PSIMA Levelpacks and Levels

Schumacher B., Balzar F.; 02.05.2018

Level	**1-1**	**Title:** Single Process
Task	Create a Source and a Drain. Place a SingleProc in between and link them to each other as follows: Source – SingleProc – Drain. Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters		
Purpose	Setting up the first model (drag and drop, connecting, run the simulation)	
Popup Text	Congratulations! You have just created and executed your first simulation model. Go ahead and find out about all the other possibilities PlantSimulation has to offer!	

Level	**1-2**	**Title:** Single Process - Serial
Task	Create a Source and a Drain. Place two SingleProc in between, rename them to M1 and M2 and link them to each other as follows: Source – M1 – M2 – Drain. Set the ProcTime of M1 and M2 to 1 min and 3 min and run the simulation for 1 hour. Check the results after the simulation has finished. Understand the entries for the states of M1 and M2.	
KPI	Working percentage in M1 and M2	
Parameters	M1.ProcTime	1 min
	M2.ProcTime	3 min
Purpose	Change parameters of the stations	
Popup Text	The production line is only as productive as its weakest or slowest piece. Maybe we can work around that with a parallel arrangement?	

Level	**1-3**	**Title:** Parallel Process Station
Task	Create a Source and a Drain. Place a ParallelProc in between and link them to each other as shown in the figure. Set the ProcTime of ParallelProc to 1 min and the dimensions to x = 1 and y = 2. Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters	ParallelProc.ProcTime	1 min
	ParallelProc.XDim	1
	ParallelProc.YDim	2
Purpose	Use Parallel Process Station	
Popup Text	The ProcTime of the ParallelProc's individual stations can be assigned differently via a table. The ParallelProc substitutes only parallel SingleProcs. Nevertheless, you will not use it very often.	

Level	**1-4**	**Title:** Methods – Change Attributes
Task	Create a Source and a Drain. Place a ParallelProc in between and link them to each other as shown in the figure. Set the ProcTime of ParallelProc to 1 min and the dimensions to x = 1 and y = 2. Add a method and rename it to „init“. Provide the following code: is do ParallelProc.yDim := 5; end; Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters	ParallelProc.ProcTime	1 min
	ParallelProc.XDim	1
	ParallelProc.YDim	2
Purpose	Change attributes via methods	
Popup Text	You can change the attributes of a station or MU via methods. As you can imagine, this gives you the opportunity to implement behaviors based on different situations. You will see plenty of this in the upcoming levels.	

Level	**1-5**	**Title:** Assembly Station
Task	Create two Sources and a Drain. Place an Assembly in between and link them to each other as shown in the figure. Change the MU type of the first source to „Container“. The assembly mode is „Attach MUs“. This means that the entity is now bound to the container. The main MU is from predecessor #1. Run the simulation for 1 hour. Check the Mean Life Time of the MUs and explain it.	
KPI	Throughput per hour in drain	
Parameters	Assembly.ProcTime	1 min
	Assembly.AssemblyMode	Attach MUs
	Source_Container.Path	.MUs.Container
Purpose	Change the MU type in the source, use the assembly station, use predecessors	
Popup Text	You just learned about predecessors. In complex production systems, it will be crucial to keep an eye on them when troubleshooting your model.	

Level	**1-6**	**Title:** Dismantle Station
Task	Add another Drain and a Dismantle station. Link and name (!) them according to the figure. The dismantle mode is "Detach MUs". In the assembly station, set the number of entities per container to 2 (Assembly table: Predecessors - Open - Predecessor 2, Number 2). In Dismantle, the main MU goes to Drain_Container (now mind the number of the successor...). Run the simulation for 1 hour. Check the throughput in Drain_Entity.	
KPI	Throughput per hour in drains	
Parameters	Assembly.ProcTime	1 min
	Assembly.AssemblyMode	Attach MUs
	Dismantle.ProcTime	1 min
	Dismantle.DismantleMode	Detach MUs
	Source_Container.Path	.MUs.Container
Purpose	Use the assembly table, use the dismantle station, use successors	

Popup Text	With introducing the successors, you have everything you need to build branched production lines. The assembly table allows you to implement a wide range of product variety.

Level	**1-7**	**Title:** Buffer
Task	Set the ProcTime of M1 to 1 min. Set the ProcTime of M2 to a uniform distribution with the start value 0:30 and the stop value 2:00. Set the dimension of the Buffer to 5. Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters	M1.ProcTime	1 min
	Buffer.Capacity	5
	M2.ProcTime	Uniform: [0:30, 2:00]
Purpose	Use the buffer, use the uniform distribution as ProcTime	
Popup Text	There is hardly a real production line without buffers. But be careful to use only those capacities you need. A big buffer can't solve a lack in productivity.	

