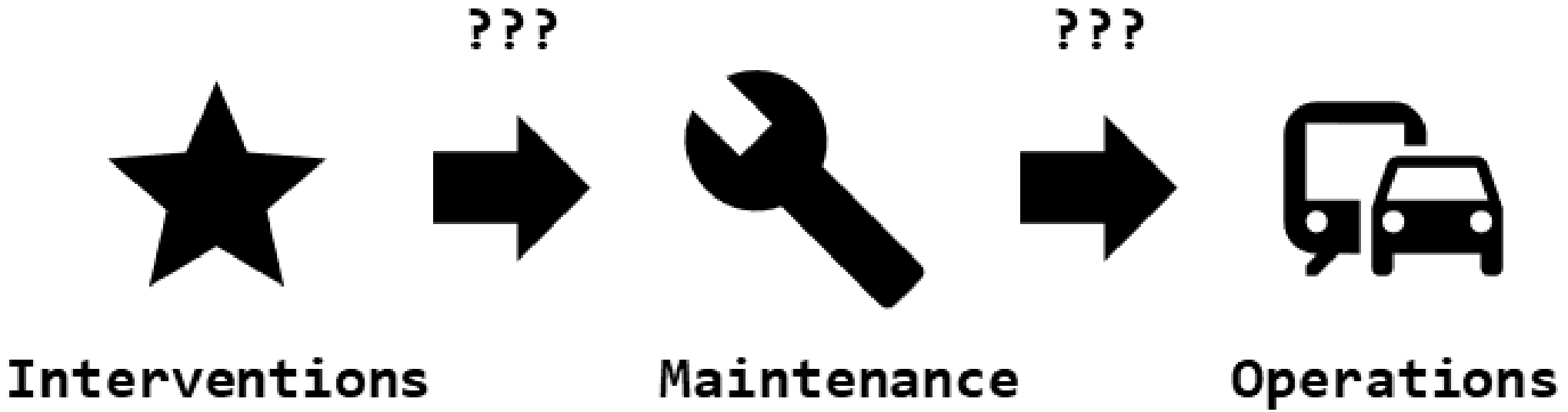
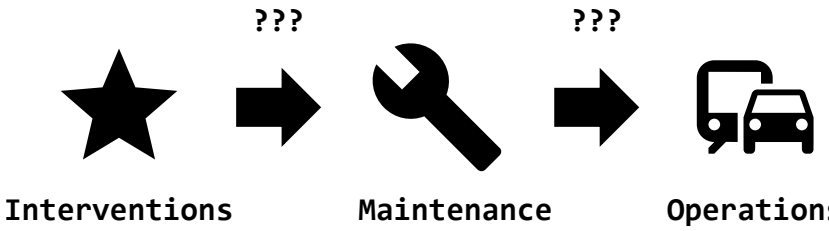


The Tensor Approach to Enterprise Asset Management with a Root Cause Analysis Model

Sasha Lubyansky, PhD

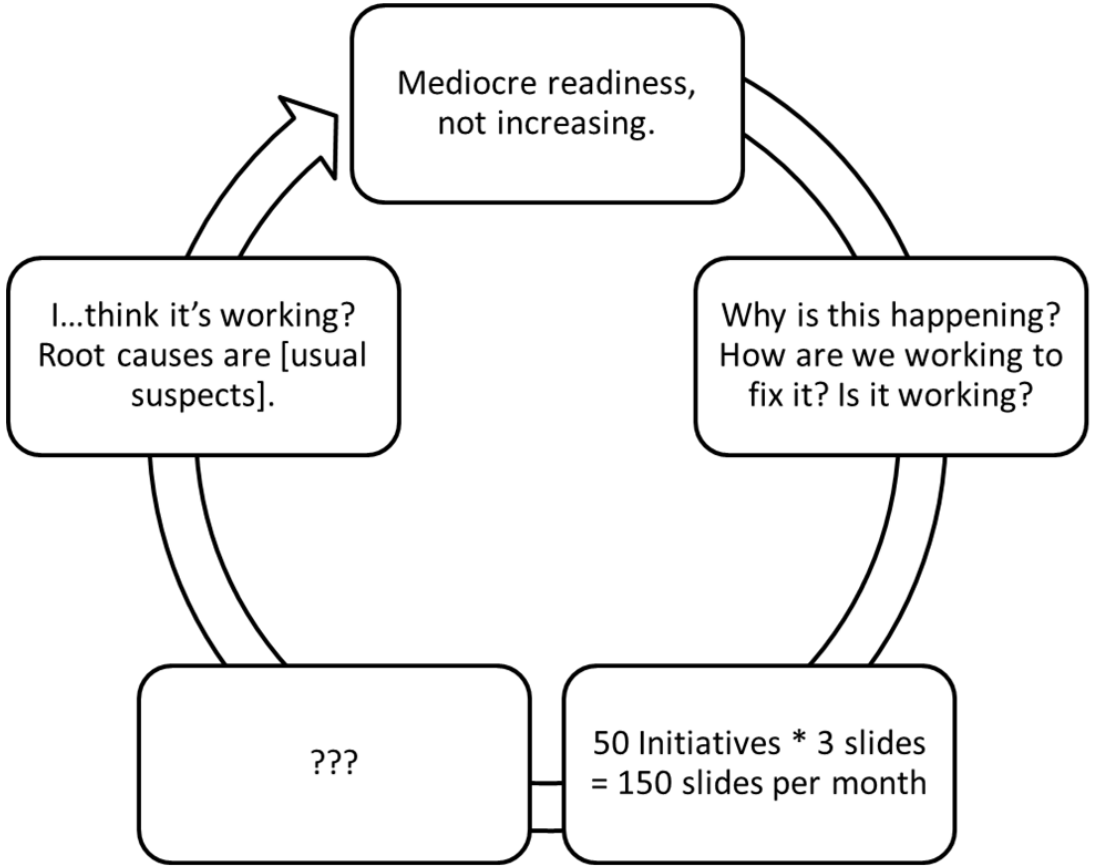
The Problem



- Let's imagine there's an organization trying to manage a **fleet of vehicles**. They want to improve maintenance to maximize the **readiness** of their fleet to operate.
- There are hundreds of interventions (initiatives and ongoing work) to improve readiness, but stakeholders have trouble **quantifying their effects**. Their Q's:

1. Was the intervention **implemented**?
2. How did the intervention affect **maintenance performance**?
3. How did maintenance performance affect **operations performance and readiness**?

