save_active

if Q_galvanizing_wait(A_qgalvanizing_wait_index) remaining space = 0 then begin
    if Q_galvanizing_bath(1) remaining space = 0 then
        wait to be ordered on OL_galvanizing_bath(1)
        send to P_galvanizing_bath
    end
    wait to be ordered on OL_galvanizing_batch(A_qgalvanizing_wait_index)
end

begin P_galvanizing_bath arriving procedure

// loop through all 10 baths
set A_galvanize_counter to 1
while A_galvanize_counter < 11 do begin

    if Q_galvanizing_bath(A_galvanize_counter) remaining space = 0 then
        wait to be ordered on OL_galvanizing_bath(A_galvanize_counter)
        move into Q_galvanizing_bath(A_galvanize_counter)

    if A_galvanize_counter = 1 then begin
        order all loads from OL_galvanizing_batch(A_qgalvanizing_wait_index) to
        die
        order 5 loads from OL_galvanizing_wait(A_qgalvanizing_wait_index) to
        continue
    end
    if A_galvanize_counter <> 1 then
        order 1 load from OL_galvanizing_bath(A_galvanize_counter-1) to
        continue

    //print "move into Q_galvanizing_bath " A_galvanize_counter " " this load to
    message

    // use time depends on length
    if A_core_length = 5 then
        use R_galvanizing_bath(A_galvanize_counter) for
        270*pow(0.9,V_reduce_galvanizing_cycle_times_investment) sec
    else if A_core_length = 6 then
        use R_galvanizing_bath(A_galvanize_counter) for
        300*pow(0.9,V_reduce_galvanizing_cycle_times_investment) sec
    else
        use R_galvanizing_bath(A_galvanize_counter) for
        500*pow(0.9,V_reduce_galvanizing_cycle_times_investment) sec

    inc A_galvanize_counter by 1
end

send to P_unbatching_before_middle_storage
end

begin P_Downtime_galvanizing arriving procedure
move into Q_dummy
while 1=1 do begin
    wait for e 16*pow(0.8, V_increase_MTTT_galvanizing_investment) hr
    take down R_galvanizing_bath(1)
    take down R_galvanizing_bath(2)
    take down R_galvanizing_bath(3)
    take down R_galvanizing_bath(4)
    take down R_galvanizing_bath(5)
    take down R_galvanizing_bath(6)
    take down R_galvanizing_bath(7)
    take down R_galvanizing_bath(8)
    take down R_galvanizing_bath(9)
    take down R_galvanizing_bath(10)

```

```

        use R_Britta_the_problem_solver for (w 1.5, 40)*pow(0.8,
V_increase_MTTR_galvanizing_investment) min
        bring up R_galvanizing_bath(1)
        bring up R_galvanizing_bath(2)
        bring up R_galvanizing_bath(3)
        bring up R_galvanizing_bath(4)
        bring up R_galvanizing_bath(5)
        bring up R_galvanizing_bath(6)
        bring up R_galvanizing_bath(7)
        bring up R_galvanizing_bath(8)
        bring up R_galvanizing_bath(9)
        bring up R_galvanizing_bath(10)
    end
end

begin P_unbatching_before_middle_storage arriving procedure
    set A_unbatching_before_middle_storage to 0
    while A_unbatching_before_middle_storage < 5 do begin
        inc A_unbatching_before_middle_storage by 1
        if Q_middle_storage remaining space = 0 then begin
            set R_middle_storage state to active state
            wait to be ordered on OL_middle_storage
        end
        clone 1 loads to P_middle_storage new load type L_core
    end

    order 1 load from OL_galvanizing_bath(10) to continue // last bath is now empty
    send to die
end

begin P_middle_storage arriving procedure
    move into Q_middle_storage
    set R_middle_storage state to passive_state

    // use Gert to move the load to storage
    use R_Gert_the_storage_operator for u 21, 5 sec

    // let galvanization dry
    if A_galvanize_dried = 0 then begin
        wait for 2*pow(0.9, V_reduce_middlstorage_cykletime_investment) hr
        set A_galvanize_dried to 1
    end

    // set product type
    if V_Ordering_task_C = 1 then begin
        wait to be ordered on OL_market_request