Level	**1-8**	**Title:** Line
Task	Set the ProcTime of M1 and M2 to a uniform distribution with the start value 0:30 and the stop value 2:00. Set the length of the Line to 10 m and the speed to 0.5 m/s. Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters	M1.ProcTime	Uniform: [0:30, 2:00]
	Line.Length	10 m
	Line.Speed	0.5 m/s
	M2.ProcTime	Uniform: [0:30, 2:00]
Purpose	Use the line to transport entities between stations	
Popup Text	Since connectors are only a logical connection between stations and do not require time, material flow is often modeled using conveyors such as lines or tracks (for transporters to drive on, we will learn that further on).	

Level	**1-9**	**Title:** Store
Task	Set the ProcTime of M1 to 1 min. Set the dimensions of the Store to x = 3 and y = 3. Run the simulation for 1 hour. Check the throughput in the drain.	
KPI	Throughput per hour in drain	
Parameters	M1.ProcTime	1 min
	Store.XDim	3
	Store.YDim	3
Purpose	Use the Store, learn its shortcomings	
Popup Text	You have seen that MUs can't leave the store on their own via a logical connector. That is where methods are required. You can pull out MUs out of the store with them.	

Level	**1-10**	**Title:** Method	
Task	Create a Method and rename it to StoreCtrl. Set the ProcTime of M1 to 1 min. Set the dimensions of the store to x = 3 and y = 3. Set the method StoreCtrl as entrance control of the store. In the method, write „Store.Cont.move(Drain);“ as the only line. Run the simulation for 1 hour. Check the throughput in the drain.		
KPI	Throughput per hour in Drain		
Parameters	M1.ProcTime		1 min
	Store.XDim		3
	Store.YDim		3
Purpose	Introducing the method		
Popup Text	Of course you will not use the store in this way, but you get the idea how to direct MUs via methods.		

Level	**1-11**	**Title:** Complex production line	

Task	Create a Source and a Drain. Place an Assembly station, a SingleProc, a Dismantle station, a second Assembly station and a Line in between and link them to each other. Rename them to „S_parts“, „Load“, „M1“, „Unload“, „Packaging“ and „Line“.
	
	Create a second Source („S_pack“), which produces MUs of the type „Container“ for the Packaging station. In Packaging, load 4 parts on each container.
	
	Create a Method and rename it to "init". Paste the following source code:
	
	is
	  i : integer;
	do
	  for i := 1 to 4 loop
	  .MUs.Container.create(Buffer);
	  next;
	end;
	
	The pallets from the buffer are the transportation cases for the parts. Their path leads from the buffer to the Load station (1 part per container). They get dismantled from the part at the Unload station. They go back to the buffer to transport a new part.
	Set the station parameters according to the table below.
	Be careful with successors and predecessors.
	
	Run the simulation for 1 day. Check the throughput in the drain.		

KPI	Throughput per day in Drain		
Parameters	Load.ProcTime		1 min
	Load.AssemblyMode		Attach MUs
	M1.ProcTime		1 min
	Unload.ProcTime		1 min
	Unload.DismantleMode		Detach MUs
	Packaging.ProcTime		1 min
	Packaging.AssemblyMode		Attach MUs
	Line.Length		10 m
	Line.Speed		1 m/s
	Buffer.Capacity		4
Purpose	Utilize previously learned skills, init method		
Popup Text	Well done, you just modelled a complex production line. The init method allows you to set parameters before the simulation begins. You can also set some MUs up on any		

	station at the beginning of the simulation. It might be useful in the next levels. Now move on to the second level pack.

A2 - Levelpack 2

Level	**2-1**	**Title:** Designators
Task	The designator „?“ which is used in programming code, refers to the object that called the method. By this, a method can handle behaviors of different objects with the same code attached to it. The designator „@“ refers to the MU which triggered the method. Let’s try this out: Create a production line with a Source, a Drain, a SingleProc „M1“ and two Methods „Entrance“ and „Exit“. In the method „Entrance“, we need the following source code: is do ?.ProcTime := 60; end; In the method „Exit“, we need also a move command, since we interfere with the normal exit procedure: is do ?.ProcTime := 600; @.move; end; Attach the first method to the entrance of M1 and the second to its exit. Run the simulation for 1 hour. Check the throughput in the drain. Did the station require 60 or 600 seconds as the ProcTime?	
KPI	Throughput per hour in Drain	
Parameters	M1.ProcTime	1 min
Purpose	Designators, use methods to control material flow behavior	
Popup Text	Okay, now we can control the material flow with methods. Attaching them to the sensors of machines is the easiest way to do so. The possibilities here are nearly endless. Be sure to know what you want to achieve with the code and check if it behaves as it should.	

Level	2-2	**Title:** Random Numbers - Introduction

Task	Create a production line with a Source, two Drains „Drain_ok“ and „Drain_broken“, a SingleProc „M1“ and a Method „Inspection“, which is called by the exit control of M1.
	
