# Automated Continuous Program Estimation for Pipelines and Factories

November 2022

Bill Nichols
Chris Miller
Luiz Antunes
Julie Cohen

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

# Document Markings



This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.



DM22-1049

# Abstract

Automation in DevSecOps transforms the practice of building, deploying, and managing software intensive programs.

While this automation supports continuous deployment and rapid builds, information for program status remains laborious creating a lag between the arrival of program status metrics and the decision they are intended to inform. The pace of production and deployment runs weeks or even months ahead of programmatic information.

The emerging DevSecOps metrics such as deployment rates and lead times provide insight to how the software development is progressing but fall short to in terms of replacing program control metrics for assessing progress (e.g., burn rates against spend targets, integration target dates, and schedule for the minimum viable capability deployment. By instrumenting

the DevSecOps Pipeline and the pipeline's supporting environment continuous measurement of status, identification of emerging risks, and probabilistic projections is possible and practical. This presentation discusses research on the information modeling, measurement, metrics, and indicators necessary to establish a continuous Program control capability which can keep pace with

DevSecOps management needs. The importance of interactive visualization dashboards targeted to addressing program information needs is discussed.

We will also address gaps in the current state of the practice and barriers we have identified. Finally, we present examples we recommend needed future research based on our initial findings and discuss open research questions

# CMU SEI is a DoD R&D Federally Funded Research and Development Center


Established in 1984 at Carnegie Mellon University (CMU)

Charged to improve the state of the practice of software engineering, cybersecurity, and AI

Collaborates with CMU and broadly in academia, government, and industry

Conducts both fundamental research and classified work

~610 staff members

FY20 total funding $140M Offices in Pittsburgh and DC, with locations near customer facilities in MA, TX, and CA

# Outline

- The programmatic challenge with DevSecOps
- Scaling up to multiple pipelines
- Automating measurement and analysis
- Description of our prototype
- Lessons learned
- Conclusion

# Automation for Agile Program Management

How Agile is your program?

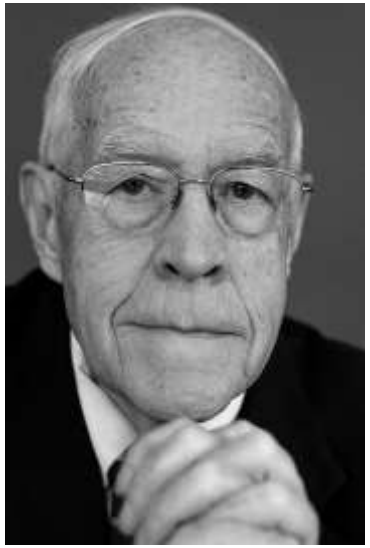
Programs slip a day at a time; how quickly can you recognize and respond?

How long does it take

- to recognize a problem?
- to respond to user needs?
- to a critical issue?

Is your data from **today**? Builds happen constantly, what is the data lag?

- **How credible is your data?** Accurate? Complete?
- Do you have the necessary data?



*How does a project get to be a **year** late?*
*One day at a time.*
*-Fred Brooks*

# Program Managers Need Answers

As the PM of a complex agile program, you need to have an answer to these types of questions

- If a new high priority capability (or vulnerability) emerges when will my next two capabilities be delivered?
- The next capability has slipped twice – when will it really be delivered?
- How many more teams would it take to deploy three specific capabilities in 6, 12, and 18 months?
- Why am I finding so many software defects during final test and how do I improve software quality?

# As a Program Manager, what do you need to know?
## And when do you need to know it?

Where are we? (**Capability % Complete, Schedule Consumed)**

Where did we expect to be? (**Planned Capability Complete** and **Schedule Variance Estimation accuracy)**

When will we be done? (**Schedule projections to complete)**

What will it cost? **(Often dominated by Schedule and staffing)**

When **could** we have it? (**projections)**

- If we adjust priorities?
- If we add staff or other resources?

# Automated Continuous Estimation for **Continuous Deployment** and **Pipelines of Pipelines**

**Automation** drives continuous integration and delivery of software, but outpaces program control

To solve this problem:
Automate data collection
Model DSO systems with **Monte Carlo**, and provide continuous reporting.

- Determine status
- Project future events
- Provide evidence for corrective actions

**Goal:** Programs using DSO(DevSecOps) have constant access to information needed to monitor and control schedule and cost commitments.

Status and projection models should be available in real time.

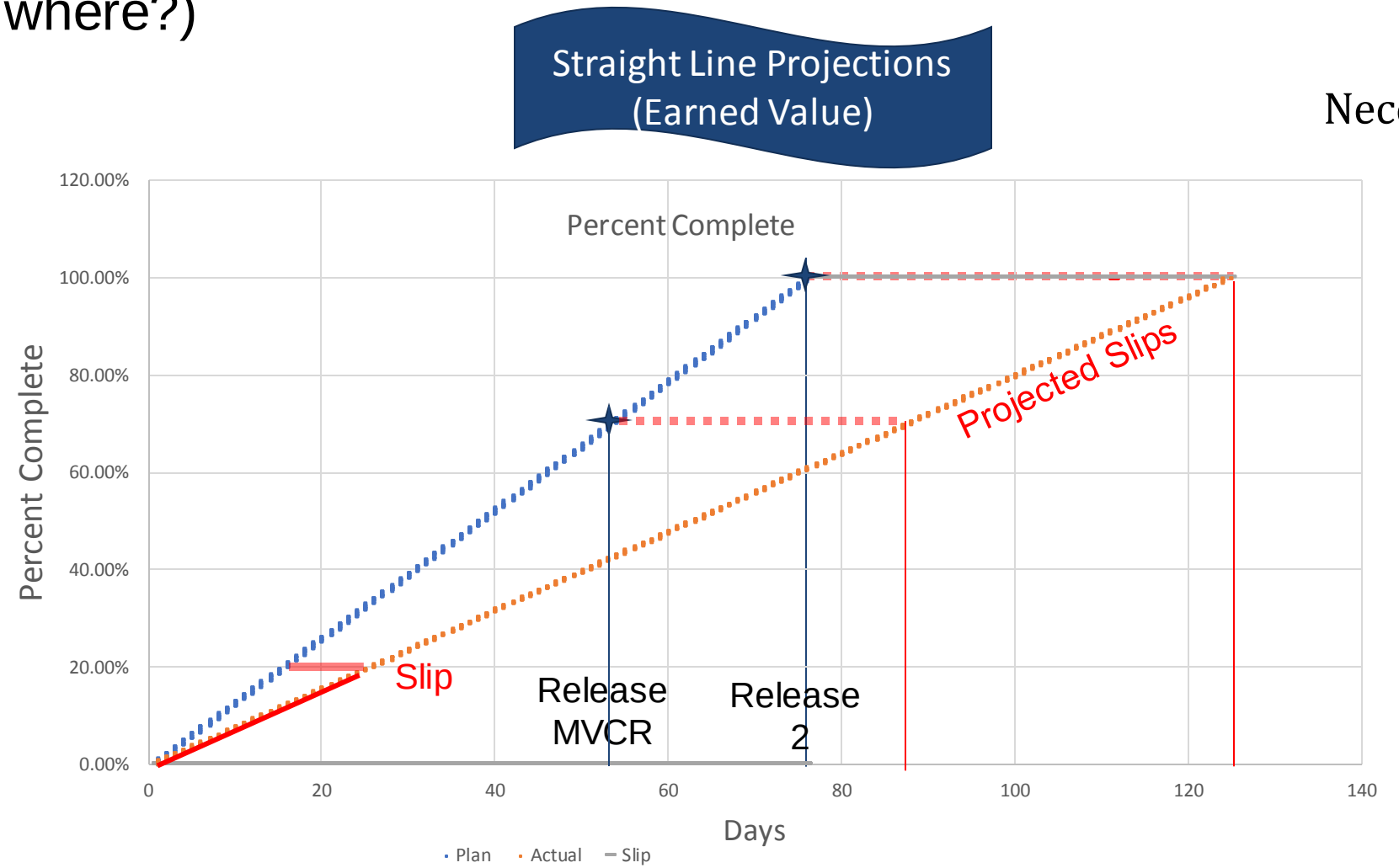
Model and simulate pipeline-of-pipeline systems.

Automate data collection and Program Management Status Reporting for DevSecOps pipelines.