        //if A_core_length = 5 then begin
        if V_market_request(1+(A_core_length-5)) > 0 then begin
            set A_product_type to 1
            set A_color to 1
            decrement V_market_request(1+(A_core_length-5)) by 1
        end
        else if V_market_request(4+(A_core_length-5)) > 0 then begin
            set A_product_type to 1
            set A_color to 1
            decrement V_market_request(4+(A_core_length-5)) by 1
        end
        else if V_market_request(7+(A_core_length-5)) > 0 then begin
            set A_product_type to 1
            set A_color to 2
            decrement V_market_request(7+(A_core_length-5)) by 1
        end
        else if V_market_request(10+(A_core_length-5)) > 0 then begin
            set A_product_type to 3
            set A_color to 1
            decrement V_market_request(10+(A_core_length-5)) by 1
        end
        else if V_market_request(13+(A_core_length-5)) > 0 then begin
            set A_product_type to 3
            set A_color to 1
            decrement V_market_request(13+(A_core_length-5)) by 1
        end
        else if V_market_request(16+(A_core_length-5)) > 0 then begin
            set A_product_type to 3
            set A_color to 2
        end
    end
end

```

```

        decrement V_market_request(16+(A_core_length-5)) by 1
    end
    else if V_market_request(19+(A_core_length-5)) > 0 then begin
        set A_product_type to 2
        decrement V_market_request(19+(A_core_length-5)) by 1
    end
    else
        print "something is wrong" to message
        increment V_middle_storage_request(1) by 1
    //end
end
else
    set A_product_type to oneof(33.3:1,33.3:2,33.3:3)

    // for 1% galvanization needs to be reworked
    set A_rework_galvanizing to oneof(100-1*pow(0.5,
V_reduce_middlstorage_rework_investment):0, 1*pow(0.5,
V_reduce_middlstorage_rework_investment):1)
    if A_rework_galvanizing = 1 then begin
        //set priority to 0
        use R_Gert_the_storage_operator for n 6, 1 min
        //set priority to 1
    end

    // send HL and ML to painting
    if A_product_type = 1 or A_product_type = 3 then
        send to P_painting
    end
    // let the paint dry
    else if (A_paint_dried = 0 and A_painted = 1) then begin
        if V_fast_drying_paint_investment = 1
            wait for 3*pow(0.9, V_reduce_middlstorage_cykletime_investment) hr
        else
            wait for 6 hr
            set A_paint_dried to 1
            set A_rework_painting to oneof(100-3*pow(0.5,
V_reduce_middlstorage_rework_investment):0, 3*pow(0.5,
V_reduce_middlstorage_rework_investment):1)
            if A_rework_painting = 1 then
                use R_Gert_the_storage_operator for n 3, 0.4 min
            end

            // use Gert to move the load from storage
            // assign task to Gert
            /*if Q_Gert_the_storage_operator remaining space = 0
                wait to be ordered on OL_Gert_the_storage_operator
            clone 1 load to P_Gert_the_storage_operator new load type L_task*/
            use R_Gert_the_storage_operator for u 21, 5 sec
            send to P_assembly_loading
        end

begin P_Gert_the_storage_operator arriving procedure

    move into Q_Gert_the_storage_operator
    wait to be ordered on OL_Gert_the_storage_operator_done
    order 1 load from OL_Gert_the_storage_operator to continue
    send to die
end

begin P_painting arriving procedure
    set priority to 1

    //Paint the product
    use R_Gert_the_storage_operator for ((w 1.5, 50)*pow(0.9,
V_reduce_painting_cycle_time_investment))*pow(0.5, V_automatic_painting_station_investment)
    sec

    if V_Ordering_task_C = 0 then
        set A_color to oneof(60:1, 30:2, 10:3)

    if A_color = 1 then
        set color to brown
    else if A_color = 2 then
        set color to black
    else if A_color = 3 then

```

```

        set color to green
        set A_painted to 1
        inc V_painting_refill by 1
        inc V_painting_repair by 1

        if V_painting_repair = 150 then begin
            use R_Gert_the_storage_operator for n 240, 18 sec
            set V_painting_repair to 0
        end

begin
    if V_painting_refill = 20*pow(4, V_increase_painting_refill_capacity_investment) then
        use R_Gert_the_storage_operator for u 3, 1 min
        set V_painting_refill to 0
    end

    if V_automatic_painting_station_investment = 0 then
        set V_slumpen to oneof(95:0,5:1)
    else
        set V_slumpen to oneof(99:0,1:1)
    end

    if V_painting_inspection_done_by_Gert_investment = 1 and V_slumpen = 1 then begin
        //Paint the product
        use R_Gert_the_storage_operator for ((w 1.5, 50)*pow(0.9,
V_reduce_painting_cycle_time_investment))*pow(0.5, V_automatic_painting_station_investment)
        sec

        inc V_painting_refill by 1
        inc V_painting_repair by 1

        if V_painting_repair = 150 then begin
            use R_Gert_the_storage_operator for n 240, 18 sec
            set V_painting_repair to 0
        end

        if V_painting_refill = 20*pow(4,
V_increase_painting_refill_capacity_investment) then begin
            use R_Gert_the_storage_operator for u 3, 1 min
            set V_painting_refill to 0
        end
    end

    send to P_middle_storage

end

begin P_assembly_loading arriving procedure
    if A_color = 0 then
        set color to ltyellow
    else if A_color = 1 then
        set color to orange
    else if A_color = 2 then
        set color to dkgray
    else if A_color = 3 then
        set color to ltgreen