	Assume that M1 has a percentage of good parts to broken parts of 80:20.
	
	Provide the following code in „Inspection“:
	/* declare a variable i of the type integer in the „is“-section */
	i := z_uniform(1,0,100);
	
	if i < 80 then
	@.move(Drain_ok);
	else
	@.move(Drain_broken);
	end;
	
	Run the simulation for 1 day. Check the throughput in the drains.	

KPI	Throughput per day in Drains	
Parameters	M1.ProcTime	1 min
Purpose	Use random numbers	
Popup Text	„z_uniform(seed value, start, stop)“ creates uniformly a random number with the start and stop value being the interval borders. Note that the stop value is not part of the generated numbers. The seed value is not so important for now. Let’s give this one another twist.	

Level	2-3	**Title:** Random Numbers – Revisited
Task	Create the same model as in the level before and add a third Drain „Drain_repair“. Assume that M1 has a percentage of good parts to broken parts to repairable parts of 80:10:10. How do you have to adjust the source code to achieve a proper assignment? Run the simulation for 1 day. Check the throughput in the drains.	
KPI	Throughput per day in Drains	
Parameters	M1.ProcTime	1 min
Purpose	Programming without guidance	
Popup Text	If you have to accommodate more cases than three, it may be useful to let an „inspect“ case selection do that job. Let’s have a look at this.	

Level	2-4	Title: Random Numbers – Inspect Selection
Task	Create a Source, a Buffer and four Drains „M1“ to „M4“ in parallel, which function as blackboxes for machines outside of our area of interest. Add a Method called „Distribution“ and attach it to the exit of the Buffer. The Drains will simulate machines in different departments of our plant, which we must supply. The various supply needs are based on individual skill levels of the workers. Since the workers on M4 are the fastest, they are likely to need supplies four times more often than the workers on M1. M2 and M3 will get two and three times more supplies than M1. We will accomplish this with the method „Distribution“. Use the random numbers, generated by „z_uniform(1,1,11)“. The inspect syntax is as follows: inspect i when 1 then <statements> when 2,3 then <statements> … else <statements> end; You can separate different numbers in the <number> block with a comma. How can we assure the correct assignment of the MUs? Run the simulation for 1 day. Check the throughput in the drains.	
KPI	Throughput per day in the drains	
Parameters	Buffer.ProcTime	1 min
	Buffer.Capacity	10
Purpose	Use the inspect selection	
Popup Text	You can not only inspect numbers but also strings, booleans etc.. We will get to that later.	

Level	2-5	Title: Waituntil
Task	Create three Sources „S1“ to „S3“, three SingleProcs „M1“ to „M3“ in parallel, a Buffer and a Drain. Add a Method called „Ctrl“ and attach them to the exit of each of the SingleProcs. Only the Sources are connected to their machine. The Buffer is connected to the Drain. In the method, write: waituntil Buffer.empty prio 1; @.move(Buffer); Run the simulation for 1 day. Check the throughput in the machines and the statistics of the Buffer.	
KPI	Throughput per day in the machines	
Parameters	M1.ProcTime	1 min
	M2.ProcTime	1 min
	M3.ProcTime	1 min
	Buffer.ProcTime	1 min
	Buffer.Capacity	5
Purpose	Learn about waituntil	
Popup Text	The „prio“ prioritizes the suspended statements. The higher the number, the more priority is granted (allowing one method to wake up ahead of another).	

Level	2-6	Title: Stopuntil
Task	The layout is the same as in the level before. In „Ctrl“, write: stopuntil Buffer.empty prio 1; @.move(Buffer); Run the simulation for 1 day. Check the throughput in the machines and the statistics of the buffer.	
KPI	Throughput per day in the machines	
Parameters	M1.ProcTime	1 min
	M2.ProcTime	1 min
	M3.ProcTime	1 min
	Buffer.ProcTime	1 min
	Buffer.Capacity	5
Purpose	Difference between waituntil and stopuntil	
Popup Text	Have you noticed the difference in the buffer workload? Before resuming with the statements behind waituntil, the condition is checked a second time, while using stopuntil, the following commands are executed no matter if the condition is still fulfilled. As always, you have to decide which behavior fits your simulation the best.	

Level	2-7	Title: Failures
Task	Create a Source, a SingleProc "M1" and a Drain. Assume that the machine has an availability of 90 percent and a mean time to repair (MTTR) of 3 minutes. Since PSIMA does not (yet) support the failure settings in the property window, we must assign the failures by the init method. This is equally effective and gives us the opportunity to take a look on failures here. In Plant Simulation, you can use the user dialogue boxes to assign failures to machines. In the "init" method, write: is do M1.Availability := 90; M1.MTTR := 180; end; Run the simulation for 1 day. Check the throughput in M1.	
KPI	Throughput per day in M1	
Parameters	M1.ProcTime	1 min
	M1 Availability	90 %
	M1 MTTR	3 min
Purpose	Using Failures	
Popup Text	Since Failures are always a part of a plant’s everyday life, we will revisit them again later.	