- A common "day in the life" problem from program managers in this organization goes as follows:



Methodological Problems and Solutions

- Data
  - Transactional data is in the **wrong format**.
  - Convert data into a big table (**tensor**).
- Methodology
  - **Machine learning** and **discrete event** models strike out. Trends, details not key.
  - Use **system dynamics** to get at aggregate control dynamics, but with enough detail.
- Implementation
  - The input data is **big**. The tensor makes the output data even **bigger**.
  - Run sophisticated **Spark** pipeline to allow performant local, server, cluster, cloud ops.

- "Standard" System Dynamics Won't Work
- Conceptually, system dynamics is the right approach, since it focuses on aggregate continuous control (AKA **feedback loops**).
- Trouble is: **SD software can't scale** (Vensim, Stella/iThink, PowerSim, AnyLogic).
- The methodology of the Root Cause Analysis Model (RCAM) uses the three concepts of:
  1. **Tensors**
  2. **System dynamics**
  3. **Spark**
- …To scale system dynamics in a novel way, allowing for **the solution of entirely new classes of problems**.

Data Scale

- Here is the approximate size of the biggest data tensor.
- Work Orders
  - **3000 vehicles** (30 types of vehicles * 100 average vehicles per type).
  - **24000 hours** (3 years).
  - **36 job statuses** (stocks and flows).
  - **13 work centers** (to deal with different vehicle parts).
  - **6 action taken codes** (inspection, repair, etc).
  - **6 units of measurement** (counts, parts, labor, etc).
  - **4 difficulties** (bins of labor hours per work order).
  - **2 effects** (prevents operations or not).
  - **2 categories** (generic or scenario-specific).
- At 1/20 density, a sparse tensor of an average vehicle type (100 vehicles) is **32 billion records** (**647 billion dense records**).
- A typical model output per vehicle type is a ***compressed* 10GB – 100GB!** (parquet/gzip).

- What benefit does this data granularity give?

- An **omniscient view of the entire maintenance process** for hundreds of vehicles, dozens of vehicle maintenance orgs, and hundreds of work centers.
- It's the **perfect environment to study intervention effects** in the context of production control, bottlenecks, Theory of Constraints, etc.
- The view is limited only by the problem **scope**, the fidelity of the transactional **data**, and the **hardware** used to model the data.

The Root Cause of the Problem

- Despite the sheer scale of the problem, the root cause is the typical system dynamics problem: the inability to causally link cause (intervention) to effect (readiness).
- There are limits to the degree to which we can simplify the problem so that the model is useful for analysis.
- The system itself is not amenable to the standard “five stocks or less” SD model. Any such model would produce banal results.
- There are 12 reasons why the structure of the system requires a relatively complex and granular model.

12 Reasons for Complexity: 1 to 6

- There is **tight coupling between the operational and maintenance sectors**. Operations generate the need for maintenance and maintenance enables operations.
- Maintenance is conducted as a **job shop instead of a flow shop**. That means there is no set of standard routings or BOMs to represent or optimize in a standard way.
- There are **dozens of interventions** being applied in the same time period for each type of vehicle. Interventions can be parts, labor, or process, and can affect different vehicle systems.
- Certain interventions are direct (add parts and labor) while others are **indirect** (publish a document to improve some process).
- The **age of individual vehicles** follows a **bathtub curve** for maintenance demand: lessening demand right at the beginning, then progressively more demand as the vehicle gets old.
- There is constant **cycling** of vehicles by status:
  - Into and out of field, depot maintenance.
  - Into and out of elevated operations by organization.
  - Into hot and cold status in-org.

12 Reasons for Complexity: 7 to 12

- Each **vehicle type** has very different performance and maintenance characteristics, sometimes even different work centers.
- While, officially, there is a dichotomy of **scheduled versus unscheduled** maintenance, this is partly hidden in the data.
- The transactional data has **fundamentally poor data quality** due to fat-fingering, legacy processing IT systems (ERP) and rules, and a very complex data pipeline between data entry and analytics-ready data views.
- There is a large amount of **inter-coder variance**. Each organization has a different "style" of data entry.
- There is **limited visibility into parts and labor dynamics**, so work orders form the backbone of the transactional data. Similarly, there is some data off limits for now, such as depot data, etc.
- The existing **high-level metrics** used to monitor vehicle performance in terms of operations and maintenance are **highly flawed**: noisy, overly aggregated, and with a questionable correlation to lower-level metrics.

How to "Do" System Dynamics the RCAM Way – Tensors

- The goal is to think in terms of dataframes. Then, think in terms of dimensions (primary and foreign keys) and measures (the data describing things about the record).
- For example, high-level work order data looks like this:

Work Order Name	Vehicle Serial Number	Vehicle Type	Vehicle Organization	Work Center	Action Taken	Labor Hours Spent	...
1234	243535325	First Type	AAA	12Q	A	10	...
2345	96978	Second Type	BBB	34B	B	100	...
563	5423052378	Second Type	AAA	19C	C	1	...