Directly collect data from DevSecOps pipeline tools

- Automate data collection, storage, and reporting
- Correlate data to project outcomes
- Present completion to-date and milestone predictions to Program Management in smart dashboards

# Behind by 2 weeks, when will the next release deploy?
(to where?)




Necessary data and information

$\bar{t}_{lead}$ - lead time

$\bar{t}_{Factory}$ - factory (cycle) time

*Throughput* – output rate

$\overline{Staff\ Hours}$

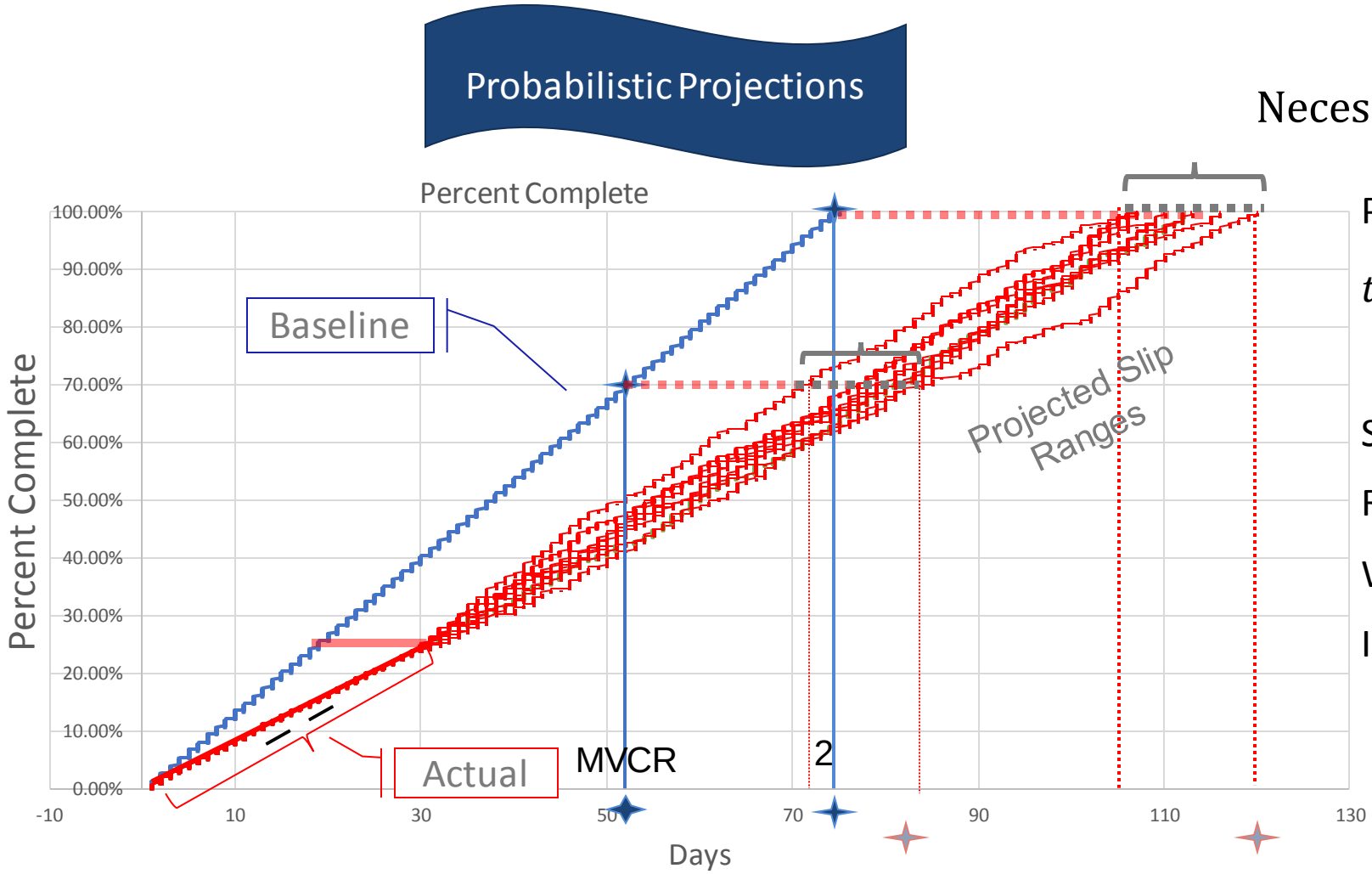
Roadmap

WBS (work packages)

IMS (time sequenced work)

This provides a reasonable estimate. But, not a **probability** of success, is **manual** (pivot tables?), uses last month's data, assumes all work is (more or less) the same, and doesn't scale well with multiple-pipelines.

# Range and confidence level requires **understanding** variation




Necessary data and information

Parameterizations for

$t^i_{lead}, t^i_{facory}, t^i_{deliver}$

Staff Hours($t$)

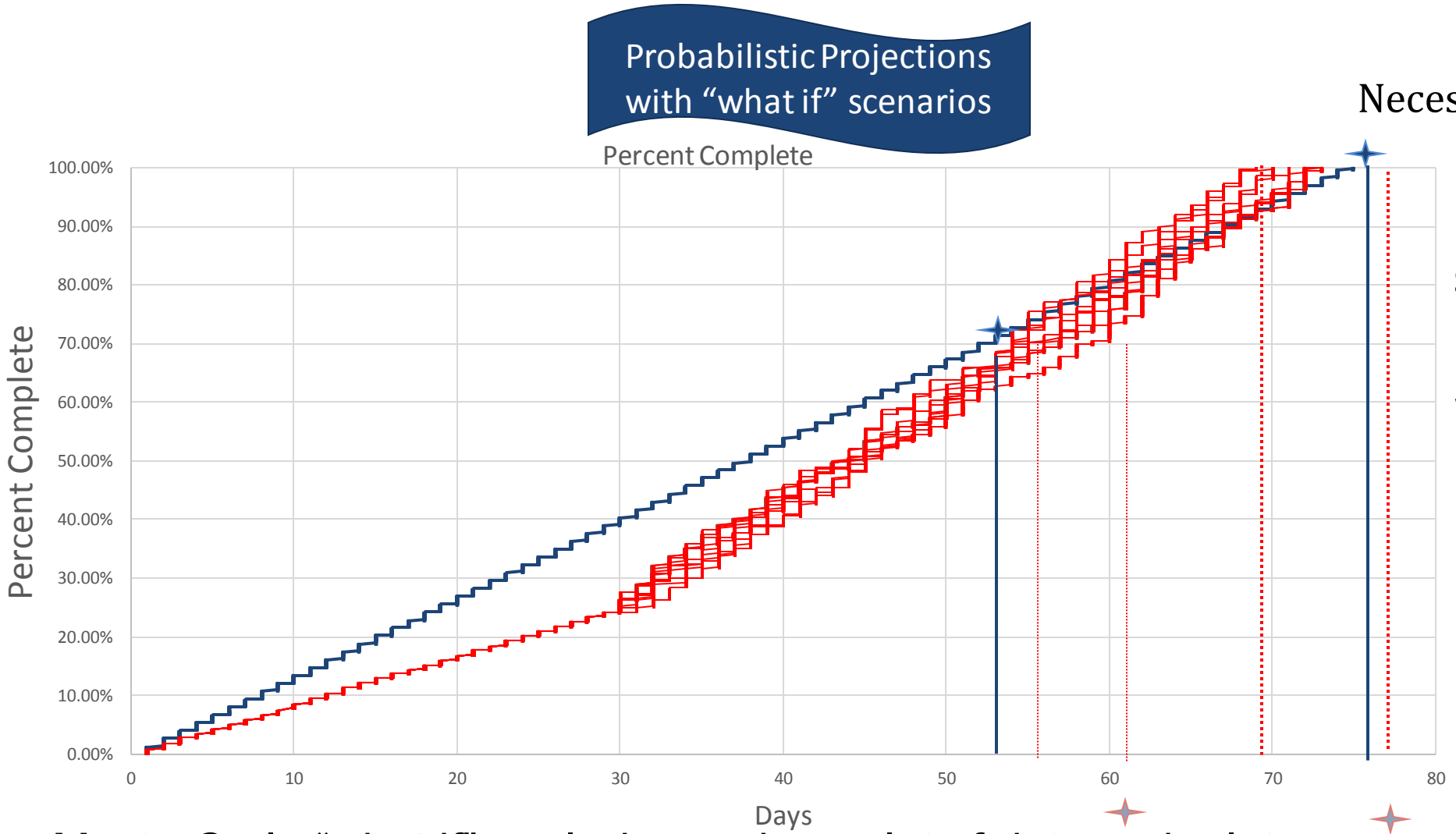
Roadmap

WBS

IMS

Using distributions to capture and analyze variation instead of using averages requires a not only maintaining a lot more data, but is also a lot more work!

