Orders and production plans in Pharma3

Updated: 2022-09-27

This document describes how different parts of the Pharma3 application send orders to each other, and how planning is done.

This document is based on Pharma3 ver 2.007, which implements (with some slight simplifications and "removal of apparent contradictions") the business rules described in Ben's Pharma-Model_Parameters.docx from 2022-08-23, and supplemented by email communication with Ben. The graphs are from a baseline run, `2.007/baseline-2.007-no-safety` on the SOD numbers, with no safety stocks and no disruptions.

For the flow chart, please see p. 2 of the document PharmaSim-SC-Map_With-Disruptions.pptx (2022-07-29, with safety stocks).

The post-production pools

The post-production pools include the Distribution Center (DC), the Wholesaler Pool (WP), the Hospital/Pharmacy Pool (HP), and the End Consumer Pool (ECP). There is also the Untrusted Supplier Pool (UP), which is an alternative (not completely benign) source of supplies for the HP.

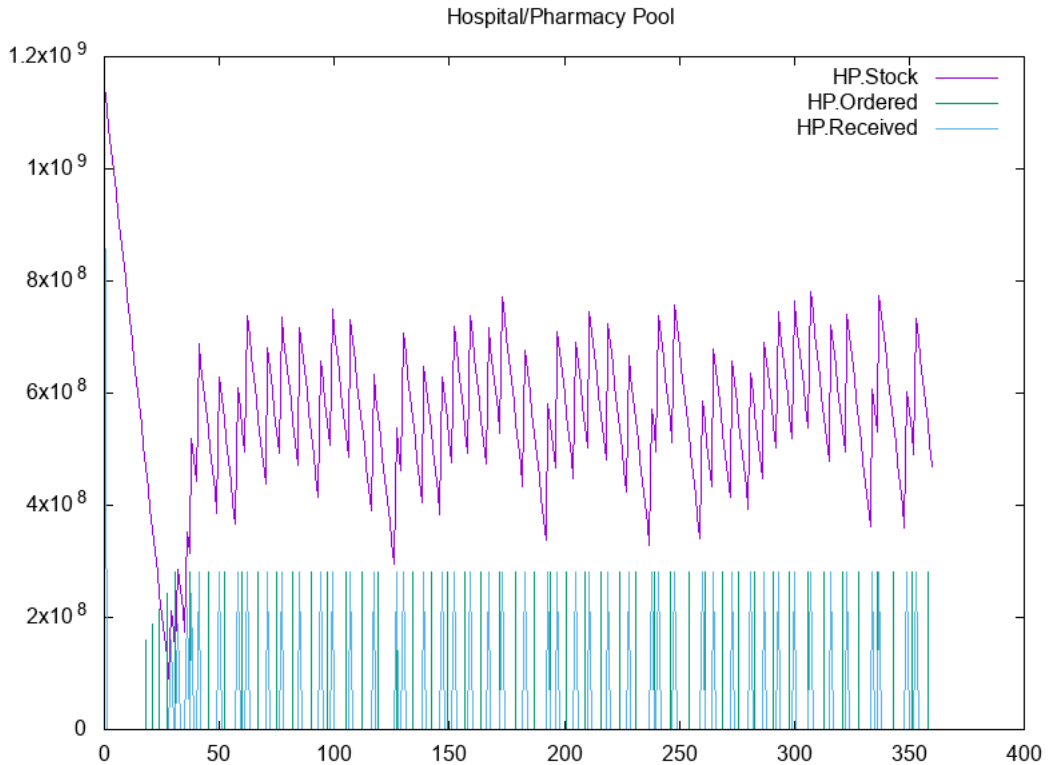
The End Consumer Pool is implemented as a DED `Sink` (because it consumes stuff without storing it); the UP as a `Source` (because it's simply an inexhaustible source). For the other pools listed above we have class `pharma3.Pool`, derived from `Queue`; it "stores" the product in the form of `Batch` object (each of which represents an identifiable product lot), and supports pull requests.

For all pools, the Pharma3 app saves the important time series (daily orders sent by the pool, daily arrivals of the material to the pool, daily orders received by the pool, daily sending of stuff by the pool) into CSV files; I later plot those to PNG images with Gnuplot. Naturally, all those amounts (daily or all-time cumulative) are also available to an optimizer app through the Pharma3 API.