- Low-level work order data looks like this:

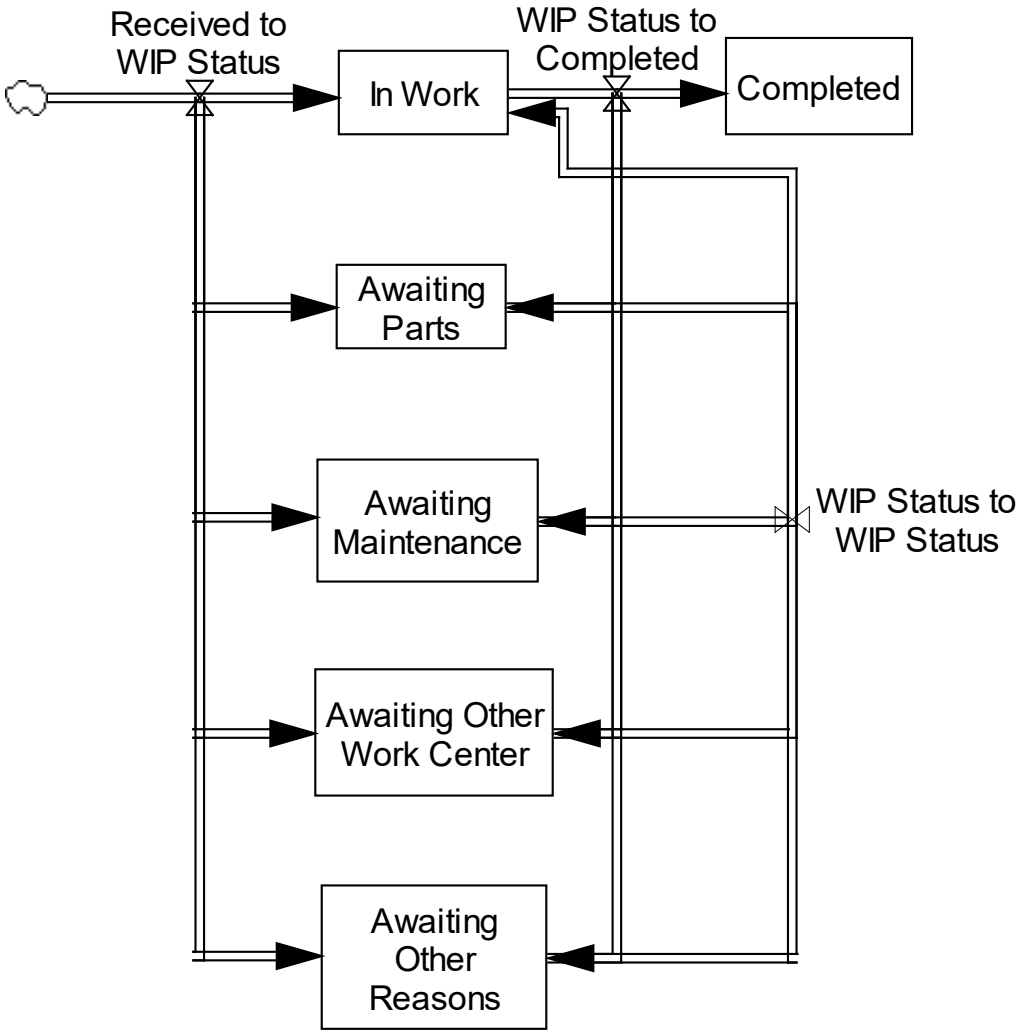
Work Order Name	Job Status Name	Job Status Start Time	Job Status End Time	...
1234	In Work	12:12	12:15	...
1234	Awaiting Parts	12:15	12:20	...
1234	Awaiting Maintenance	12:20	12:30	...
1234	In Work	12:30	12:32	...

- You must profile, cleanse, transform, and join all that data. This data wrangling is 90% of the work.

- Once joined and converted into a tensor, all the dataframe's indexes work together and all time is in interval (not event) form.
- Here are the indexes for work orders:

How to "Do" System Dynamics the RCAM Way – System Dynamics

- The stocks and flows are in the lower-level work order data. Simply:
  - A stock is a job status in an hour.
  - A flow is the transition between two job statuses per hour.
- The trouble with maintenance is that it is a job shop. Work of any type can go through any sort of path. Any number of job statuses and transitions until completion. So, all possible flows must be modeled.
- To do this, key groups of job statuses are modeled. Even with this simplification, there are 36 possible stocks and flows.



How to "Do" System Dynamics the RCAM Way – Spark

- In order to transform the data into this tensor format is way outside the abilities of system dynamics software.
- That doesn't mean a modeler has to start off in the deep end of the pool.
- RCAM went through several stages of development.

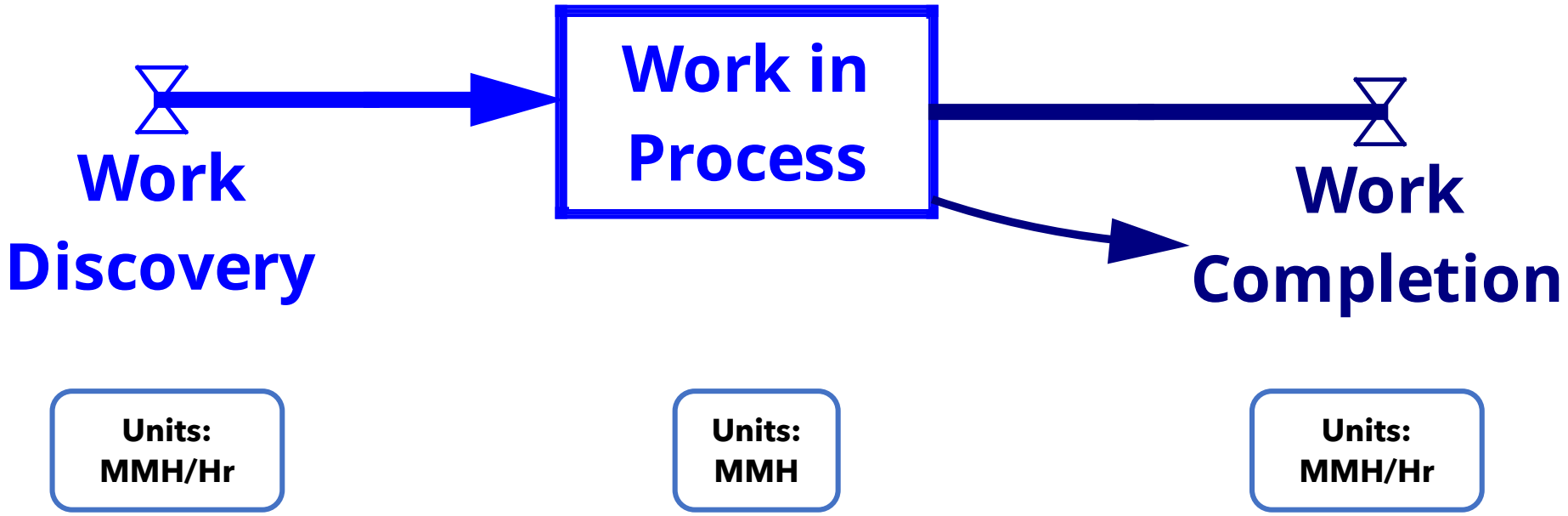
- RCAM's proof of concept was in Vensim. This was enough to show that, at the very highest level of aggregation system dynamics could capture historical behavior and predict system behavior.
- RCAM's prototype was in Pandas. This allowed the use of dataframes and design of the basic cleansing and transformation pipeline, but quickly hit Pandas' and Python's limits.
- RCAM's proper DevOps stage is in Spark, allowing for massively improved performance on existing hardware, and scaling to alternatives (CPU cluster, GPU cluster, cloud).