# Add How Many Teams to Achieve Date Targets?




Necessary data and information

$t^i_{lead}, t^i_{facory}, t^i_{deliver}$

**Staff Hours(t)**

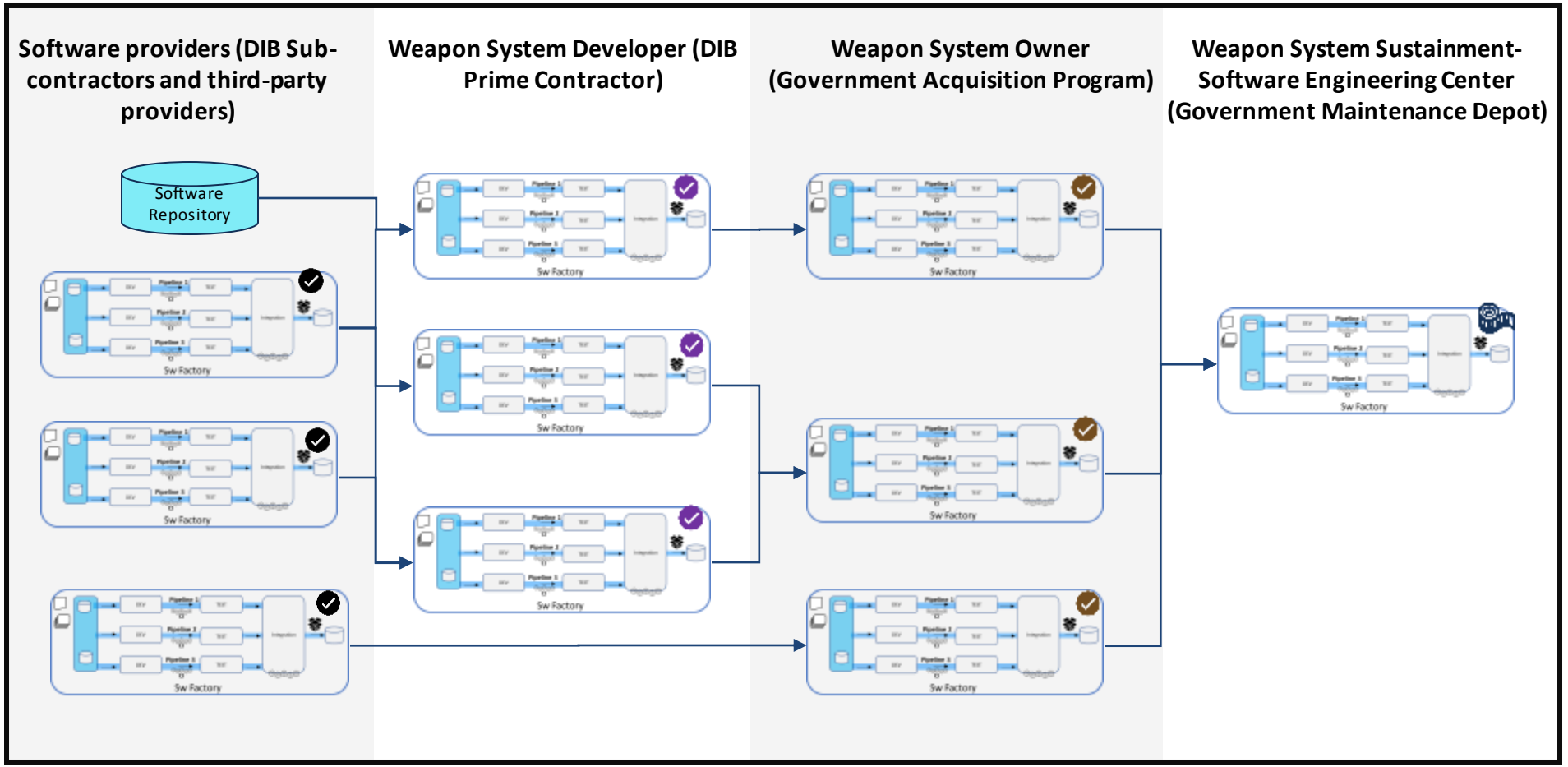
Roadmap

WBS

**IMS**

Monte Carlo "what if" analysis requires a lot of data and a lot more computation!

# Scaling Up! ‘In a PoPs Environment, When will we merge?



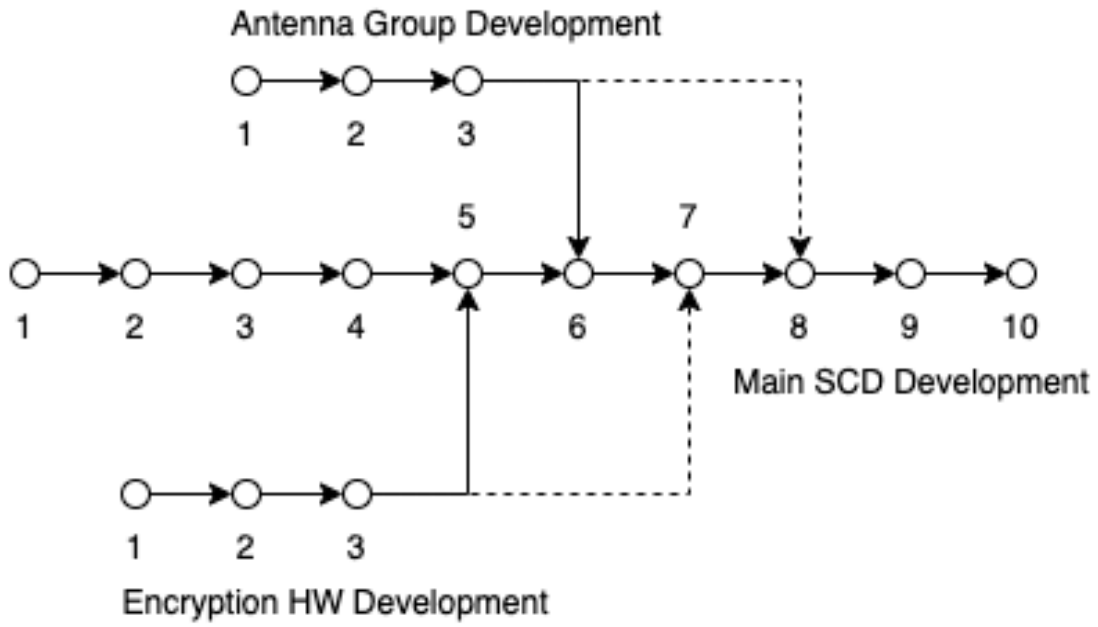

= Done

= Measure

= Tech Debt

= Size / SW Features / Systems Functions / Mission Capabilities

# Pipeline of Pipelines PoPs Workflow Network Example




Model a fictitious device that captures characteristics of a real project dependencies between hardware and software capabilities.

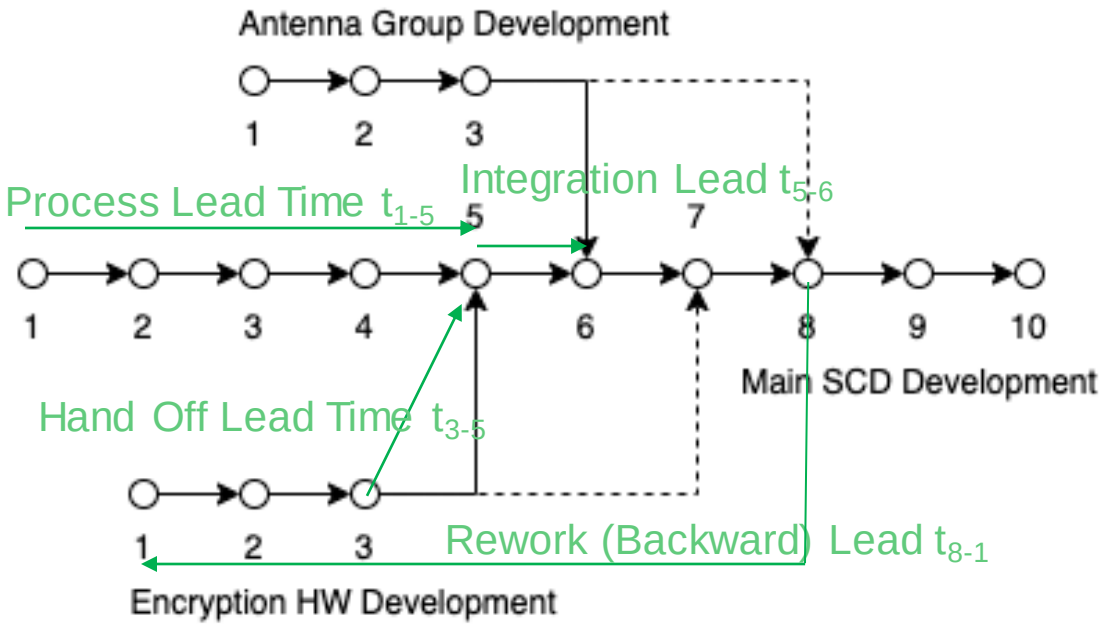
Different pipelines produce dependencies used to model schedule, cost, and technical performance risks resulting from production variation, accumulated variance, and rework.

