# When Is It Feasible to Use the Software Acquisition Pathway?

## ...and when should you start?

David Tate and John Bailey

May 2022

# DoDI 5000.87: "The PM and the sponsor... will define a Minimum Viable Capability Release (MVCR)"

An MVCR has <span style="color:red">fixed mandatory requirements</span>

Delivers initial warfighting capabilities

Enhances mission

Analogous to a commercial minimum marketable product

MVCR must have

- Authority to Operate / safety releases / etc.
- Architecture to support at least annual upgrades
- Effectiveness and suitability

**Challenge:  How much MVCR functionality can be ready to field in 1 year?**

# Derived MVCR requirements

## Architecture

- Cybersecurity architecture
- MOSA / IP strategy to support future agile iteration
- Standards compliance
- Interoperability with specific other systems

## Test, evaluation, verification, and validation

- Safety
- Cybersecurity
- Interoperability
- Effectiveness and suitability
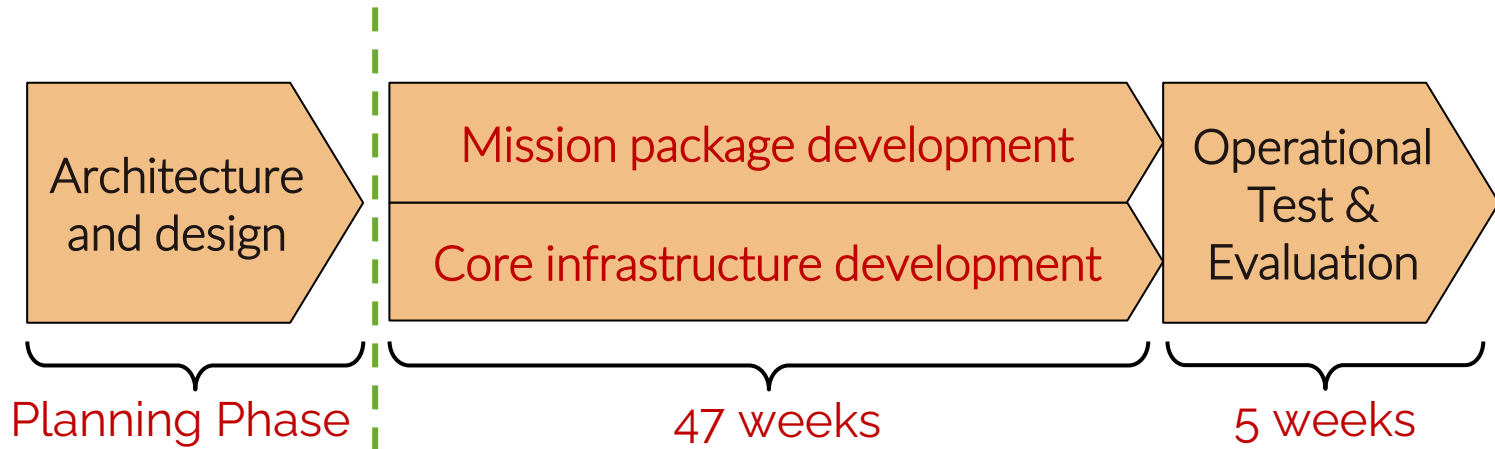- Authority to Operate

## Mission packages

- Provide minimal useful capability

## Core infrastructure code

- As necessary to support mission software

# Maximally optimistic development timeline – 1 year

| Architecture and design | Mission package development | Operational Test & Evaluation |
|---|---|---|
| | Core infrastructure development | |

Planning Phase    47 weeks    5 weeks

Best case: 47 weeks of actual software development

- Assumes complete CNS and major design / architecture choices *before* start of spending

- Assumes integrated T&E, cATO from the beginning

- Assumes clear requirements, fastest possible OT&E

**How many ESLOC can be finished in 47 weeks?**

# Estimating best-case output of a 47-week development

We used COCOMO II to estimate how many equivalent source

lines of code (ESLOC) could be finished in 47 weeks

- Settings chosen for shortest schedule (not lowest cost)
- Optimistic assumptions: maximum use of software tools, maximum developer process maturity, assumed "inception phase" complete