<table>
<tr><td>Level</td><td>2-8</td><td>Title: Transporter</td></tr>
<tr><td>Task</td><td colspan="2">This level is a little heavy on programming. This is because PSIMA does not allow to create and delete sensors via property dialogue.
This is the one and only time you have to set the sensors by yourself.

Create a Source, a Track, a SingleProc "M1" and a Drain. Connect M1 with the Drain.
Add a Method "init" with the lines:

/* declare a variable i of type integer */
.MUs.Transporter.create(Track, 5);
Track.cont.Backwards := true;
Track.cont.Speed := 0.5;
for i := 1 to Track.numSensors loop
 Track.deleteSensor(i);
next;
Track.createSensor(0,"Relative", "Track_Ctrl", false, true);
Track.createSensor(1,"Relative", "Track_Ctrl", true, false);

Add a Method "Track_Ctrl" and write:
(SensorID : integer) /* <- passed argument to method */
is
do
 if SensorID == 1 then
  Source.cont.move(@);
  @.Backwards := false;
 else
  waituntil M1.empty prio 1;
  @.cont.move(M1);
  @.Backwards := true;
 end;
end;

Run the simulation for 1 day. Check the throughput in the drain.</td></tr>
<tr><td>KPI</td><td colspan="2">Throughput per day in Drain</td></tr>
<tr><td rowspan="2">Parameters</td><td>M1.ProcTime</td><td>1 min</td></tr>
<tr><td>Track.Length</td><td>10 m</td></tr>
<tr><td>Purpose</td><td colspan="2">Use sensors and the transporter, pass arguments to the method</td></tr>
<tr><td>Popup Text</td><td colspan="2">This one was a bit tricky. Well done! We will have some more fun with transporters in the next level pack.</td></tr>
</table>

Level	2-9	Title: Delivery Table	
Task	Create a Source, three SingleProcs „M1“ to „M3“ in parallel and a Drain. Add a Table and two methods. In the first method, use an appropriate conditional structure to sort the big parts to M1, the medium parts to M2 and the small ones to M3. In the second method, move the parts from M1, M2 and M3 to the drain. In the Source, choose „Sequence Cyclical“ as „MU selection“ and assign the table. Open the Table and enter the values as shown in the figure. Run the simulation for 1 day. Check the throughput in the machines.		
KPI	Throughput per day in the machines		
Parameters	M1.ProcTime		1 min
	M2.ProcTime		1 min
	M3.ProcTime		1 min
Purpose	Delivery Table, independent programming		
Popup Text	The Delivery Table gives you more opportunities to control the influx of MUs.		

Level	2-10	Title: Delivery Table – Part 2	
Task	The layout is the same as in the level before. In the delivery table, change the numbers to 1 for big parts, 2 for medium parts and 3 for small parts. Run the simulation for 1 day. Check the throughput in the machines. Figure out and explain the resulting values in the report.		
KPI	Throughput per day in the machines		
Parameters	M1.ProcTime		1 min
	M2.ProcTime		1 min
	M3.ProcTime		1 min
Purpose	Variation to Delivery Tables, understand report entries		
Popup Text	After having learned all the above, it is time again to exercise it in a Game Set and in the next level pack as well.		

Level	2-11	**Title:** The Airport
Task	When you open the model surface, you will see that the building and connecting of the objects is already completed for you. Your task is to submit the necessary code in the two methods to ensure full functionality. For an easier understanding of the model, read the following elaborations: The sources generate passengers and planes, as you can see by their names. In the table „Planes“, the Delivery Time row sets the time where the MU is generated. Every MU has a name according to its destination. In the table „Schedule“, every flight is assigned to one of the three gates. The store „Airport“ contains the passengers of all 10 flights, also named according to their destination. All passengers are generated as .MUs.Entity during simulation start. The method „Dispatch_Planes“ should assign the right gate to the flight via the Schedule table and call the „Dispatch_Passengers“ method with the parameters destination and name of the associated terminal. The method „Dispatch_Passengers“ searches the Airport for MUs headed to the passed destination and moves them to the assigned terminal. In the end, the passenger list (assembly list) is passed to the gate (assembly station) to allow proper boarding. If all passengers have boarded the plane, it gets directed to the takeoff. In this game set, you get a first glance on how the third level pack is going to look. You will be faced with a setup you have to fix, make work or optimize. One last hint: Don’t think too difficult. The levels are designed to be comfortably solvable on a mobile device, so you can orientate your coding by the hints given in the method comments. Have fun and don’t give up too easily!	
KPI	Throughput in drain	
Purpose	Complex level, coding with some guidance	
Popup Text	Wow, this was a tremendous effort. Very well done! Hopefully you had a clear imagination what to do at every point in time. You are perhaps also subject to some vacation now… If you like to succeed in more complex surroundings like this, you may take a look at levelpack 3. There are way more achievements to earn. You can also review the solutions of other players and get feedback to your submits to help you improve.	