All nodes are pipeline activities, arrows are lead times. Nodes 5,6,7, and 8 are integration or test points.

Little's Law assumptions are strongly violated except for **some** linear pipeline segments.

Typical Flow Metrics do not accommodate rework, merges, or multiple entry points

# What We Have: Multiple-Pipelines




$t_{1-5}$ Lead time enter development to integration

$t_{5-6}$ Lead time for successful integration

$t_{3-5}$ Cross Pipeline lead time exit 3 to enter 5

$t_{8-1}$ Issue return lead time, identify in 9 to re-enter 1

**Issues and novel observations:**

Pipeline->Pipeline transfer lead times affect **flow efficiency**, but these lead times do not appear to be measured in the programs we know of.

Backward lead times (lag times) was a problem validated by our collaborators but is not currently accounted for.

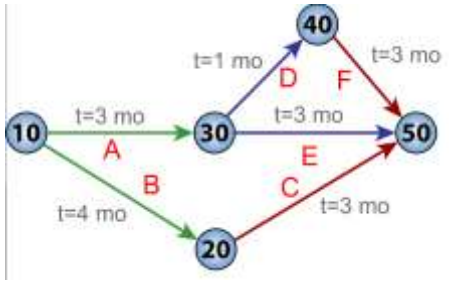
We expect different integrations to have different lead time distributions. (e.g. technical performance vs bug rework)

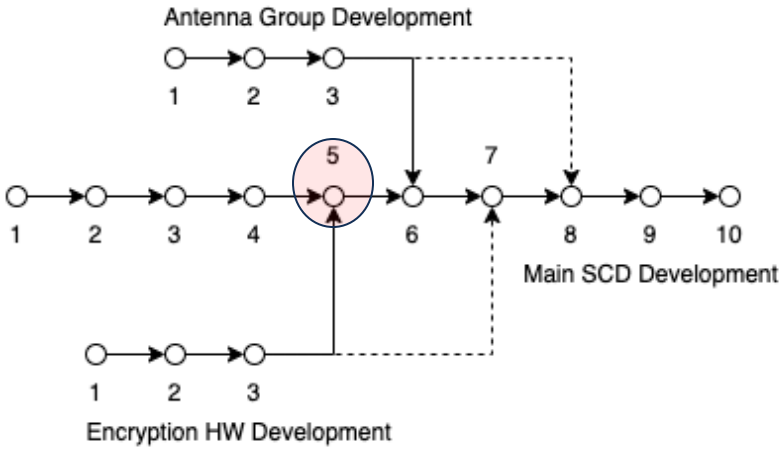
Our next version will use network Monte Carlo Simulation including new lead time measures (pipeline transfer leads, backward transfer)

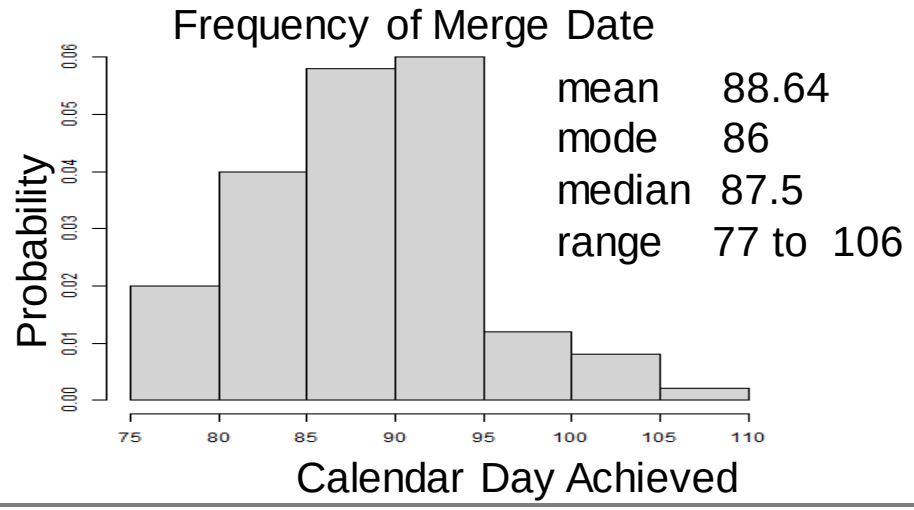
# Multi-Pipeline Projections: Approach

**Approach:**

- Assemble dependency network of work steps.
- Parameterize task durations.
- Use Monte Carol to generate many simulated histories



# What we need

**Data from DSO pipeline and other sources**

-Product state node structure

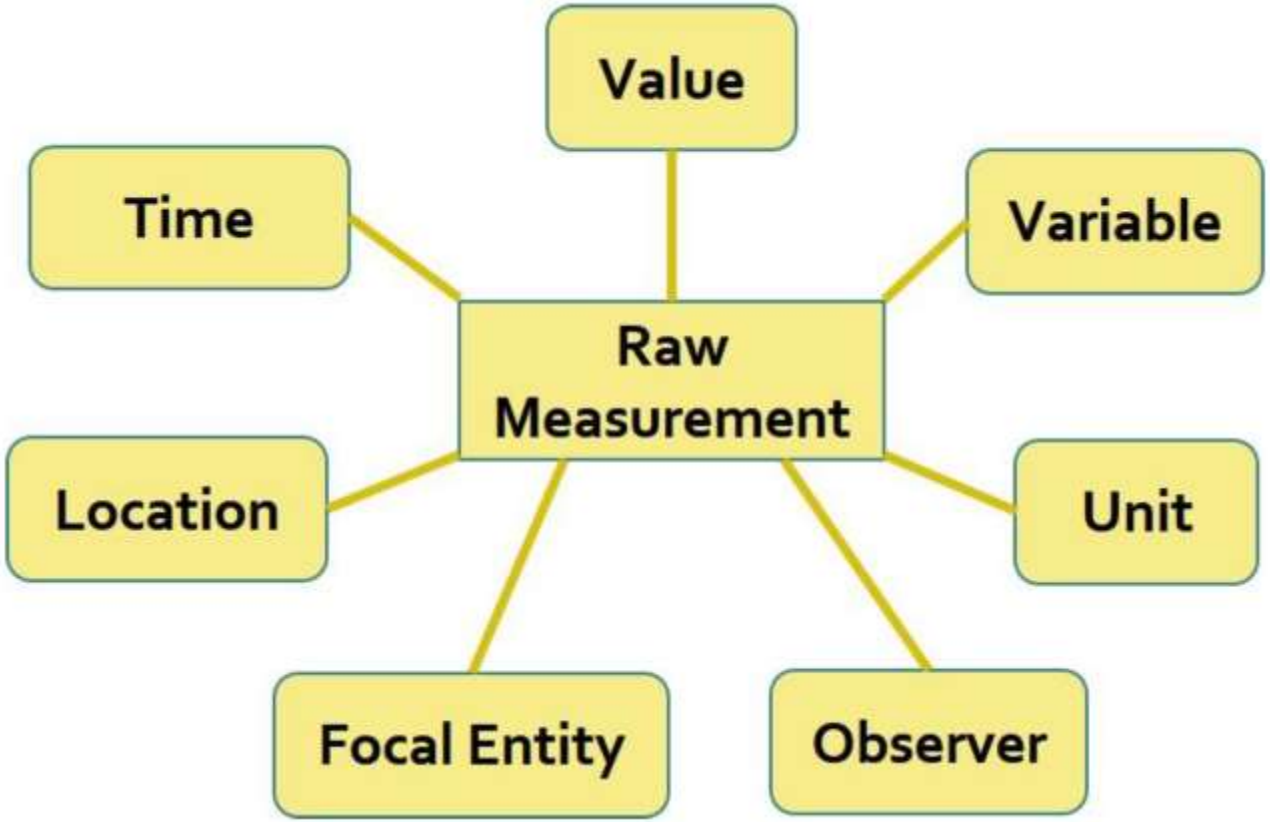
- capability based WBS and release plan,
- product dependencies,
- workflow