**Best case: ~28,000 ESLOC ...How much code is that?**

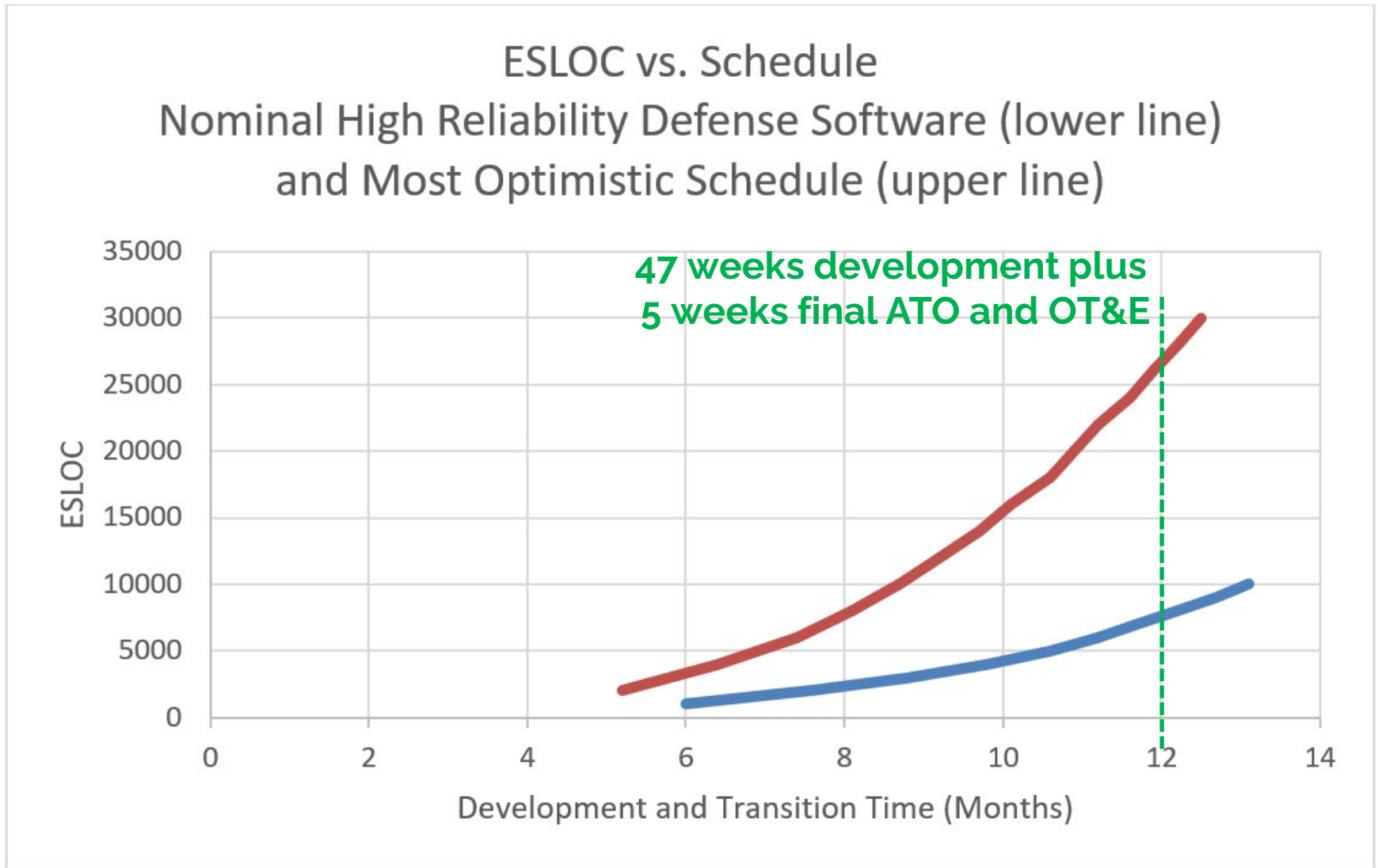*If* we can do infrastructure and mission systems in parallel, and

*If* the infrastructure code can be 90% reused and tool-supplied

- ~28,000 SLOC of mission system
- ~200,000 SLOC of infrastructure

This is an ***upper bound***, not a prediction

**Most projects will not meet these optimistic assumptions**

# COCOMO-II ESLOC Modeling



ESLOC vs. Schedule
Nominal High Reliability Defense Software (lower line)
and Most Optimistic Schedule (upper line)

47 weeks development plus
5 weeks final ATO and OT&E

# How much capability is that?

Capers Jones assessed 265 DoD software efforts. His data showed that perhaps 1/3 of those had fewer than 200,000 SLOC, some fewer than 50,000.

**Some DoD programs are small enough to meet the MVCR deadline**

As an example, AMF JTRS development required > 2 million SLOC core infrastructure

- Even a pared-down infrastructure for less capability would have been much larger than 200,000 SLOC

**Many projects could *not* meet the MVCR deadline**

2/3 of projects in past sample, possibly more today

# The constraint appears to be binding



Our excursions and samples show that many typical DoD software efforts are too big to finish the MVCR in 1 year, **even with optimistic assumptions about the developers and the difficulty of the project**

**In practice, significant useful capability within 1 year might be the exception for new start programs, not the rule**

**Post-MVCR transition to SWP\* may be a better option for many projects**

\*See Backups for pros and cons

# Bottom Line at the Bottom (BLAB)

The "valid operational software within 1 year from start of funding" requirement may be more binding than was envisioned by the drafters

Typical conditions will provide challenges we ignored

As will embedded software and unique infrastructures

Starting with the right architecture is vital – we need to ensure that SWP doesn't create a perverse incentive to skimp.