The **End Consumer Pool (ECP)** sends daily orders to the HP. The order size varies very little from one day to the next (within 0.01%). The fulfillment is instant (no transportation delay). (At this point I cannot find the place in the specs where this is specified, but since I implemented it this way, I assume it's either in the specs somewhere, or I got an explanation by email).

The **Hospital/Pharma Pool (HP)** sends its orders to the WP and/or DC whenever it decides its stock level is low. The replenishment criterion is based on the previous month's order level (the HP makes an order when its stock level drops below 0.75 of the monthly demand, and orders the amount equal to 0.25 of the monthly demand); the ordered amount may be shipped all at once right away (if it's available) or in several installments as it becomes available (if the WP and DC do not have the needed amount immediately available in stock); it takes 7 to 15 days (from the shipment date) for the shipment to arrive to the HP. This means that during the steady state (after the first 2 months or so of simulation) the stock level graph at the HP (see HospitalPool.png) has a characteristic sawtooth shape, with linear decline (daily draw by the ECP) and quick increases (arrival of the replenishment shipments). However, during the first month the HP does not yet know what the "monthly demand" is (because it has not accumulated a month's worth of data yet); that results in the anamoulous behaviour during the first month: the stock level dropping to a lower level than one would have in the steady-state saw-tooth, and orders sent to the WP or DC being smaller than really needed.

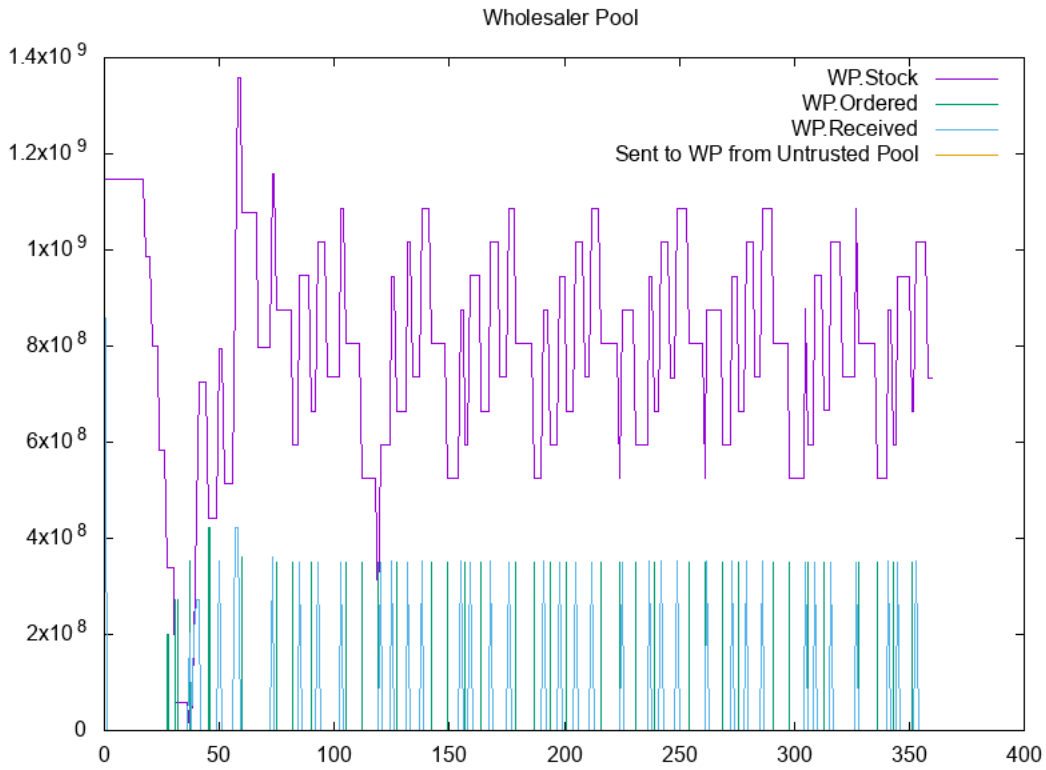
This, of course, is a rather strange behaviour, but email exchange with Ben (2022-08-30, 30, Subj: Reorder policy for the Hospital/Pharmacy Pool) has affirmed this policy.



The orders sent by the HP have an anmoalous pattern during the first month, because at that time the HP does not yet know what the "normal monthly order amount is". The orders become regular after that. Subsequently, the HP stock level pattern remains steady, because the WP has enough material in stock.