-For each Pipeline, empirical data for

- **Time Based Effort** and **Variation** (by skill?)
- **Production Rate and Variation** by **work type**
- **Primary, Rework, and Support** work by activity
- Defect Rates
- Defect Fix latencies

# Prototype DSO Measurement

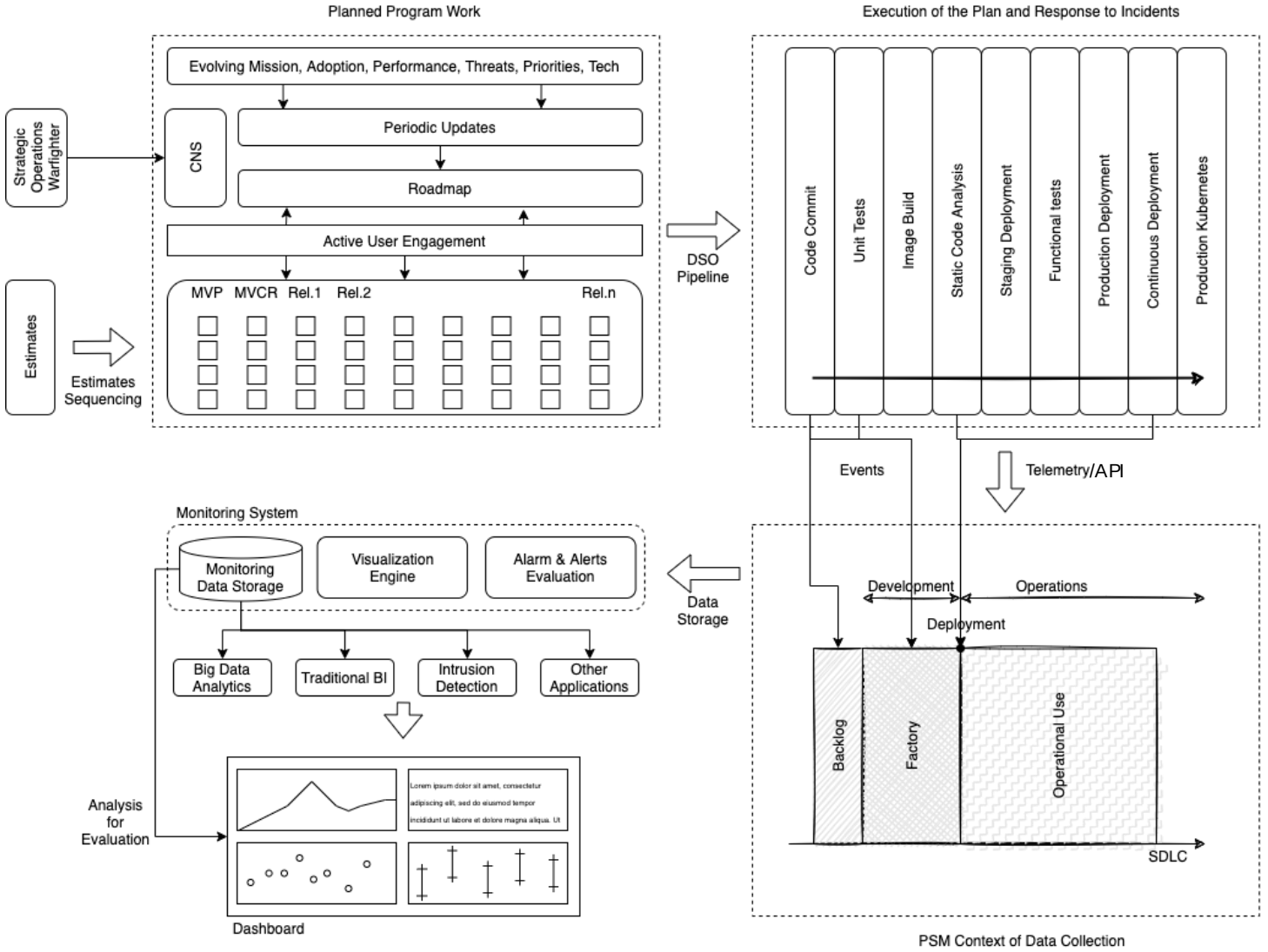
Measurement *"A set of observations that reduce uncertainty where the result is expressed as a quantity.“*



# How do we get the data? Data Collection Context

*Managed with Jira, Gitlab, Rally*

*Factory Pipelines*




**Planned work** includes the WBS, work packages, work sequencing, and estimates

Work packages **Execute** plan development stages, tools trigger events (time stamps, package labels,

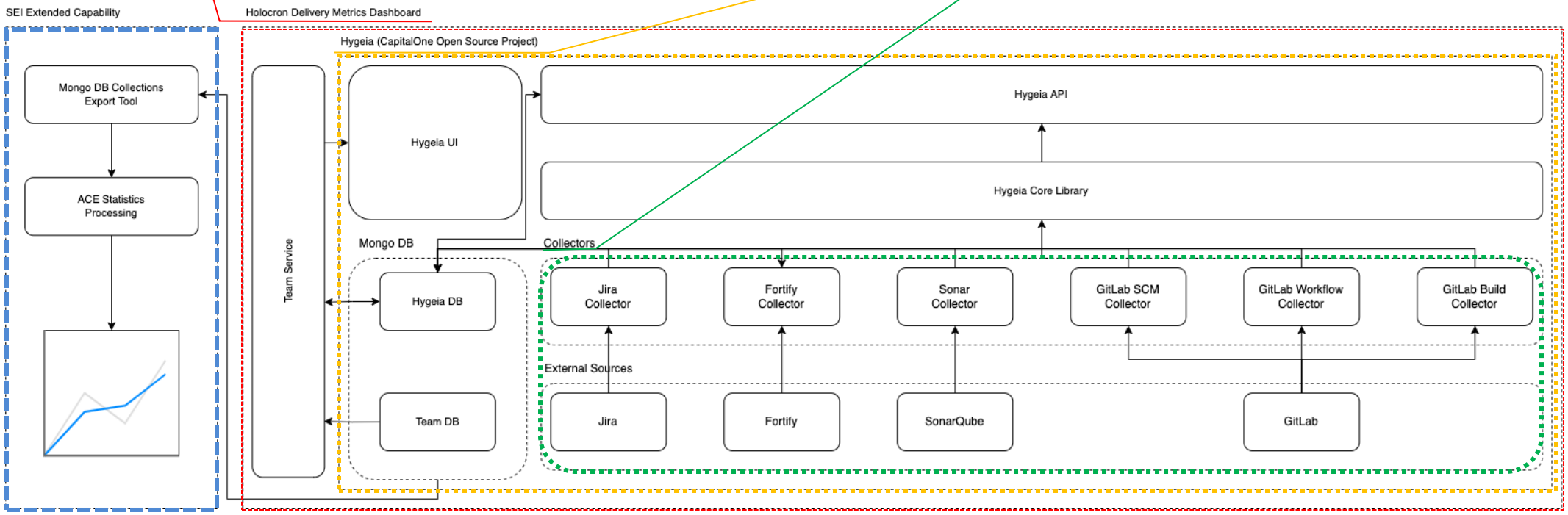
Data is collected and **Transformed** for storage

The **Warehouse** loads the data and provides the interface for analysis and dashboards