Decision-makers need to be aware of these factors when deciding whether to approve SWP for a specific program.
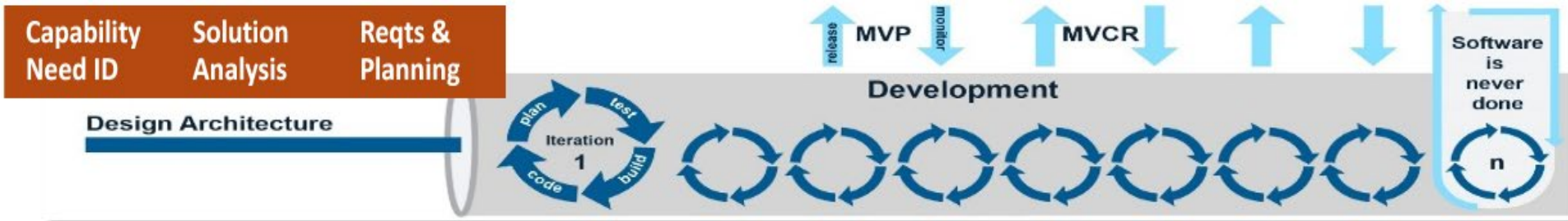
PMs need to be aware of how much software they need when choosing an acquisition strategy and pathway(s).
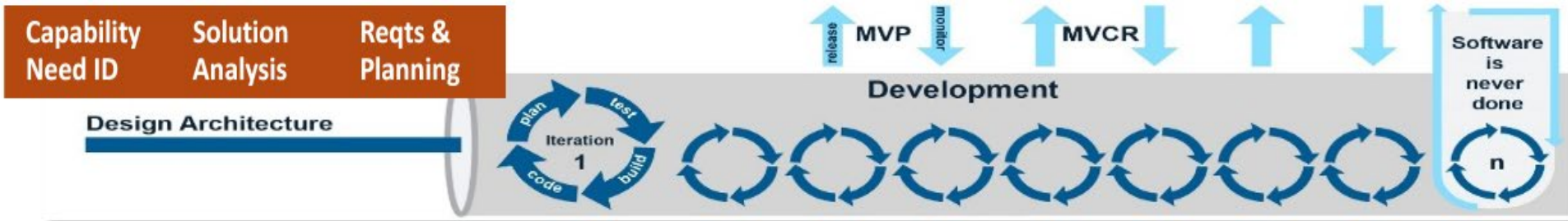
# Backup Slides

# SWP Intro



DoDD 5000.87 defines a **Software Acquisition Pathway** (SWP) as part of the Adaptive Acquisition Framework.

The intent of the pathway is to enable **rapid deployment** and **frequent ongoing upgrades** to software-intensive defense systems.

Diagram from OUSD(A&S) "Let's Talk Agile – Take Two" October 2021

# SWP Phases and Milestones



The pathway consists of a **Planning Phase** and an **Execution Phase**

The Planning Phase produces
- A **Capability Need Statement (CNS)** defining the purpose
- An **acquisition strategy (AS)** describing how it will be achieved
- A **system architecture** and design concept

# "The PM and the sponsor ...will define a Minimum Viable Capability Release (MVCR)"

The CNS and AS are precursor products.

The CNS provides an initial list of prioritized missions and capabilities.

The AS describes how the program will achieve and maintain **continuous authority to operate (cATO)**.

**MVCR**:
"The initial set of features **suitable to be fielded to an operational environment** that **provides value to the warfighter or end user** in a rapid timeline. The MVCR delivers initial warfighting capabilities to enhance some mission outcomes. The MVCR is analogous to a minimum marketable product in commercial industry."

# Estimating best-case output of a 47-week development

We used COCOMO II to estimate how many equivalent source lines of code (ESLOC) could be finished in 47 weeks *under the best possible conditions:*

- Assumed maximum schedule compression (not lowest cost)
- Assumed maximum use of software tools, maximum developer process maturity CMMI 5
- Assumed "inception phase" complete
- Assumed no requirements volatility

COCOMO II is calibrated to waterfall development – but remember that the MVCR is (approximately) a waterfall project. Requirements are fixed and untradeable.