The **Wholesaler Pool (WP)** sends orders to the DC, and has them fulfilled, in a similar way to how the HP pulls stuff from the WP. In the steady step, the stock level graph thus has a sawtooth shape as well, but with rectangular "teeth", rather than "triangular" ones. This is because the WP stock level drops happens in large drops (roughly once a week, as the orders from the HP come and are fulfilled), rather than in small daily installments.

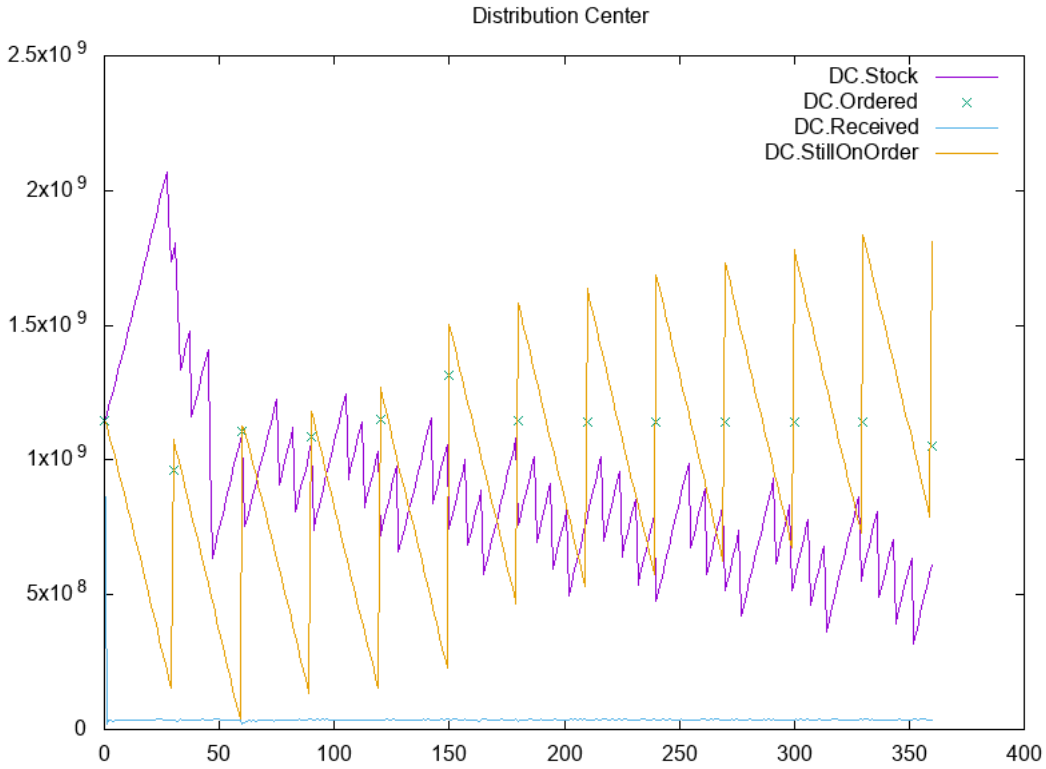
When WP cannot have its order fulfilled from the DC right away, it (unlike the HP) does not place a "back order" (which will be fulfilled when available), but instead gets the missing supply from the USP, which is a "magic source" (inexhaustible and un-disruptable). This can be viewed as a "*poisonous countermeasure*", because some percentage of stuff coming from the UP is bad, and this is not discovered, thus allowing bad stuff to propagated to the ECP eventually.



The orders sent by the WP have an anmoalous pattern during the first month, because at that time the WP does not yet know what the "normal monthly order amount is". The orders become regular after that. Subsequently, the WP stock level pattern remains steady, because the DC has enough material in stock.

The **Distribution Center (DC)** sends orders to the Focal Company (FC). Unlike the other pools, it sends its orders on a pre-set schedule (monthly), rather than being driven by the stock level. Unlike the downstream pools, the order amount is based on the averaged monthly demand from downstream (WP and/or HP) over the previous 4 months. Moreover, unlike the HP and WP, which early on do not know what the monthly demand is, the DC knows it, sort of. Its initial stock level is equal to what the monthly demand of the EC happens to be, and is allowed to use this value as the surrogate for the monthly demand. Still, since the WP's or HP's orders coming to the DC early on are anomalous, the DC's orders during the first 2-3 months are slightly anomalous too.