One Insight So Far: A High-Level Concept Model of Operations and Maintenance

- The use of RCAM to analyze historical operations and maintenance behavior has yielded- and continues to yield useful insights.
- Unfortunately, it's not possible to discuss RCAM scenarios or results in this paper.
- However, one general result of this work is that it's possible to tie operations and maintenance together quantitatively by:
  - Focusing on a common maintenance efficiency metric, maintenance man hours per operating hour (MMH/OH).
  - Filling in the causal structure for two sets of hidden variables that converts operations into maintenance and back.

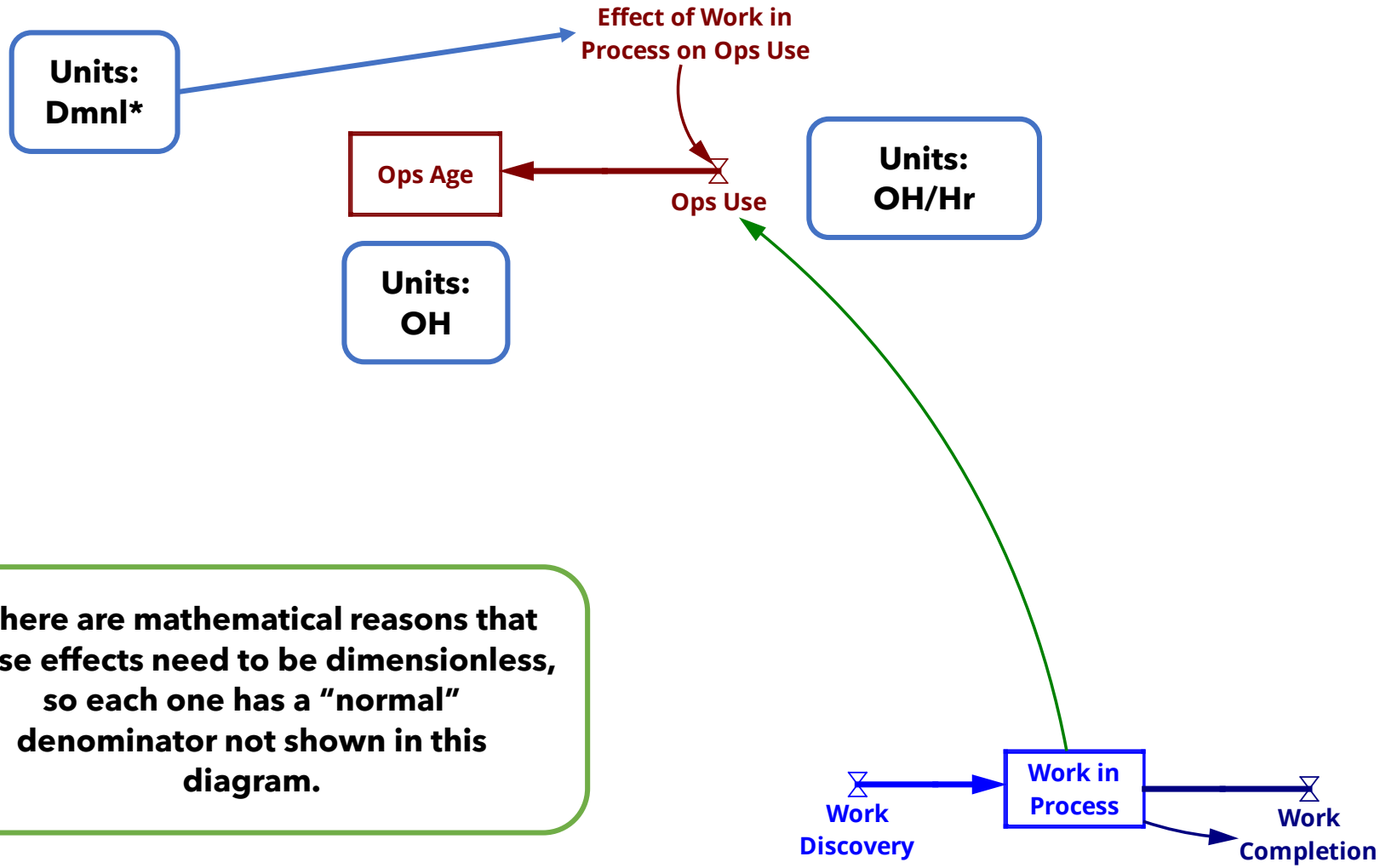
The stock and flow model of operations and maintenance

Let's start building the actual stock and flow structure starting with the obvious variables in the data. First, we know how much work is received over time, is in WIP at any given time, and is completed over time. Let's call this the "work" side AKA maintenance. We're measuring maintenance work using units MMH.