A3 - Levelpack 3

Level	**3-1**	**Title:** Introduction – Goods Arrival
Task	You are a Junior Simulation Engineer who is hired by an automotive supply manufacturing company to establish a small simulation division. The management wants to improve the throughput through a lean production line. Upon arrival, you take a close look at all of the departments in the material flow chain. While inspecting your first station, the delivery of the goods, you see a lot of opportunities for improvement at a first glance. The delivery intake consists of three terminals. One of them has the capability of handling big parts, one can handle small parts and the third terminal is suitable for both categories. Since the third terminal is due to expensive makeovers, the management is looking for ways to invest the money on a better spot to ensure a rise in production. Aware of your troubleshooting skills, you set off to work. Hint 1: The parameters of both the machines and the MUs are not subject to change. Hint 2: Detect the bottleneck and implement a workaround. Hint 3: You have to edit some code in the methods. One side note: In level pack 3, your solution will be reviewed by a fellow simulation expert who has already solved the certain level. After you have successfully mastered a level, you are also eligible to review other solutions and gain achievements for your reviewing contributions. Back to business: Run the simulation for 1 day. Check the contents of the stores and compare with the previous setup.	
KPI	Parts intake per day in Stores	
Purpose	Finding the bottleneck	
Popup Text	Alright, the bosses are excited. You proved your craft already on the first days. The second forklift is easy to implement and the costly construction works would have interfered with the daily production business anyway.	

Level	**3-2**	**Title:** Multi Order Picking
Task	Your next project is the parts logistics and assembly of a gearbox. While inspecting the machines and material flows, you can't help to scratch your head. Who invented this? But why? How do you have to adjust the processes to cut the time needed for those 25 gearboxes down by nearly 90 percent? Hint 1: The forklift may transport more than one component… Hint 2: Yeah right, but you have to assign the right values to the vehicle, of course.	
KPI	Time needed to produce 25 assemblies	
Purpose	Adjust a method to accommodate a better purpose	
Popup Text	Of course, the storage has to be reorganized to create collectable batches with everything needed for a gearbox on it. This was an easy one. Those tasks are done before the first coffee break. Let's move on to the next case.	

Level	3-3	**Title:** Second Source
Task	Your supplier of gearbox components has some problems with the availability of his production line. This results in delivery shortage which subsequently threatens the productivity in your own gearbox assembly. Your company has acquired a new supplier, which is willing to deliver your needed parts. Unfortunately, his prices are slightly higher and the traffic is much tighter around his plant, resulting in a slower truck speed. Since you can't afford to let your own production stop, the management decides to give it a shot. Because of the lower prices and the better infrastructure connection, the first supplier is still preferred, when available. You have the assignment to investigate, how many percent of productivity in your own assembly line are to gain by taking deliveries of the second supplier in case the first supplier is unable to produce. Open the model and add the required infrastructure and code in the methods „init" and „SensorCtrl". Run the simulation for 1 day. Check the throughput in the drain.	
KPI	Throughput per 1 day in the drain	
Purpose	Accessing object properties via methods	
Popup Text	The promising possible rise of up to 30 percent of productivity seals the deal. The increased creation of value in your plant outnumbers the higher logistics effort. But, let's think like business people: we might let the supplier manage our goods entry…	

Level	3-4	**Title:** Vendor Managed Inventory
Task	In your multi-branch company, there is also a unit that needs metal semi-finished products, like pipes, plates and profiles. The management obligates your supplier to restock the company's storage every time it needs to, based on a given required inventory list. You as logistics expert are in charge of this case. After doing a little research, the easiest and fastest way to complete this task is to establish a Vendor Managed Inventory system. Have a look at the next setup. Provide the necessary code snippets in the method „VendorManagedInventory". All other adjustments have already been done. Run the simulation for one day. Check the throughput in the drain.	
KPI	Throughput per day in Drain	
Purpose	Programming with little guidance	
Popup Text	This kind of VMI is known as „Continuous Replenishment", that means that the vendor collects the necessary data on his own during his visits. Other concepts are characterized by collection and transmission of the data via IT by the customer or a common stock between vendor and customer.	