Once the DC has sent its monthly order, it will see the product come in, in small increments, over the next month (and, actually, a bit longer than that). So the stock level graph for the DC has triangular sawtooth with slow increase (continuous arrival of product) and sharp drops (fulfilling large orders received from downstream pools).



The DC stock is being gradually drawn down, and the "still on order amount" never quite gets to zero (ater the second month's smaller order, t=60), because the production capacity is a bit lower than the demand.

The DC receives the packaged product (PP) contniously. A downward spike (around t=60) is due to the factory managing to fulfill the month's order before the next order came in, and shutting down for a short while. This phenomenon would be more common if we had the demand a bit lower than the production capacity.

Orders and plans within the production network

Once the FC receives an order from the DC, it needs to know how to fulfill it: how much raw material (RM) to order from the external suppliers, and what production plans to issue to the production nodes. This is how it's done at present.

Ordering materials

Since we know the input:output ratio for all industrial processes (if QA issues are disregarded, 1 unit of the RM makes 1 unit of the API, etc), we can tell how much RM, excipient, and packaging material (PM) is needed to make the desired amount of the end product (PP, packaged product). More precisely, we carry out the following

calculation (using the `pharma3.GraphAnalysis` class).

The operations model prescribed by the parameter file is very rigid: it is know in advance what the "split ratio" at each step are. For example, 90% of all incoming RM is supposed to be sent to the API Production step of the FC, 5% to the CMO Track A, and the remaining 5% to the CMO Track. Similarly, the API and the bulk drug produced by appropriate FC production units are split in certain fixed percentages between the subseqent FC stages and the appropriate CMO tracks. The average QA parameters ("tested good" / "tested bad" / "rework" percentages) for each production stage are known as well. This is summarized in the following report, where, for each production node, α="tested bad" proportion, β=reword proportion, and γ=(1-α-β)/(1-β) is the "good output":"input" ratio. This can be summarized by the following report:

Table 1: Production Graph Parameters

```
Below, 1 ba = 500000
Root. Send: to CmoTrackA 5%, to CmoTrackB 5%, to ApiProduction 90%
[0] CmoTrackA: α=0.01, β=0.01, γ=0.99. Max gross thruput=3.829 ba. MaxIn=3.791 ba, MaxOut=3.752 ba. Send: to DrugProduction 100%
[1] CmoTrackB: α=0.025, β=0.00, γ=0.975. Max gross thruput=3.829 ba. MaxIn=3.829 ba, MaxOut=3.733 ba. Send: to DC
[2] CmoTrackC: α=0.015, β=0.00, γ=0.985. Max gross thruput=21.821 ba. MaxIn=21.821 ba, MaxOut=21.494 ba. Send: to DC
[3] CmoTrackD: α=0.03, β=0.00, γ=0.97. Max gross thruput=25.203 ba. MaxIn=25.203 ba, MaxOut=24.447 ba. Send: to DC
[4] ApiProduction: α=0.01, β=0.01, γ=0.99. Max gross thruput=68.915 ba. MaxIn=68.226 ba, MaxOut=67.537 ba. Send: to CmoTrackC 30%, to DrugProd
[5] DrugProduction: α=0.01, β=0.00, γ=0.99. Max gross thruput=50.915 ba. MaxIn=50.915 ba, MaxOut=50.406 ba. Send: to CmoTrackD 50%, to Packagi
[6] Packaging: α=0.02, β=0.00, γ=0.98. Max gross thruput=25.077 ba. MaxIn=25.077 ba, MaxOut=24.575 ba. Send: to DC
```

This allows our software to carry out a complete forecasting calculation: given a certain amount of RM at the input, how much of it will get (in the appropriately processed form) to each stage. In the assumption of no throughput constraints (i.e. the capacity of each production node is sufficient to process all materials given to it).