    /*move into Q_infinite
    order 1 load from OL_middle_storage to continue
    send to die*/

    while V_free_fixtures < 1 do begin
        // middle storage is blocked. check if limiting flow
        if Q_middle_storage remaining space > 0 then
            set R_middle_storage state to passive_state // not limiting flow

        wait to be ordered on OL_free_fixture
    end
    dec V_free_fixtures by 1
    move into convassembly.stalloading
    /*use R_Gert_the_storage_operator for u 21, 5 sec // Gert back to storage. not sure
    order 1 load from OL_Gert_the_storage_operator_done to continue*/
    order 1 load from OL_middle_storage to continue

    travel to convassembly.stal
    send to P_conveyor_work_stations

```

```

end

begin P_conveyor_work_stations arriving procedure
    set R_conveyor_station(1) state to active_state
    if V_convassembly_products_before_getting_material(1) = 50 then begin
        order 1 load from OL_convassembly_worker_needs_material(1)
        wait to be ordered on OL_convassembly_worker_got_material(1)
    end
    use R_conveyor_station(1) for (n 110, 11)*pow(0.8,
V_work_station_decrease_cycle_time_investment(1)) sec
    inc V_convassembly_products_before_getting_material(1) by 1
    set R_conveyor_station(1) state to passive_state

    travel to convassembly.sta2
    set R_conveyor_station(2) state to active_state
    if V_convassembly_products_before_getting_material(2) = 50 then begin
        order 1 load from OL_convassembly_worker_needs_material(2)
        wait to be ordered on OL_convassembly_worker_got_material(2)
    end
    use R_conveyor_station(2) for (n 90, 8)*pow(0.8,
V_work_station_decrease_cycle_time_investment(2)) sec
    inc V_convassembly_products_before_getting_material(2) by 1
    set R_conveyor_station(2) state to passive_state

    travel to convassembly.sta3
    set R_conveyor_station(3) state to active_state
    if V_convassembly_products_before_getting_material(3) = 50 then begin
        order 1 load from OL_convassembly_worker_needs_material(3)
        wait to be ordered on OL_convassembly_worker_got_material(3)
    end
    use R_conveyor_station(3) for (n 75, 4)*pow(0.8,
V_work_station_decrease_cycle_time_investment(3)) sec
    inc V_convassembly_products_before_getting_material(3) by 1
    set R_conveyor_station(3) state to passive_state

    travel to convassembly.sta4
    set R_conveyor_station(4) state to active_state
    if V_convassembly_products_before_getting_material(4) = 50 then begin
        order 1 load from OL_convassembly_worker_needs_material(4)
        wait to be ordered on OL_convassembly_worker_got_material(4)
    end
    use R_conveyor_station(4) for (n 95, 7)*pow(0.8,
V_work_station_decrease_cycle_time_investment(4)) sec
    inc V_convassembly_products_before_getting_material(4) by 1
    set R_conveyor_station(4) state to passive_state

    travel to convassembly.staunloading
    if Q_Julia_the_packer remaining space = 0
        wait to be ordered on OL_Julia_the_packer
    clone 1 loads to P_conveyor_unloading new load type L_core
    wait to be ordered on OL_staunloading_done

    travel to convassembly.stadie
    inc V_free_fixtures by 1
    order 1 load from OL_free_fixture to continue
    send to die
end

begin P_conveyor_work_stations_get_material_dummy arriving procedure
    move into Q_dummy
    set A_convassembly_worker_index to nextof(1,2,3,4)
    while 1=1 do begin
        if
V_convassembly_products_before_getting_material(A_convassembly_worker_index) < 50 then
            wait to be ordered on
OL_convassembly_worker_needs_material(A_convassembly_worker_index)
            set R_conveyor_station(A_convassembly_worker_index) state to active_state
            use R_conveyor_station(A_convassembly_worker_index) for u 7, 3 min // #
            set
V_convassembly_products_before_getting_material(A_convassembly_worker_index) to 0
            order 1 load from
OL_convassembly_worker_got_material(A_convassembly_worker_index) to continue
        end
    end
end

```