# Other options #1 – the <u>wrong</u> way

We could leave out key architectural features in order to finish *something* in less than a year

- Skimp on safety, cybersecurity, or suitability
- Don't bother with MOSA or other agility-enhancing restrictions
- Release to the field before really operationally useful
- Pretend that the MVP is really an MVCR
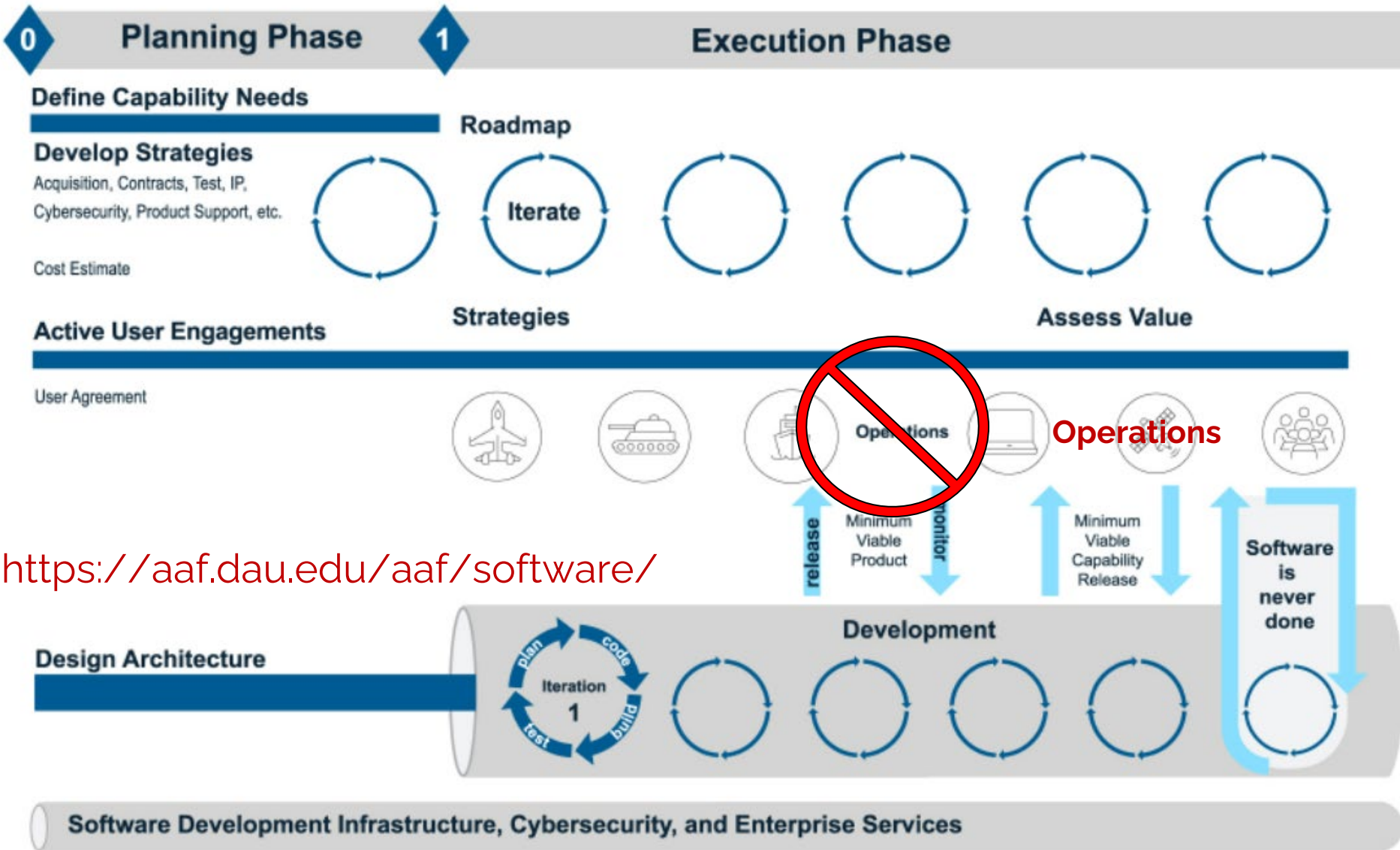
# Other options #2 – the <u>right</u> way

Start programs on a different pathway (e.g., MTA); transition to SWP later

- Pros:
  - Gives time for architecture/design to fully support eventual annual (or faster) capability upgrades
  - Allows users to provide MVP feedback to guide design of MVCR user interfaces and functionality
  - Allows development at a more cost-efficient pace if desired
  - Removes uncertainty about meeting 1-year deadline
  - Reduces risk of rushed MVCR implementations requiring later rework
- Cons:
  - Potential paperwork, depending on initial pathway used
  - Potential friction in transition from original pathway to SWP
  - Effort and time to obtain approval for transition

Can also transition existing mature programs to SWP where appropriate

- Caveat:
  Existing software architecture must support 12-month release cycles

# MVP ≠ MVCR!



https://aaf.dau.edu/aaf/software/