Level	**3-5** (idea)	**Title:** Tugger Train - Milkrun
Task	Your next case is the production line of lightweight rims. In the setup, you see six transporters carrying the parts between the stations. The occupancy rate of each transporter is inacceptable low, wasting money on human resources and maintenance. The management tells you to find a more efficient way to connect the elements of the production line. Your task is to remodel the track layout, so that only one transporter is needed. By a fortunate coincidence, you recently read about the „Tugger Train“ concept in a specialized logistics magazine. While remodeling, you can get rid of the methods „init“, „SensorCtrl“ and „ExitingCtrl“. Clear them also in the exit controls of the stations. Reliable alternative methods are to be found in the upper right-hand corner. Be sure to rename the „init“ and „endsim“ methods to allow proper functioning. You don’t have to worry about the sensors on the track, since those two methods will take care of them. Hint 1: Make sure to connect the end and the start of the track to allow circular trips. Hint 2: To get comparable results, try to achieve a track length of about 75 m. Run the simulation for 1 day. Check the throughput in the drain.	
KPI	Throughput per 1 day in the drain	
Purpose	Remodel a functioning layout to accomplish an improvement	
Popup Text	Your throughput remains on the same level, but it is not always about those numbers, right? There is more to take in consideration when driving a successful business.	

Level	**3-6**	**Title:** The Canteen
Task	The management gets a whole bunch of complaints about the miserable layout of the canteen. It is just not possible to get the lunch during the break. You feel slightly abused to be asked for this kind of job, but you also feel with your coworkers and at least know the situation by yourself. During the two hours of lunch break – every unit has its own designated lunch time within that period, roughly 650 people can be served. Way too less, if you ask the 300 other hungry people. The management is willing to take some money in the hands and reorder the canteen layout if the peace will be restored. Now, you have a shot at clinching that „Employee of the year“ title for good. What is your plan? Hint: You are free to model whatever works, as long as you keep the source influx and the food station properties at given values. Hint 2: You are still required to take the „foot path“ in account. Every distance Source – Meal, Meal – Dessert and Dessert – Drain is min. 6m long. Run the simulation for 2 hours. Check the throughput in the drain.	
KPI	Throughput per 2 hours in drain	
Purpose	Improve a given system only with a target performance	
Popup Text	You are not only a hot candidate for that trophy, but the thankful canteen crew also insisted to feature a „Simulator’s Recommendation Meal“ every Monday at your own suggestion, which you selfishly use to order everything your spouse puts on your diet at home.	

Level	**3-7** (idea)	**Title:** Mirrors
Task	To expand the supplier product range, the bosses want to sell rearview mirrors as well. They contacted producers of assembly machines and received the necessary performance indicators. Your task is to find the combination with the highest throughput. In machine 1 (M1), the mirror glasses are attached to the holding case. In machine 2 (M2), the assembly gets its mounting attachment. In machine 3 (M3), the product is packed and labeled. Since every machine has the same ProcTime, the failure values are of interest. M1_Av70_MTTR10 has an availability of 70 percent and a Mean Time To Repair of 10 minutes. The same scheme applies for the other machines. As you can imagine, it would take eight simulations to get all the necessary results. Hint: With some thinking, you may cut it down to four experiments or even less if you are willing to do some modelling. Coding is not required. For reviewing, submit a model with only the best-performing machine combination connected. Run the simulation for 1 day. Check the throughput in every combination.	
KPI	Throughput per 1 day in production line	
Purpose	Conduct a small simulation experiment	
Popup Text	With more options to inspect, the amount of necessary simulation runs increases rapidly. In Plant Simulation, you can take a look at the ExperimentManager. Since the right machines are selected, the production can begin. The first fabricated mirror now tells you who entered your office without you needing to turn around.	

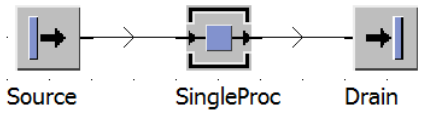
Level	**3-8** (idea)	**Title:** Seat Assembly
Task	The seat assembly ties up dead money in stock. Due to the recently added mirrors to the product portfolio, there is less space for the seats. Instead of producing them as usual, the new approach is to assemble them only when requested. This allows to customize padding and colors of the seats according to the client. Thus, a better flow-through in the stock may be granted. The term for this new approach is „Build to order“ (BTO). This method has many benefits and disadvantages too, of course. The management emphasizes the low stock costs, qualifying BTO for this case. Implement the required code snippets in the „init“ and „BuildToOrder“ methods. Run the simulation for 1 day. Check the throughput in the drain.	
KPI	Throughput per 1 day in drain	
Purpose	Programming without guidance in a rather simple setting, assembly table per method	
Popup Text	Great! Thanks to your tireless efforts, the company’s revenue will increase drastically due to new products to sell. In your last big case study, you have to roll up the shipping department. Let’s add your fingerprint there as well. But take a deep breath first, for things are going to escalate quickly.	