```

end

begin P_conveyor_unloading arriving procedure
  /*move into Q infinite
  order 1 load from OL_staunloading_done to continue
  send to die*/

  move into Q Julia the packer
  use R_Julia_the_packer for u 17.5, 2.5 sec
  order 1 load from OL_staunloading_done to continue
  //move into Q_packing
  send to P_inspection
end

begin P_DownTime_conveyor arriving procedure
  move into Q dummy
  while 1=1 do begin
    wait for e 16 hr
    take down convassembly.M_sec1
    use R_Britta_the_problem_solver for w 1.5, 50 min
    bring up convassembly.M_sec1
  end
end

begin P_inspection arriving procedure
  use R_Julia_the_packer for u 50, 10 sec

  //set an attribute to track the parts and send them to the painting process, rework or
  packing.
  set A_part_just_visiting_from_inspection to 1
  //send to P_packing
  send to oneof(3:P_rework_packing,5: P_repainting, 92:P_packing)
  if A_product_type <> 2 then begin
    if V_automatic_painting_station_investment = 0 then
      send to oneof(3:P_rework_packing,5: P_repainting, 92:P_packing)
    else
      send to oneof(3:P_rework_packing,1: P_repainting, 96:P_packing)
    end
  else
    send to oneof(3:P_rework_packing, 97:P_packing)
  end
end

begin P_repainting arriving procedure
  if V_painting_inspection_done_by_Gert_investment = 0 then begin
    set priority to 0

    //Lift product to painting
    if V_painting_done_by_Julia_investment= 0 then
      use R_Julia_the_packer for u 2.5, 0.5 min

      //Paint the product
      if V_painting_done_by_Julia_investment = 0 then
        use R_Gert_the_storage_operator for ((w 1.5, 50)*pow(0.9,
V_reduce_painting_cycle_time_investment))*pow(0.5, V_automatic_painting_station_investment)
sec
      else
        use R_Julia_the_packer for ((w 1.5, 50)*pow(0.9,
V_reduce_painting_cycle_time_investment))*pow(0.5, V_automatic_painting_station_investment)
sec

        inc V_painting_refill by 1
        inc V_painting_repair by 1

        if V_painting_repair = 150 then begin
          use R_Gert_the_storage_operator for n 240, 18 sec
          set V_painting_repair to 0
        end

        if V_painting_refill = 20*pow(4,
V_increase_painting_refill_capacity_investment) then begin
          use R_Gert_the_storage_operator for u 3, 1 min
          set V_painting_refill to 0
        end
      end
    end
  end
end

```

```

        //Lift product from painting
        if V_painting_done_by_Julia_investment = 0 then
            use R_Julia_the_packer for u 2.5, 0.5 min

            set priority to 1
        end

        send to P_packing

    end

begin P_rework_packing arriving procedure
    use R_Julia_the_packer for e 60 sec
    send to P_packing
end

begin P_packing arriving procedure

    if A_core_length = 5 then
        set A_core_length_index to 1
    else if A_core_length = 6 then
        set A_core_length_index to 2
    else
        set A_core_length_index to 3

        // CL are not painted but should be given an index
        if A_color = 0 then
            set A_color_index to 1
        else
            set A_color_index to A_color

        if A_product_type = 2
            set A_product_type_index to 3
        else if A_product_type = 3
            set A_product_type_index to 2
        else
            set A_product_type_index to 1

        if A_product_type = 2 then
            set A_color_index to 1

        set A_packing_boxes_index to (A_product_type_index-1)*9 + (A_color_index-1)*3 +
(A_core_length_index-1) + 1

        if Q_packing_boxes(A_packing_boxes_index) remaining space = 0 then
            wait to be ordered on OL_packing_boxes(A_packing_boxes_index)
            move into Q_packing_boxes(A_packing_boxes_index)
            order 1 load from OL_Julia_the_packer to continue

        if Q_packing_boxes(A_packing_boxes_index) remaining space = 0 then begin
            send to P_deliver_box
        end
        wait to be ordered on OL_packing_boxes_batch(A_packing_boxes_index)

    end

end

begin P_deliver_box arriving procedure
    use R_Julia_the_packer for (u 2,1)*pow(0.75, V_reduce_packing_cykletime_investment)
min
    move into Q_deliver box
    if V_increase_packing_boxes_size_investment = 0 then
        order 4 loads from OL_packing_boxes_batch(A_packing_boxes_index) to die
    else
        order 9 loads from OL_packing_boxes_batch(A_packing_boxes_index) to die

    order 1 load from OL_packing_boxes(A_packing_boxes_index) to continue

    // raw material, assembled components, galvanizing
    set A_direct_cost to 70+100+10
    // paint cost for every light except CL
    if A_product_type <> 2 then
        inc A_direct_cost by 5

    if A_repainted = 1 then
        inc A_direct_cost by 5

```

```

set A_direct_cost to A_direct_cost*5*pow(2, V_increase_packing_boxes_size_investment)