Table 2: Production Graph Report (ignoring thruput constraints)

```
Root: in=1.00, γ=1.00. Send: to CmoTrackA 0.05 (5%); to CmoTrackB 0.05 (5%); to ApiProduction 0.90 (90%)
[0] CmoTrackA: in=0.05, γ=0.99. Bad=0.001. Send: to DrugProduction 0.049 (100%)
[1] CmoTrackB: in=0.05, γ=0.975. Bad=0.001. Send: to Distributor: 0.049
[2] CmoTrackC: in=0.267, γ=0.985. Bad=0.004. Send: to Distributor: 0.263
[3] CmoTrackD: in=0.333, γ=0.97. Bad=0.01. Send: to Distributor: 0.323
[4] ApiProduction: in=0.90, γ=0.99. Bad=0.009. Send: to CmoTrackC 0.267 (30%); to DrugProduction 0.624 (70%)
[5] DrugProduction: in=0.673, γ=0.99. Bad=0.007. Send: to CmoTrackD 0.333 (50%); to Packaging 0.333 (50%)
[6] Packaging: in=0.333, γ=0.98. Bad=0.007. Send: to Distributor: 0.327
Distributor receives 0.962
```

(Ben's parameter file has similar calculation in Sec. 15, but I did not follow them carefully. I trust that those above are more less equivalent to his).

According to the above, for 1 unit of (good) RM entering the FC+CMO production system, 0.962 units of the PP will eventually make it to the DC. Using this number, and taking into account the fact that not all RM arriving from the external supplier is good (and the "bad" percentage is known), the system then calculates how much RM should be ordered from the external supplier to produce (at the FC + CMO) the desired amount of PP. Similar, but much simpler, calculations are also made to decide how much excipient and packaging material to order.

These calculations have some problems, however:

- The ratios above are long-term averages (since the accepting or discarding any given batch is random, with a specified probability). Thus in reality we may slightly over- or under-order the input materials. The difference, though, is pretty small, considering how many batches need to be produced per month.
- The calculations above ignore the throughput constraints, i.e. they are made in the assumption that all production nodes have suffcient capacity to process everything given to them. In reality that is not the case: the monthly demand a few percentage point higher than the capacity; additionally, the fixed splits allocate the material among various tracks not exactly proporional to their production capacities. As a result, at present, in a baseline run one will have a slowly but continuously increasing backlog of input materials at some of the production nodes, while the outstanding order amount (how much the DC has ordered, but has not received yet) will gradually increase. This is not all that noticeable on a typical 1-year run, but will be quite pronounced on, say, a 30-year run.

At present, the above issues have not been addressed.

Controlling production in the CMO

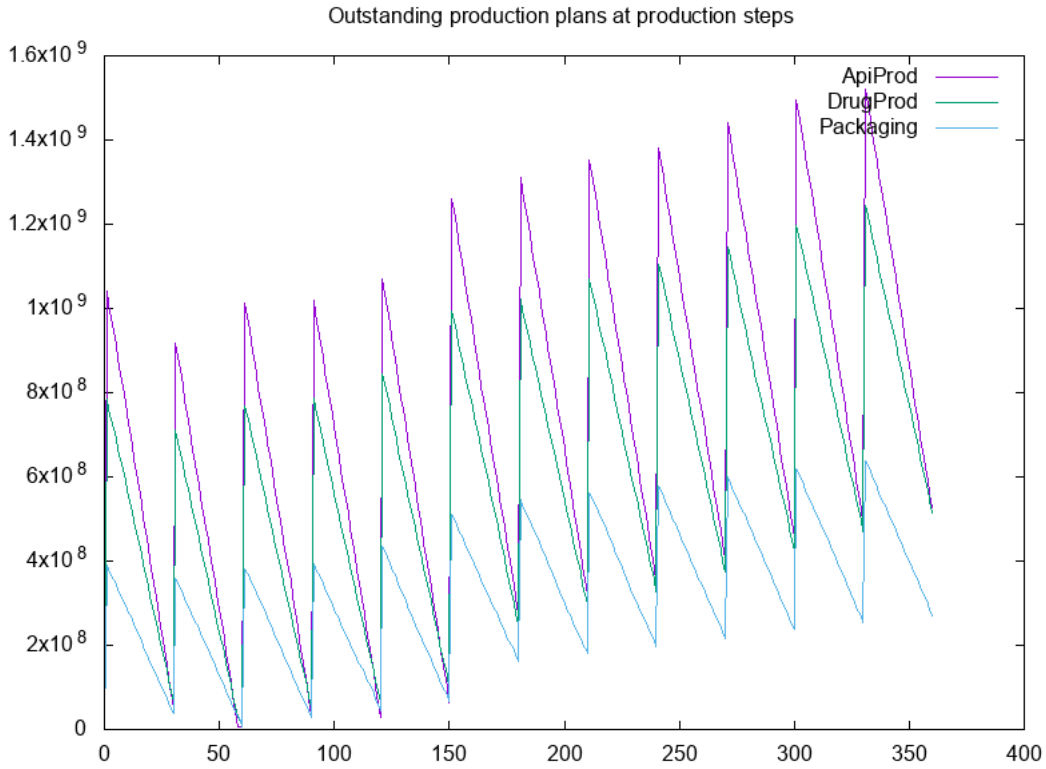
The production in the CMO is entirely supply-side driven. Each of the CMO tracks is supposed to keep working as long as it has any material in its input buffer. Therefore, at present, no production plan amounts are explicitly computed for CMO tracks. If we needed such plans (or, rather, "forecasts") in order to enable the CMO tracks to carry out disruption detection, they can be computed in a similar way as for the FC stages (see below).