# Technical Approach: Using Hygeia/Holocron

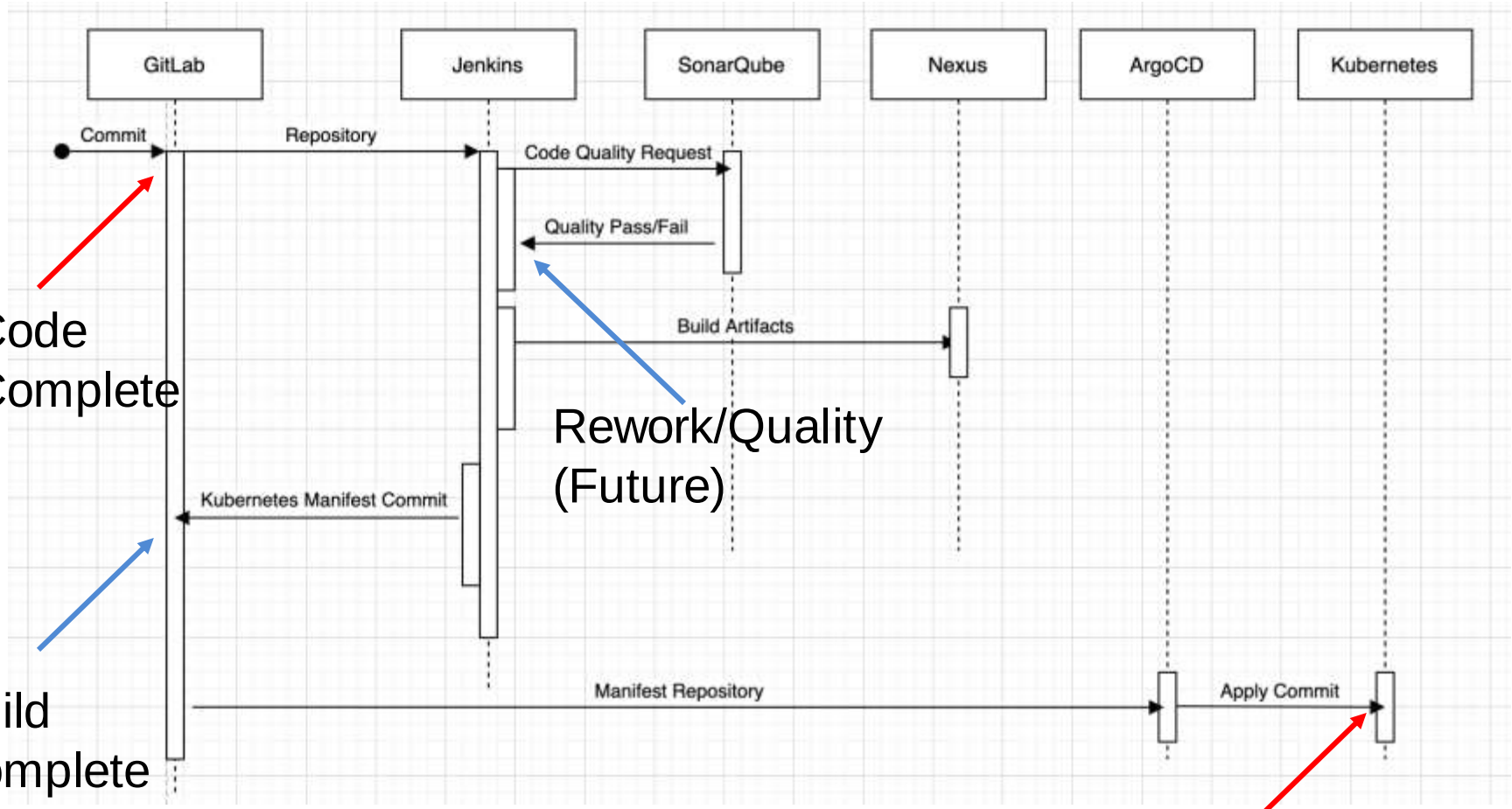
Apply on local development pipeline (instrumenting a local research project)

Holocron provided by Platform One, uses Hygeia Collectors




**External Sources**
- Jira
- Gitlab
- Sonarqube
- Fortify

# How do know work is DONE? Look inside the Pipeline




Code Complete

Rework/Quality (Future)

Build Complete

Deployed = Done

Date is collected from key events

The data specification is on the following slide

Use Labels to connect WBS, Roadmap, Backlog to work packages

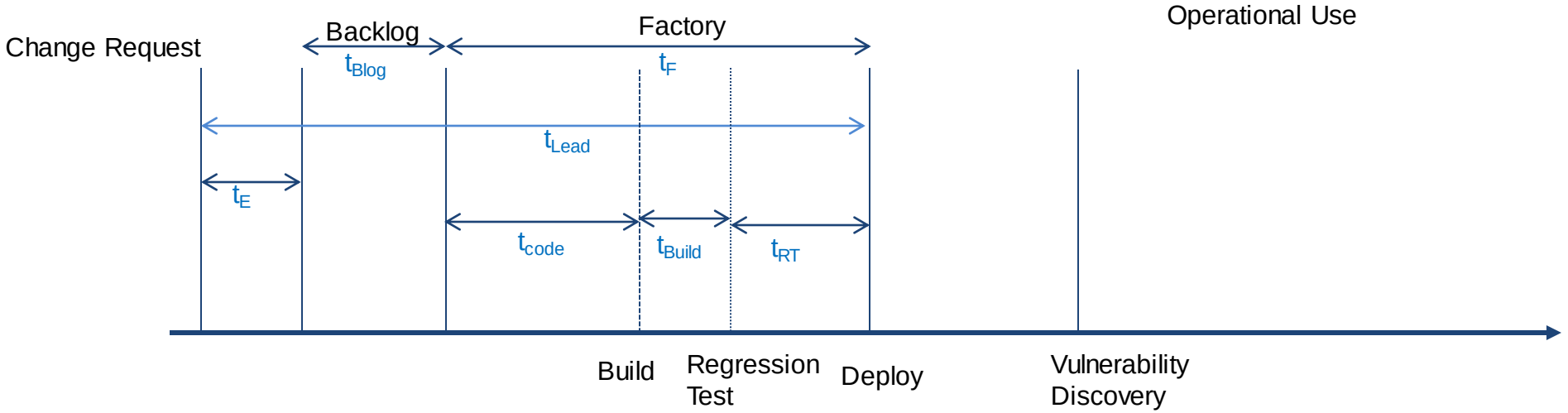
- Lead times,
- Estimated Dates
- Actual Times

# Vulnerability Fix, What do you Need to Know?

What if it's more than a rebuild?

I need some time ranges.

I must consider code and test.


$t_{Lead}$ Lead time

$t_E$ Evaluation time

$t_{Blog}$ Backlog time

$t_{Build}$ Build and test time

$t_{RT}$ Regression Test time

$t_F$ Factory time

$t_{Build}$ Build time

$t_{RT}$ Regression Test time

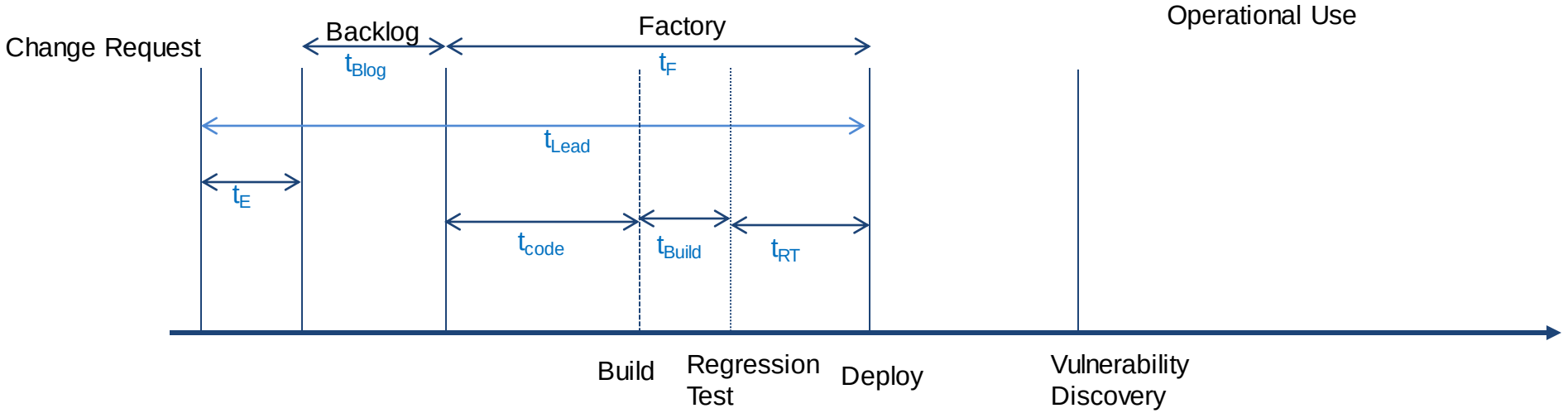
$t_F$ Factory time

# Some Lead times

What if it's more than a rebuild?