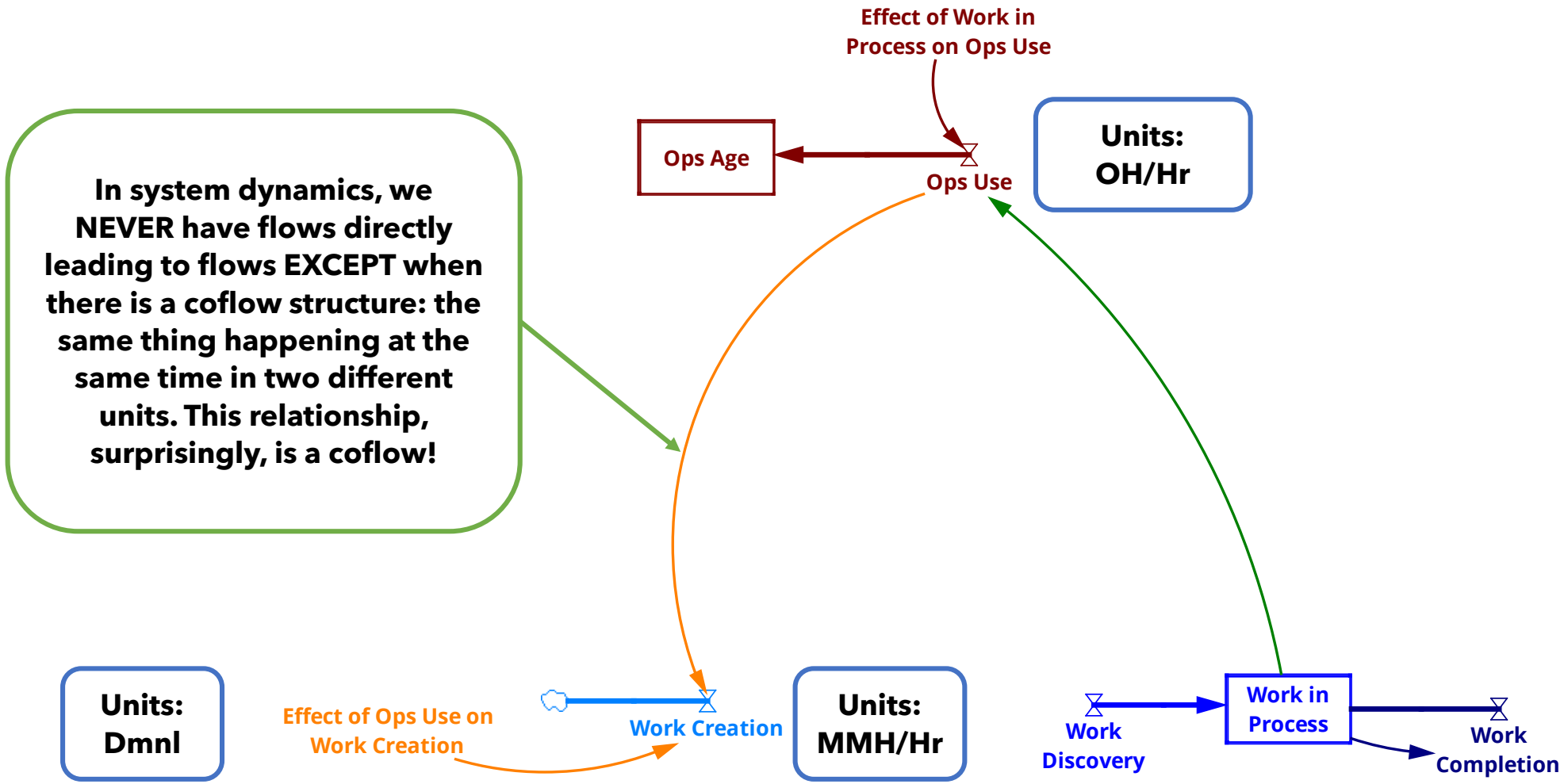
The stock and flow model of operations and maintenance

We also know how many operating hours are used over time and the total operating hour age since delivery. Let's call this side the "ops" side AKA operations. We're measuring operations using units "OH". WIP has affects the maximum operating hours directly. Open work orders of a certain type prevent ops.



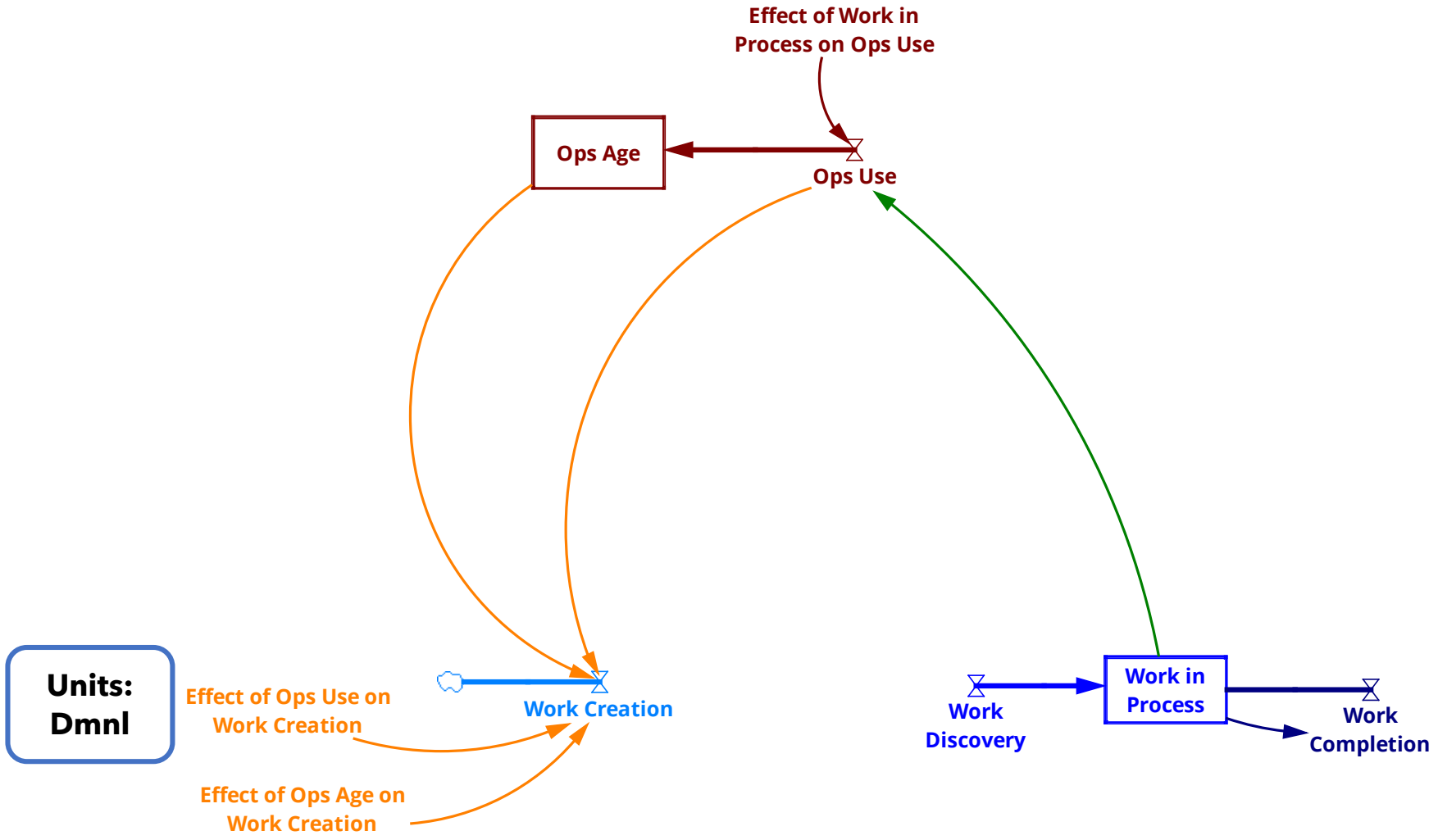
The stock and flow model of operations and maintenance