Controlling production in the FC

The FC stages (API production, Drug production, Packaging) cannot be controlled just by the availaibility of inputs, because if they don't have their input materials arriving through the "normal channels", they are expected to use safety stocks instead.

To keep the FC from working (and using safety stocks) when not needed, the system computes for each FC production node its production plan, based on the values in Table 2. For example, if the table states that the ratio of the amount of API used by the FC drug production node to the total amount of PP received by the DC is 0.673 : 0.962, then in order to satisfy the total order for *X* units, the FC makes a plan for the Drug production unit to process (0.673/0.962)**X* units of API.

Each FP production unit maintains its current outstanding production plan amount, which in incremented (as per the above) when the FC receives an order and gives a production plan to each production unit, and decremented whenever the production unit starts a new batch.



In a situation when the demands exceeds the production capacity, the plans will be never entirely fulfilled, and the outstanding production plan amount never drops to zero. In a lower-demand world, one will see the outstanding plan amounts reach zero once the most recent order has been fulfilled, and stay so until the next order comes in.

Adjusting model parameters to have less backlog

As mentioned above, the current model file seems to have these properties:

- The overall demand is a bit higher than what the production network can supply, even without disruptions.
- Some of the capacity values are slightly disbalanced, sending more materials to certain production stages than they can process.

In our predictor unit (GraphAnalysis), we can take into account the max throughput of each production node. The analyser then forms a report, in which it predicts, for each production node, the average daily numbers for:

- how much input material will arrive to that node every day;
- how much of that material, if any, will add to the backlog (i.e. the amount of input material that this node will never get a chance to process, because the input stream exceeds the processing capacity);
- how much of bad output will be produced;
- how much of the good product will be produced;
- what the (average) utilization of the node will be (i.e. how much of its production capacity it will use).

A node that, on average, is given less input material than it can process (i.e. operates under capacity), will have utilization under 100% and zero backlog; a node that receives more material than it can process, will have 100% utilization and a positive backlog number (i.e. daily contribution to the bakclog).

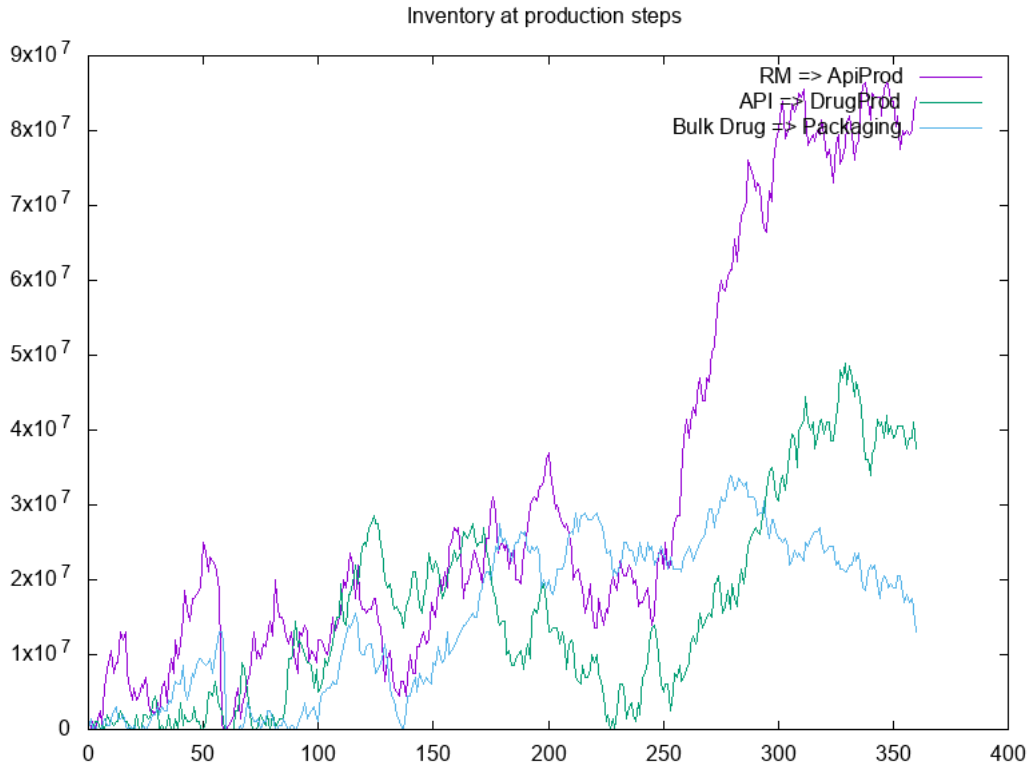
As the following table shows, the largest avg daily input that should allow the system to run without creating an input material backlogs in any node is ca. 75.262 batches (of good RM). It will result in the daily output of ca. 72.383 "notional batches", i.e. 3.6191e7 units.