I need some time ranges.

I must consider code and test.

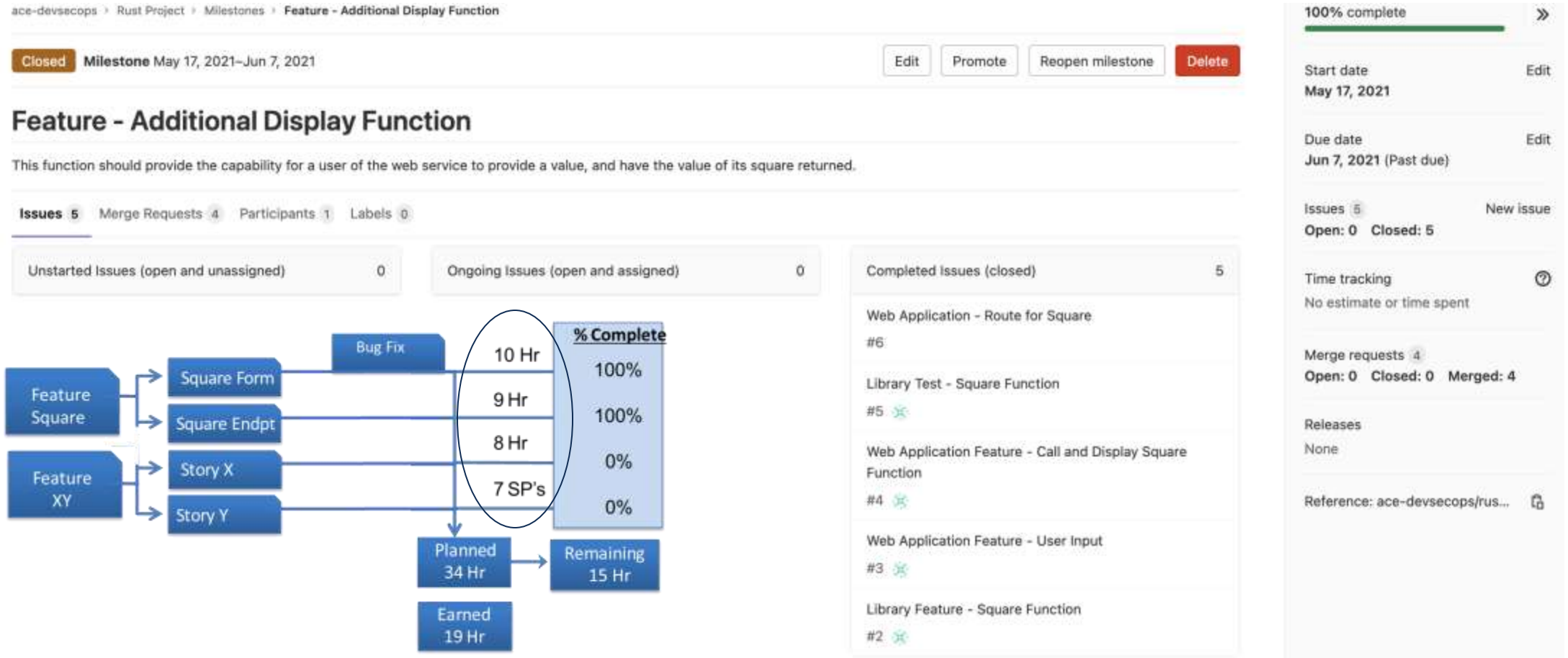

$t_{Lead}$ Lead time
$t_E$ Evaluation time
$t_{Blog}$ Backlog time

$t_{Build}$ Build and test time
$t_{RT}$ Regression Test time
$t_F$ Factory time

$t_{Build}$ Build time
$t_{RT}$ Regression Test time
$t_F$ Factory time

# Tracking Completion Through Gitlab and Pipeline

## Work Completion



# Status of Feature Physical Percent Complete

**Export done status of work items with pipeline workflow data including "Done" stages**
**Export the estimates and completion data for the Feature from Issue tracking.**
**Reconcile that and verify that the export includes all**

- **Features for the Project,**
- **supporting Stories and Tasks,**
- **the Task Est hours and**
- **task completion status.**

**Apply the Physical Percent Complete to all Features in the IMS.**
**Update any other work in the IMS not captured.**
**With the Physical Percent Complete data and Roadmap, IMS, and Dependencies**
**Compute nominal – ETC, EAC, CPI, SPI, CV, SV, TCPI**
**Compute stochastic propagation effects of actual vs plan.**

**Build dashboard displays for percent complete and range of estimates at complete**

**Assess impacts among teams for propagation effects of delay, potential conflicts, blocked work, or unfavorable outcomes.**
**Assess corrective actions**

# Multi-Pipeline Projections: Data and Challenges

Challenges Include:

- Surrounding tools such as GitLab and Jira manage the tasking
  - Use a front end (Hygeia) isolate the interface with issue tracking tools
  - Human in the loop with issue tracking tools
- Confounders such as effort variation
  - Confounders may not be visible, but add noise to the measurements
  - Include **assumptions** such as full time-team member allocation
- Pipelines may have different iteration cadence, especially across boundaries
- What additional contextual information is required?
  - How do we select work building toward a capability?
    - Use labels and WBS hierarchy
  - Are there distinct work types (components, sub-domains, front-end/back-end) that affect work rates or variation?
  - How to separate primary work from rework

# Observations and Lessons Learned

At this time, measurement tools are too siloed to work together (this is related to findings from our Digital Engineering work)

Averages don't support statistical modeling or high priority changes (not everything is average). We need distributions.

We need more specific lead time measures for process steps and baselines for zero rework lead times. (total time until test completes is a candidate quality proxy)

Automated timestamps from the toolchain provide insight into progress and workflows.

Work packages should be categorized (bug, enhancement, …) not only because different types of work have different characteristics, but also to gain insight into process health. (a bug vs new capability, vs process sustainment)

Typical flow metrics don't appear to apply to the pipeline-of-pipelines because of branching and other assumptions violations.

# Call to Action

Would you benefit from continuous updates to status and projections?

Are you using DevSecOps tool chains, Issue trackers, and workflow management?

Can you share process data and discuss results?

Will you participate in our quarterly research review?

We can help

- Specify information, data, and displays
- Recommend tools and approaches
- Evaluate your results for effectiveness

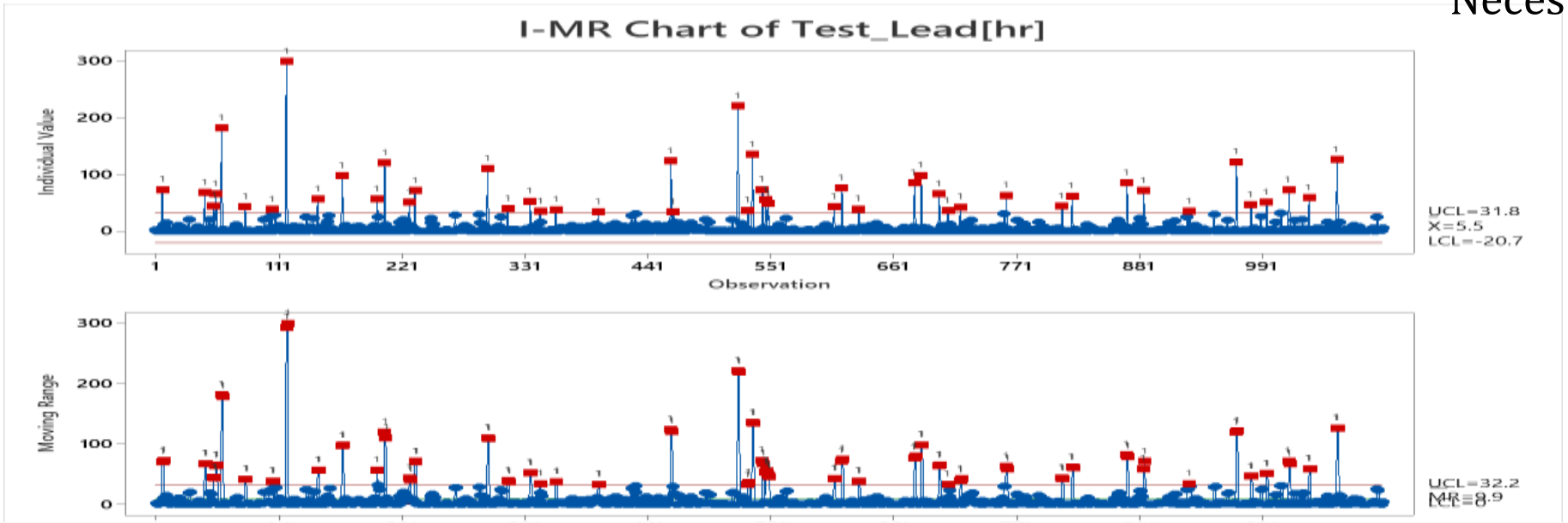
# Thank You!