Here comes the first bit of alchemy: turning the use of operating hours into creation of man hours of maintenance work. Both variables must be flows, with a time unit denominator. Because of this, this effect happens in **process time**, **not chronological time**. Both work creation and the effect are hidden variables.



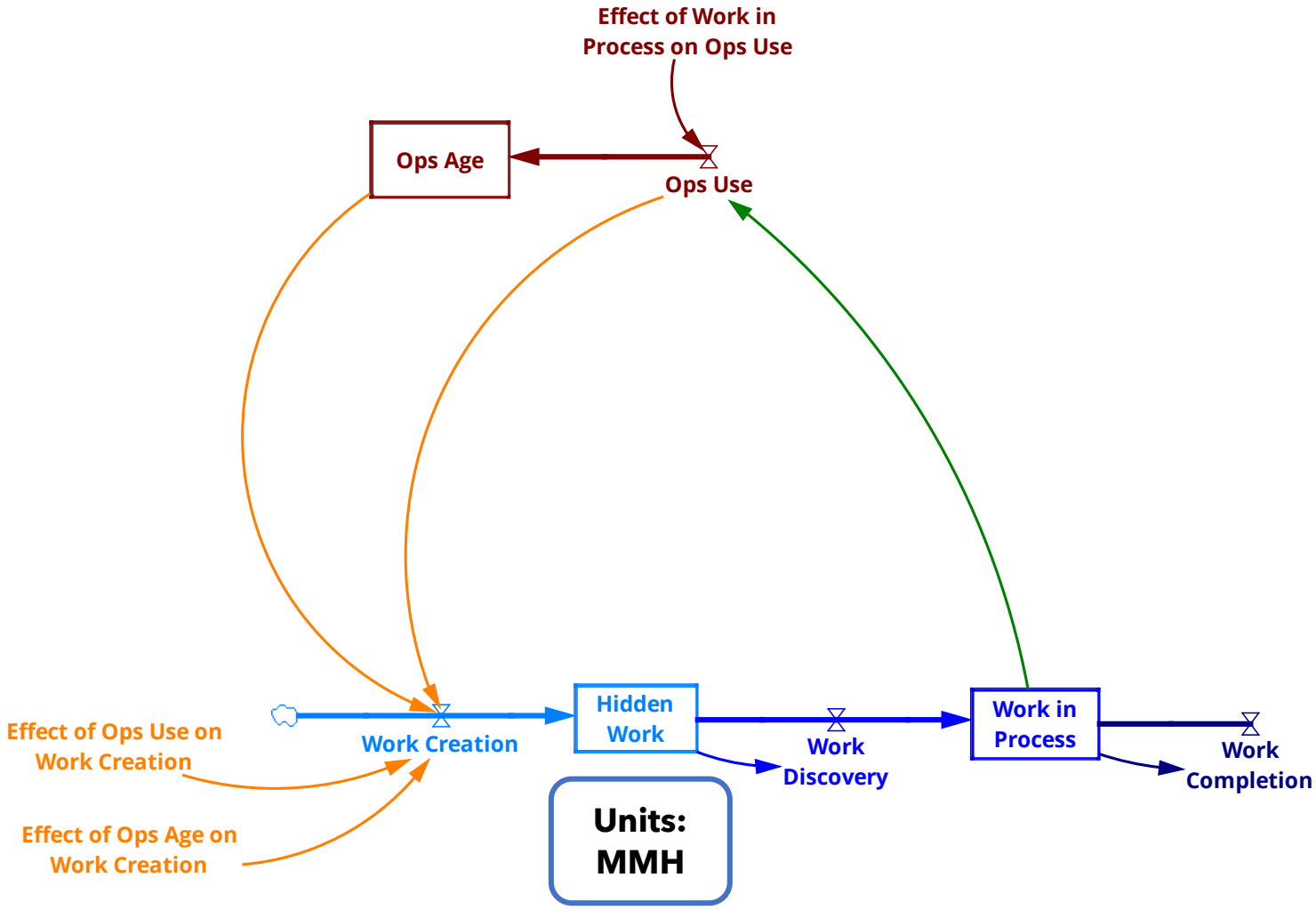
The stock and flow model of operations and maintenance

We also have a secondary effect for work creation, the effect of age. As is common knowledge (the Bathtub Curve), after an initial breaking in period, vehicles generate more maintenance work per operating hour. Again, this relationship is hidden.