Table 3: Production Graph Report for daily input=75.262 batches

```
Below, 1 ba = 500000
Root: in=75.262 ba, γ=1.00. Send: to CmoTrackA 3.763 ba (5%); to CmoTrackB 3.763 ba (5%); to ApiProduction 67.736 ba (90%)
[0] CmoTrackA: in=3.763 ba, γ=0.99. Util=99.272%. Bad=0.038 ba. Send: to DrugProduction 3.725 ba (100%)
[1] CmoTrackB: in=3.763 ba, γ=0.975. Util=98.279%. Bad=0.094 ba. Send: to Distributor: 3.669 ba
[2] CmoTrackC: in=20.115 ba, γ=0.985. Util=92.184%. Bad=0.302 ba. Send: to Distributor: 19.814 ba
[3] CmoTrackD: in=25.077 ba, γ=0.97. Util=99.501%. Bad=0.752 ba. Send: to Distributor: 24.325 ba
[4] ApiProduction: in=67.736 ba, γ=0.99. Util=99.282%. Bad=0.684 ba. Send: to CmoTrackC 20.115 ba (30%); to DrugProduction 46.936 ba (70%)
[5] DrugProduction: in=50.661 ba, γ=0.99. Util=99.502%. Bad=0.507 ba. Send: to CmoTrackD 25.077 ba (50%); to Packaging 25.077 ba (50%)
[6] Packaging: in=25.077 ba, γ=0.98. Backlog=0.00 ba. Util=100.00%. Bad=0.502 ba. Send: to Distributor: 24.575 ba
Distributor receives 72.383 ba
```

If one increases the input of the RM to this model, one can bring utilization of CMO Tracks A and B to 100% as well, but some of he FC nodes (first Packaging, and then other stages as well) will start accumulating backlog. The CMO Track C, however, will remain under-utilized, because it's fed from the FC API Production node, which is already running at near-capacity.

In our parameter file, the required daily demand is about 4% higher than what can be produced in the no-backlog prediction above (3.7690e7 vs. 3.6191e7). This means that running Pharma3 with the production targer as per the parameter file (and the proportionally computed input) will show growing backlogs of input materials in many FC production nodes, and the production plans chronically underfulfilled. This is indeed what the experiments show.



A growing inventory backlog, if persistent over a long time, indicates a production node's inability to deal with the demand. We see this in most nodes here. As we have a "forklift model", products come into the inventories continuously. If we were to switch to a "container ship model" for certain links, then the inventories (at the nodes that receive "container ships", e.g. at the RM QA node) would have occasional jumps, and then slow decline as they are used up.

Running Pharma3 with the target daily output 3.6191e7 (i.e. 72.383 notional batches) does show nearly-sustainable performance. The backlogs still grow, but very slowly; I assume this happens due to the randomness built into the production process. (Runs `enough.01`, `enough.02`) Further reducing the target by about 1% (i.e. setting demand to ca. 99% of the notional production capacity) allows running with no backlogs and with fully fulfilled production plans. (Run `low.02`). (The run data are available upon request).

More graphs

Just because.