// 220, 270, 270 per pair of ML, CL, HL
if A_product_type = 1 then
    set A_income to 220*5*pow(2, V_increase_packing_boxes_size_investment)
else
    set A_income to 270*5*pow(2, V_increase_packing_boxes_size_investment)
inc C_income by A_income
inc C_variable_costs by A_direct_cost
send to die
end

begin P_DownTime_personell arriving procedure
    move into Q_dummy
    set A_personell_index = nextof(1,2,3,4,5,6,7,8,9,10)
    if A_personell_index = 1 then
        set A_rperson to R_Bosse_the_cutter_and_driller
    else if A_personell_index = 2 then
        set A_rperson to R_Britta_the_problem_solver
    else if A_personell_index = 3 then
        set A_rperson to R_Gert_the_storage_operator
    else if A_personell_index = 4 then
        set A_rperson to R_Julia_the_packer
    else if A_personell_index = 5 then
        set A_rperson to R_Sven_the_cutter
    else if A_personell_index = 6 then
        set A_rperson to R_burr_removal
    else if A_personell_index = 7 then
        set A_rperson to R_conveyor_station(1)
    else if A_personell_index = 8 then
        set A_rperson to R_conveyor_station(2)
    else if A_personell_index = 9 then
        set A_rperson to R_conveyor_station(3)
    else if A_personell_index = 10 then
        set A_rperson to R_conveyor_station(4)

    while 1=1 do begin
        wait for w 2,4 hr
        take down A_rperson
        wait for n 5, 0.5 min
        bring up A_rperson
    end
end

begin P_Ordering_task_C arriving procedure
    move into Q_dummy
    wait for 15 hr // 8h warmup, +7 h to order at 15:00

    while 1=1 do begin
        //Lights are given types
        set V_lights_to_order(1) to gamma 10.000000, 9.828000
        set V_lights_to_order(2) to gamma 42.045511, 0.923999
        set V_lights_to_order(3) to gamma 57.050206, 1.038909

        //The same number of lights that has gotten a type above is also given a
length.
        while (V_lights_to_order(1) + V_lights_to_order(2) +
V_lights_to_order(3)) != (V_lights_to_order(4) + V_lights_to_order(5) + V_lights_to_order(6))
do begin
            set V_lights_to_order(4) to 14.995349 + weibull 1.088598, 75.998416
            set V_lights_to_order(5) to uniform 33.500000, 33.500000
            set V_lights_to_order(6) to weibull 1.019285, 71.596129
        end

        //The same number of lights that has gotten a type above is also given a color.
        while (V_lights_to_order(1) + V_lights_to_order(2)) != (V_lights_to_order(7) +
V_lights_to_order(8) + V_lights_to_order(9)) do begin//and V_lights_to_order(7)%5!=0 and
V_lights_to_order(8)%5!=0 and V_lights_to_order(9)%5!=0 do begin
            set V_lights_to_order(7) to weibull 1.217632, 31.022829
            set V_lights_to_order(8) to weibull 1.170872, 23.941400
            set V_lights_to_order(9) to lognormal 4.485275, 0.343770
        end

        while V_lights_to_order(1)>0 do begin
            if V_lights_to_order(4) > 0 then begin
                if V_lights_to_order(7) > 0 then begin

```