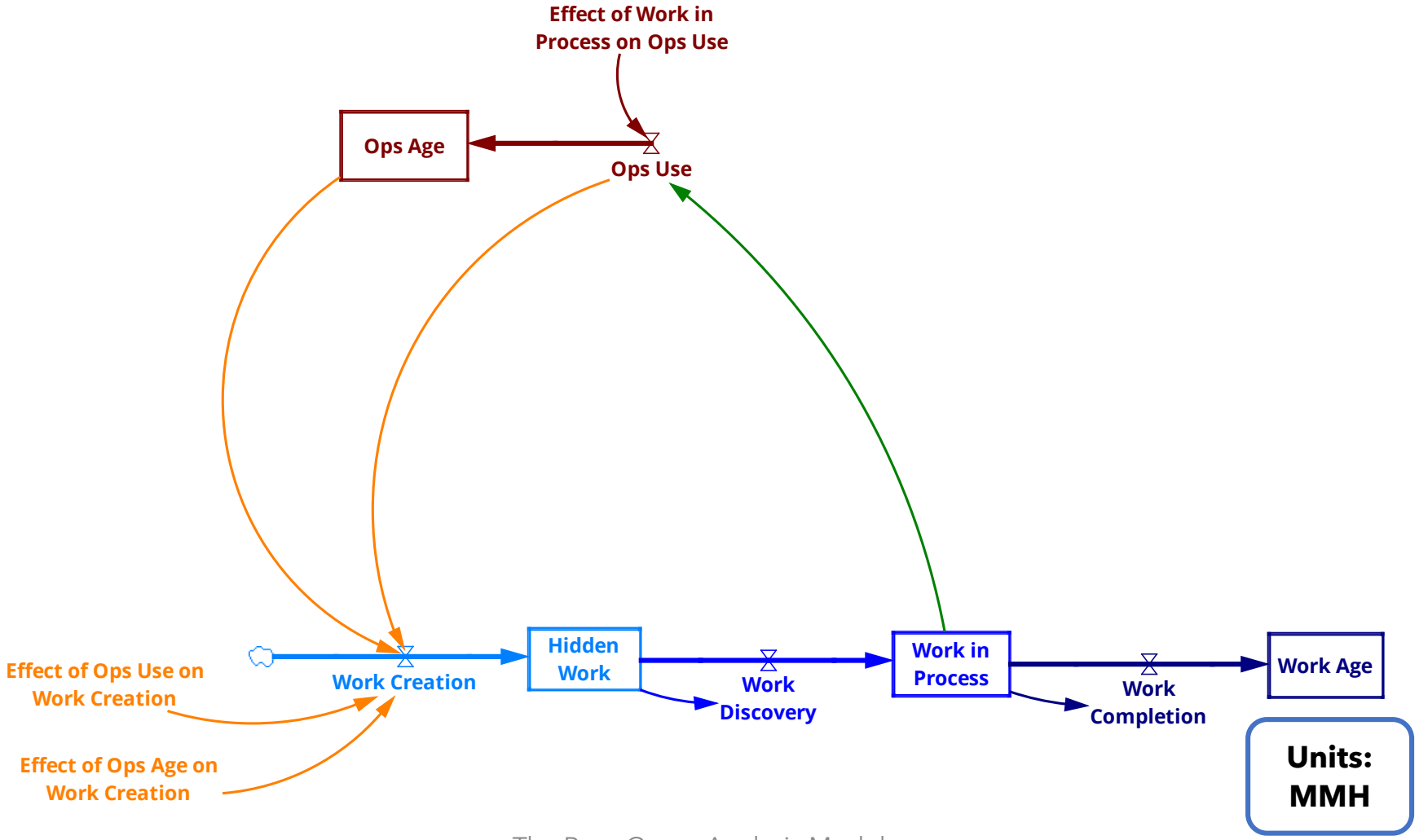
The stock and flow model of operations and maintenance

How do we link the hidden flow of work creation to the known flow of work discovery (received work orders)? With a stock, of course, in this case called hidden work. We know one special piece of information about this stock: it has a lower limit of zero!



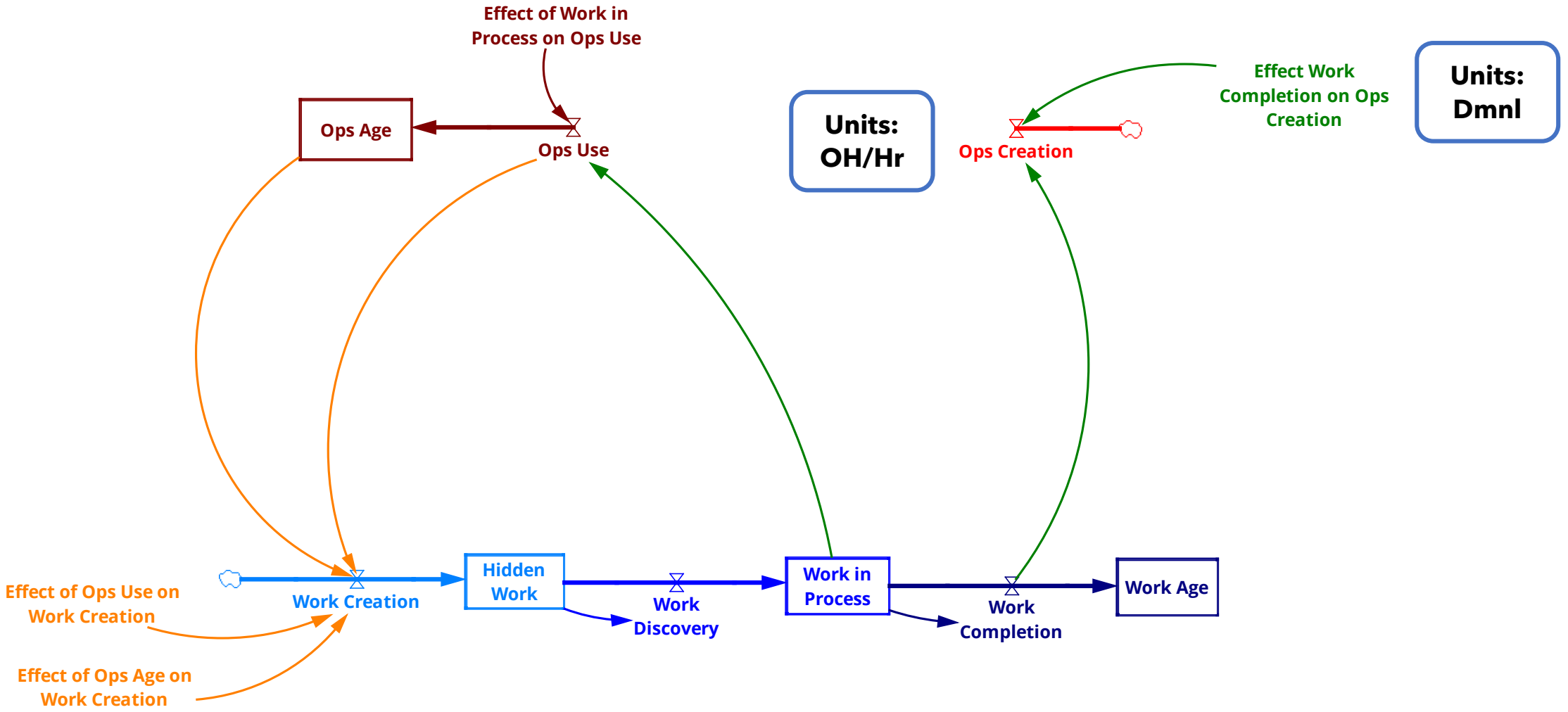
The stock and flow model of operations and maintenance