Level	3-9	**Title:** Crossdocking – serial
Task	Since you got to know the entire company, your last station is the shipping of the goods. The produced rims, gearboxes, seats and mirrors are to be commissioned for transport. Every car manufacturing customer sent his just-in-time order list for this day. Is it possible to complete these tasks in the given 12 hour shift? Remember, the parameters are like always not subject to change. Take a careful look at the setup and proceed to implement the behavior of the crossdocking car in the same-named method, approaching one order after another. If you feel advanced, you can also edit the prepared method without the guiding hints. Run the simulation. Figure out how much time is needed to complete the order list.	
KPI	Duration to complete 100 tasks	
Purpose	Implement a rather complex behavior with some or little guidance	
Popup Text	Okay, this procedure is too slow. Nightshifts are too expensive to run them on a regular basis. We have to implement a more efficient distribution pattern.	

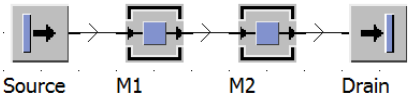
Level	3-10	**Title:** Crossdocking – parallel
Task	Your next approach is to work on four consecutive tasks at the same time. You get the needed rims and bring them to all four terminals, for example. Is this the right way to save some unnecessary trips? Run the simulation. Figure out how much time is needed to complete the order list.	
KPI	Duration to complete 100 tasks	
Purpose	More complex systems, little or no guidance	
Popup Text	With some time left on the clock, our workers can now have their well-deserved lunch, too. As for now, every major issue in the plant is solved. Very well done, congratulations! To avoid unemployment, you should try and find some less fatal glitches and continue to improve the production and logistics. A state called „perfect as is“ does not exist.	