```

        inc V_lights_to_order_2(1) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(4) by 1
        dec V_lights_to_order(7) by 1
    end
    else if V_lights_to_order(8) > 0 then begin
        inc V_lights_to_order_2(2) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(4) by 1
        dec V_lights_to_order(8) by 1
    end
    else if V_lights_to_order(9) > 0 then begin
        inc V_lights_to_order_2(3) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(4) by 1
        dec V_lights_to_order(9) by 1
    end
end
end
else if V_lights_to_order(5) > 0 then begin
    if V_lights_to_order(7) > 0 then begin
        inc V_lights_to_order_2(4) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(7) by 1
    end
    else if V_lights_to_order(8) > 0 then begin
        inc V_lights_to_order_2(5) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(8) by 1
    end
    else if V_lights_to_order(9) > 0 then begin
        inc V_lights_to_order_2(6) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(9) by 1
    end
end
end
else if V_lights_to_order(6) > 0 then begin
    if V_lights_to_order(7) > 0 then begin
        inc V_lights_to_order_2(7) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(7) by 1
    end
    else if V_lights_to_order(8) > 0 then begin
        inc V_lights_to_order_2(8) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(8) by 1
    end
    else if V_lights_to_order(9) > 0 then begin
        inc V_lights_to_order_2(9) by 1
        dec V_lights_to_order(1) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(9) by 1
    end
end
end
end
while V_lights_to_order(2)>0 do begin
    if V_lights_to_order(4) > 0 then begin
        if V_lights_to_order(7) > 0 then begin
            inc V_lights_to_order_2(10) by 1
            dec V_lights_to_order(2) by 1
            dec V_lights_to_order(4) by 1
            dec V_lights_to_order(7) by 1
        end
        else if V_lights_to_order(8) > 0 then begin
            inc V_lights_to_order_2(11) by 1
            dec V_lights_to_order(2) by 1
            dec V_lights_to_order(4) by 1
            dec V_lights_to_order(8) by 1
        end
        else if V_lights_to_order(9) > 0 then begin
            inc V_lights_to_order_2(12) by 1
        end
    end
end
end

```

```

        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(4) by 1
        dec V_lights_to_order(9) by 1
    end
end
else if V_lights_to_order(5) > 0 then begin
    if V_lights_to_order(7) > 0 then begin
        inc V_lights_to_order_2(13) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(7) by 1
    end
    else if V_lights_to_order(8) > 0 then begin
        inc V_lights_to_order_2(14) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(8) by 1
    end
    else if V_lights_to_order(9) > 0 then begin
        inc V_lights_to_order_2(15) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(5) by 1
        dec V_lights_to_order(9) by 1
    end
end
end
else if V_lights_to_order(6) > 0 then begin
    if V_lights_to_order(7) > 0 then begin
        inc V_lights_to_order_2(16) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(7) by 1
    end
    else if V_lights_to_order(8) > 0 then begin
        inc V_lights_to_order_2(17) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(8) by 1
    end
    else if V_lights_to_order(9) > 0 then begin
        inc V_lights_to_order_2(18) by 1
        dec V_lights_to_order(2) by 1
        dec V_lights_to_order(6) by 1
        dec V_lights_to_order(9) by 1
    end
end
end
end

while V_lights_to_order(3)>0 do begin
    if V_lights_to_order(4) > 0 then begin
        inc V_lights_to_order_2(19) by 1
        dec V_lights_to_order(3) by 1
        dec V_lights_to_order(4) by 1
    end
    else if V_lights_to_order(5) > 0 then begin
        inc V_lights_to_order_2(20) by 1
        dec V_lights_to_order(3) by 1
        dec V_lights_to_order(5) by 1
    end
    else if V_lights_to_order(6) > 0 then begin
        inc V_lights_to_order_2(21) by 1
        dec V_lights_to_order(3) by 1
        dec V_lights_to_order(6) by 1
    end
end
end

while V_lights_to_order_2(1)>0 do begin
    dec V_lights_to_order_2(1) by 5
    increment V_market_request(1) by 5
end
while V_lights_to_order_2(4)>0 do begin
    dec V_lights_to_order_2(4) by 5
    increment V_market_request(2) by 5
end
while V_lights_to_order_2(7)>0 do begin
    dec V_lights_to_order_2(7) by 5
    increment V_market_request(3) by 5
end

```

```

end

while V_lights_to_order_2(2)>0 do begin
    dec V_lights_to_order_2(2) by 5
    increment V_market_request(4) by 5
end
while V_lights_to_order_2(5)>0 do begin
    dec V_lights_to_order_2(5) by 5
    increment V_market_request(5) by 5
end
while V_lights_to_order_2(8)>0 do begin
    dec V_lights_to_order_2(8) by 5
    increment V_market_request(6) by 5
end

while V_lights_to_order_2(3)>0 do begin
    dec V_lights_to_order_2(3) by 5
    increment V_market_request(7) by 5
end
while V_lights_to_order_2(6)>0 do begin
    dec V_lights_to_order_2(6) by 5
    increment V_market_request(8) by 5
end
while V_lights_to_order_2(9)>0 do begin
    dec V_lights_to_order_2(9) by 5
    increment V_market_request(9) by 5
end