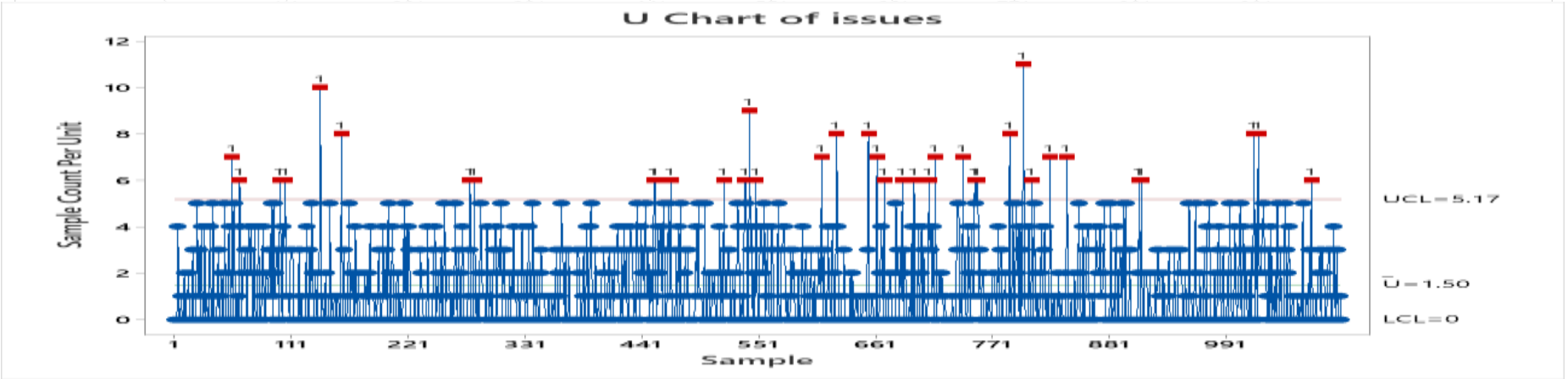
If you would like to work with us, or have any questions,

Please contact

Bill Nichols wrn@sei.cmu.edu

# What About Issues Found in Test?



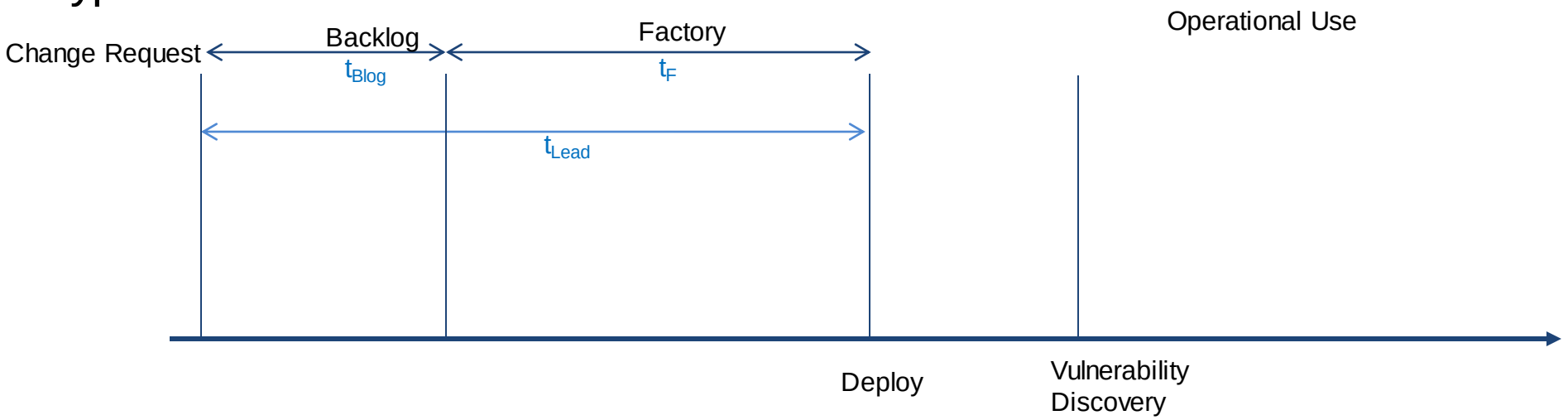

Necessary data and information

$t^i_{build}, t^i_{test}$

# Vulnerability Fix: What do you Need to Know?

How long to fix a single vulnerability? (or add a single feature?)

Typical Measures




$t_{Lead}$ Lead time

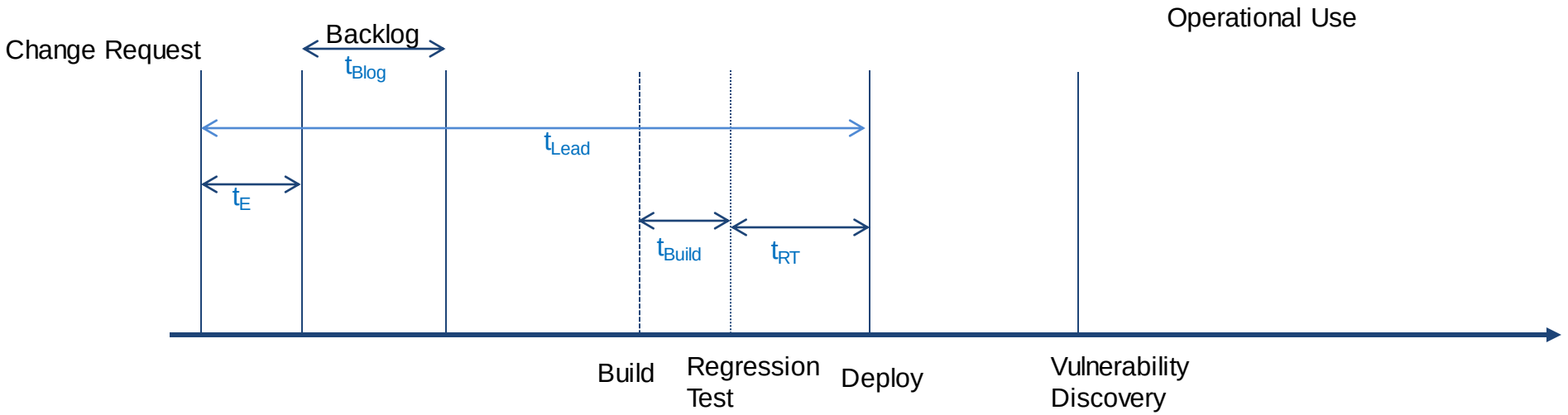
$t_{Blog}$ Backlog time $t_F$ Factory time (aka Cycle time)

# Vulnerability Fix, What do you Need to Know?

Isn't this top priority? (How fast can I have?)

Don't I need to evaluate. Is this just a rebuild or coding? How deep is the supply chain?

I don't want **average** times, I want **no wait** times.




$t_{Lead}$ Minimul Lead time
$t_E$ Evaluation time
$t_{Blog}$ Min Backlog time

$t_{Build\_0}$ No Rework Build time
$t_{RT\_0}$ No Rework egression Test time

# Some Definitions

**Roadmap** A high-level visual summary that maps out the vision and direction of product offerings over time. It describes the goals and features of each software iteration and increment. (DoDI 5000.87)

**WBS** A Work Breakdown Structure is a hierarchical decomposition of work tasks that need to be performed by project team members to accomplish project goals and objectives and create the required deliverable. (AcqNotes)

**IMP** -The Integrated Master Plan (IMP) is an event-based, top-level plan consisting of a hierarchy of program events, each supported by accomplishment criteria for goals.

**IMS** - The Integrated Master Schedule (IMS) is a time-based schedule containing the networked, detailed tasks necessary to ensure successful program/contract execution. The IMS is used to verify attainability of objectives