Figures for Levels - Levelpack 1

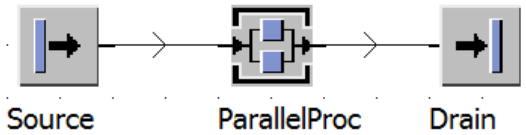
Level 1-1



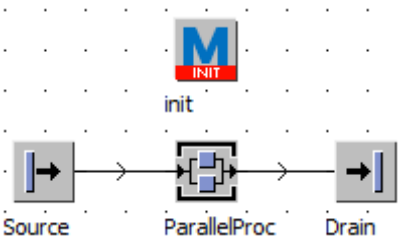
Level 1-2



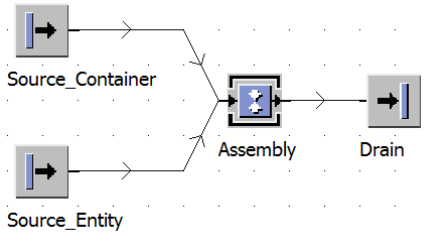
Level 1-3



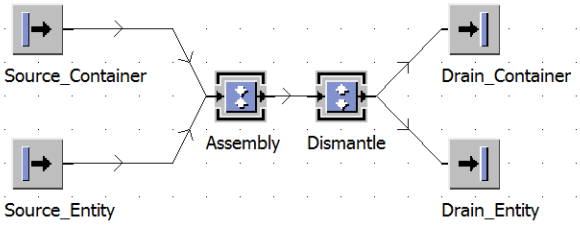
Level 1-4



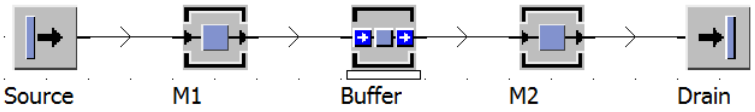
Level 1-5



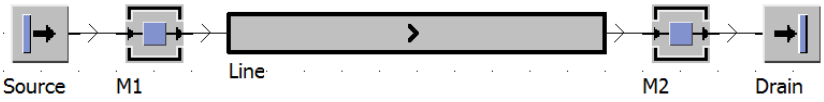
Level 1-6



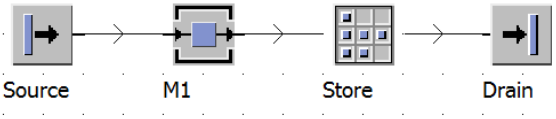
Level 1-7



Level 1-8

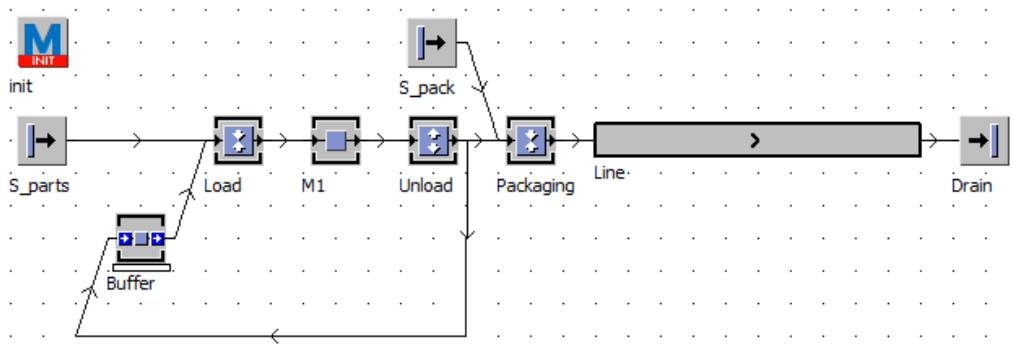


Level 1-9



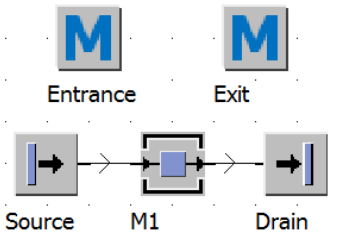
Level 1-10

Level 1-11

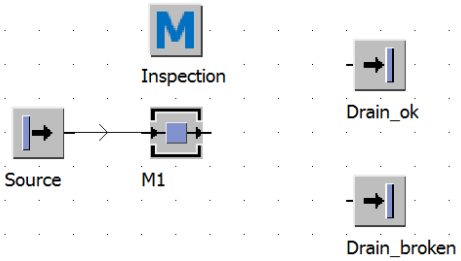

init
S_pack
S_parts
Load
M1
Unload
Packaging
Line
Drain
Buffer

Figures for Levels - Levelpack 2

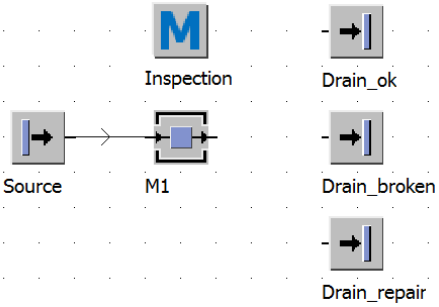
Level 2-1



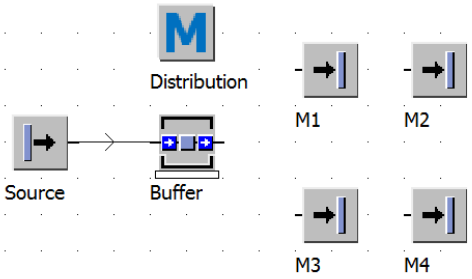
Level 2-2



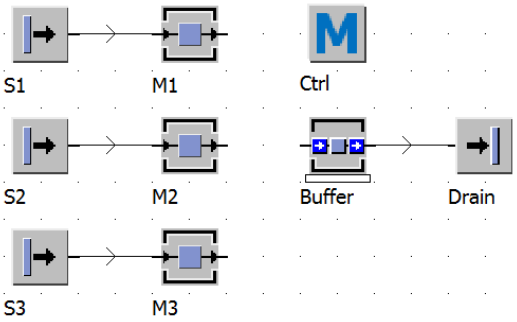
Level 2-3



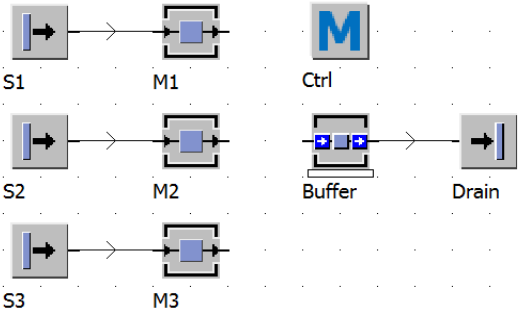
Level 2-4



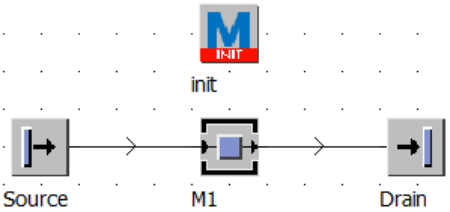
Level 2-5



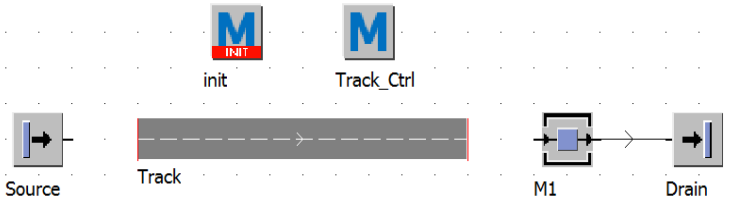
Level 2-6



Level 2-7



Level 2-8



Level 2-9

	object 1	integer 2	string 3	table 4
string	MU	Number	Name	Attribute
1	.MUs.Entity	1	big	
2	.MUs.Entity	1	medium	
3	.MUs.Entity	1	small	

Level 2-10