while V_lights_to_order_2(10)>0 do begin
    dec V_lights_to_order_2(10) by 5
    increment V_market_request(10) by 5
end
while V_lights_to_order_2(13)>0 do begin
    dec V_lights_to_order_2(13) by 5
    increment V_market_request(11) by 5
end
while V_lights_to_order_2(16)>0 do begin
    dec V_lights_to_order_2(16) by 5
    increment V_market_request(12) by 5
end

while V_lights_to_order_2(11)>0 do begin
    dec V_lights_to_order_2(11) by 5
    increment V_market_request(13) by 5
end
while V_lights_to_order_2(14)>0 do begin
    dec V_lights_to_order_2(14) by 5
    increment V_market_request(14) by 5
end
while V_lights_to_order_2(17)>0 do begin
    dec V_lights_to_order_2(17) by 5
    increment V_market_request(15) by 5
end

while V_lights_to_order_2(12)>0 do begin
    dec V_lights_to_order_2(12) by 5
    increment V_market_request(16) by 5
end
while V_lights_to_order_2(15)>0 do begin
    dec V_lights_to_order_2(15) by 5
    increment V_market_request(17) by 5
end
while V_lights_to_order_2(18)>0 do begin
    dec V_lights_to_order_2(18) by 5
    increment V_market_request(18) by 5
end

while V_lights_to_order_2(19)>0 do begin
    dec V_lights_to_order_2(19) by 5
    increment V_market_request(19) by 5
end
while V_lights_to_order_2(20)>0 do begin
    dec V_lights_to_order_2(20) by 5
    increment V_market_request(20) by 5
end
while V_lights_to_order_2(21)>0 do begin

```

```

        dec V_lights_to_order_2(21) by 5
        increment V_market_request(21) by 5
    end

    set V_lights_to_order_2(1) to 0
    set V_lights_to_order_2(2) to 0
    set V_lights_to_order_2(3) to 0
    set V_lights_to_order_2(4) to 0
    set V_lights_to_order_2(5) to 0
    set V_lights_to_order_2(6) to 0
    set V_lights_to_order_2(7) to 0
    set V_lights_to_order_2(8) to 0
    set V_lights_to_order_2(9) to 0
    set V_lights_to_order_2(10) to 0
    set V_lights_to_order_2(11) to 0
    set V_lights_to_order_2(12) to 0
    set V_lights_to_order_2(13) to 0
    set V_lights_to_order_2(14) to 0
    set V_lights_to_order_2(15) to 0
    set V_lights_to_order_2(16) to 0
    set V_lights_to_order_2(17) to 0
    set V_lights_to_order_2(18) to 0
    set V_lights_to_order_2(19) to 0
    set V_lights_to_order_2(20) to 0
    set V_lights_to_order_2(21) to 0

    set A_iloop to 1
    while A_iloop <= 21 do begin
        increment C_market_requests_total by V_market_request(A_iloop)
        increment A_iloop by 1
    end

    order V_market_request(1)+V_market_request(4)+V_market_request(7)
        +V_market_request(10)+V_market_request(13)+V_market_request(16)
        +V_market_request(19) loads satisfying A_core_length = 5 from
OL_market_request to continue
        in case order not filled backorder on OL_market_request
    order V_market_request(2)+V_market_request(5)+V_market_request(8)
        +V_market_request(11)+V_market_request(14)+V_market_request(17)
        +V_market_request(20) loads satisfying A_core_length = 6 from
OL_market_request to continue
    order V_market_request(3)+V_market_request(6)+V_market_request(9)
        +V_market_request(12)+V_market_request(15)+V_market_request(18)
        +V_market_request(21) loads satisfying A_core_length = 7 from
OL_market_request to continue