Finally, for maintenance, we can estimate a work age, in parallel to ops age. We can get a long term fit for the effects of ops use and ops age on work completion, then we can multiply ops age by that curve to estimate work age.



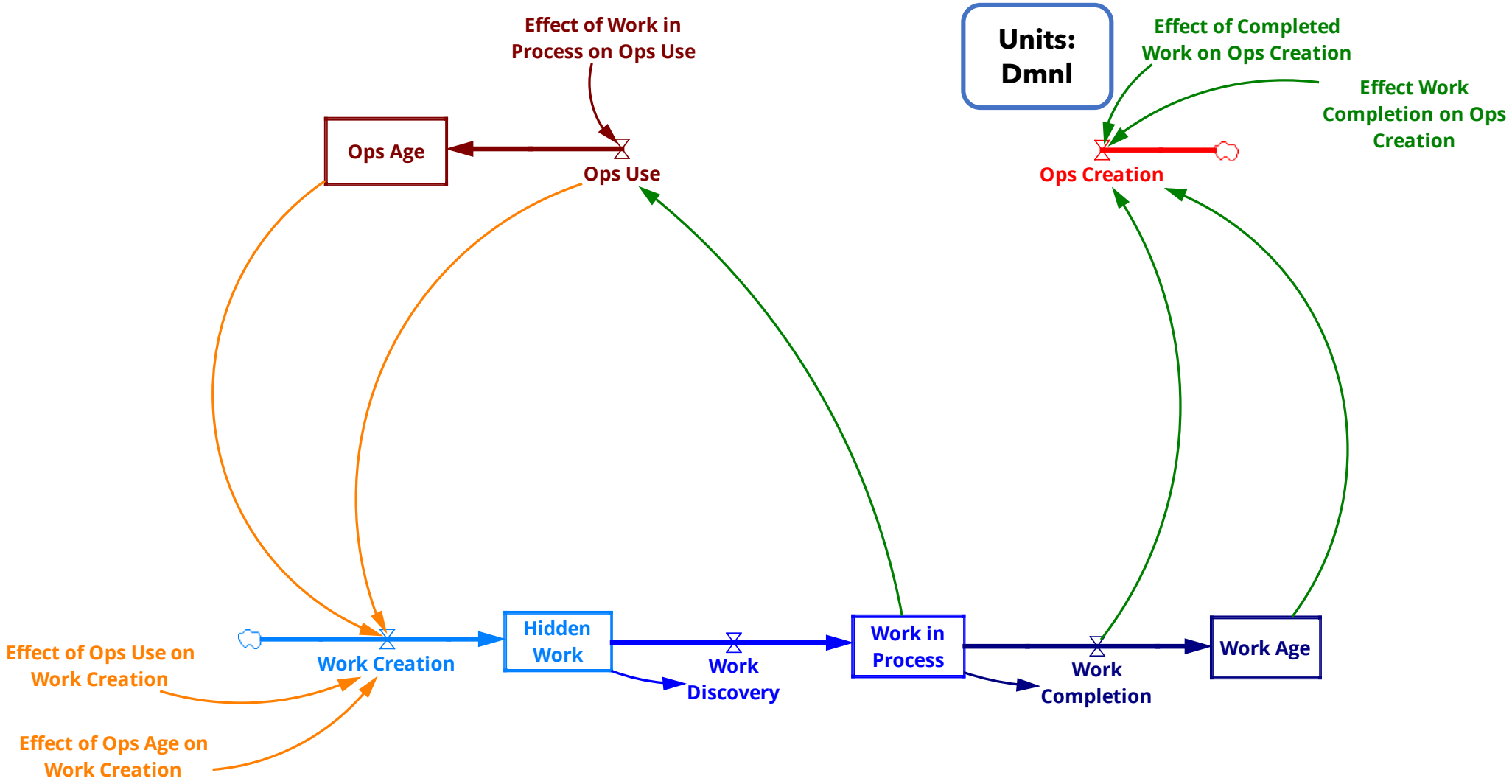
The stock and flow model of operations and maintenance

Now, for the second bit of alchemy: making the completion of work generate the ability to perform operations. **This is really the heart of readiness, not MC**. Again, this effect happens in **process time**, **not chronological time**.



The stock and flow model of operations and maintenance

We also anticipate that age, in this case work age, has some effect on creating ops, likely lessening it as vehicles get older.



The stock and flow model of operations and maintenance

The stock of potential ops completes the diagram. Like hidden work, potential ops is a hidden stock, but must be zero or more.