    //wait for 1 day
    wait for 12 hr
end
end

```

9.3 Appendix 2: Visualization of the final system

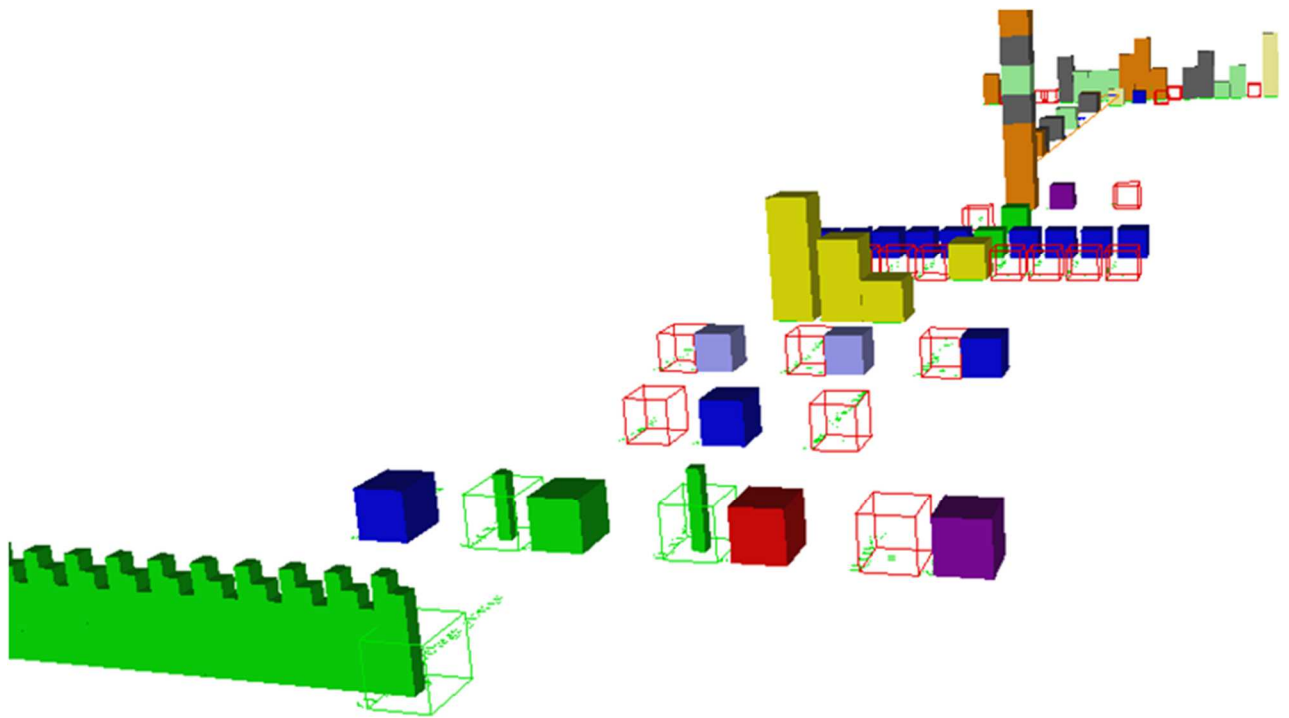


Figure 9: The model during simulation

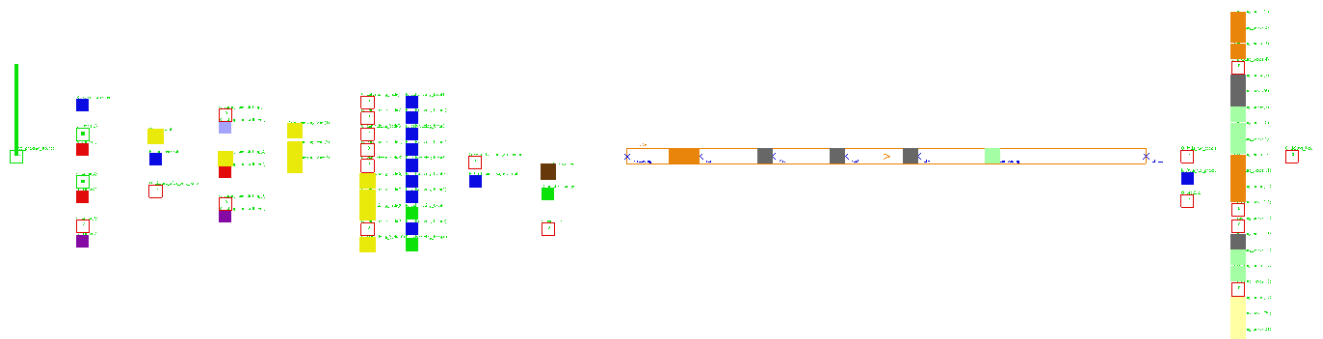


Figure 10: Overview of the system. Column by column from the left is: storage, cutters, burr removal, cutting and drilling, batching for galvanization, galvanization (starting from top and moving downwards), storage operator, middle storage, assembly line, packaging operator, bathing before delivery (sorted into 21 types) and the last one is delivery queue before they are sent to die.

9.5 Appendix 3: Flow chart

Because of the size of the flow chart it is only readable in the pdf format of this report. To view different details use the zoom function of your pdf reader software. Printing the report will make reading the flow chart impossible.

