# How Should We Estimate Agile Software Development Projects And What Data Do We Need?

Glen B. Alleman, Prime PM
Thomas J. Coonce, Institute for Defense Analyses

International Cost Estimating & Analysis Association
2017 Professional Development & Training Workshop
Portland Marriott Downtown Waterfront Portland, Oregon
June 6 - 9, 2017

## Agenda

- Overview of Software Development Life Cycle Models
- Why traditional parametric estimating tools do not help estimate a software project developed using the Agile model
- Explain and demonstrate the "nearest neighbor" analogy technique to estimate Agile software projects
- Data and actions needed to implement the nearest neighbor technique

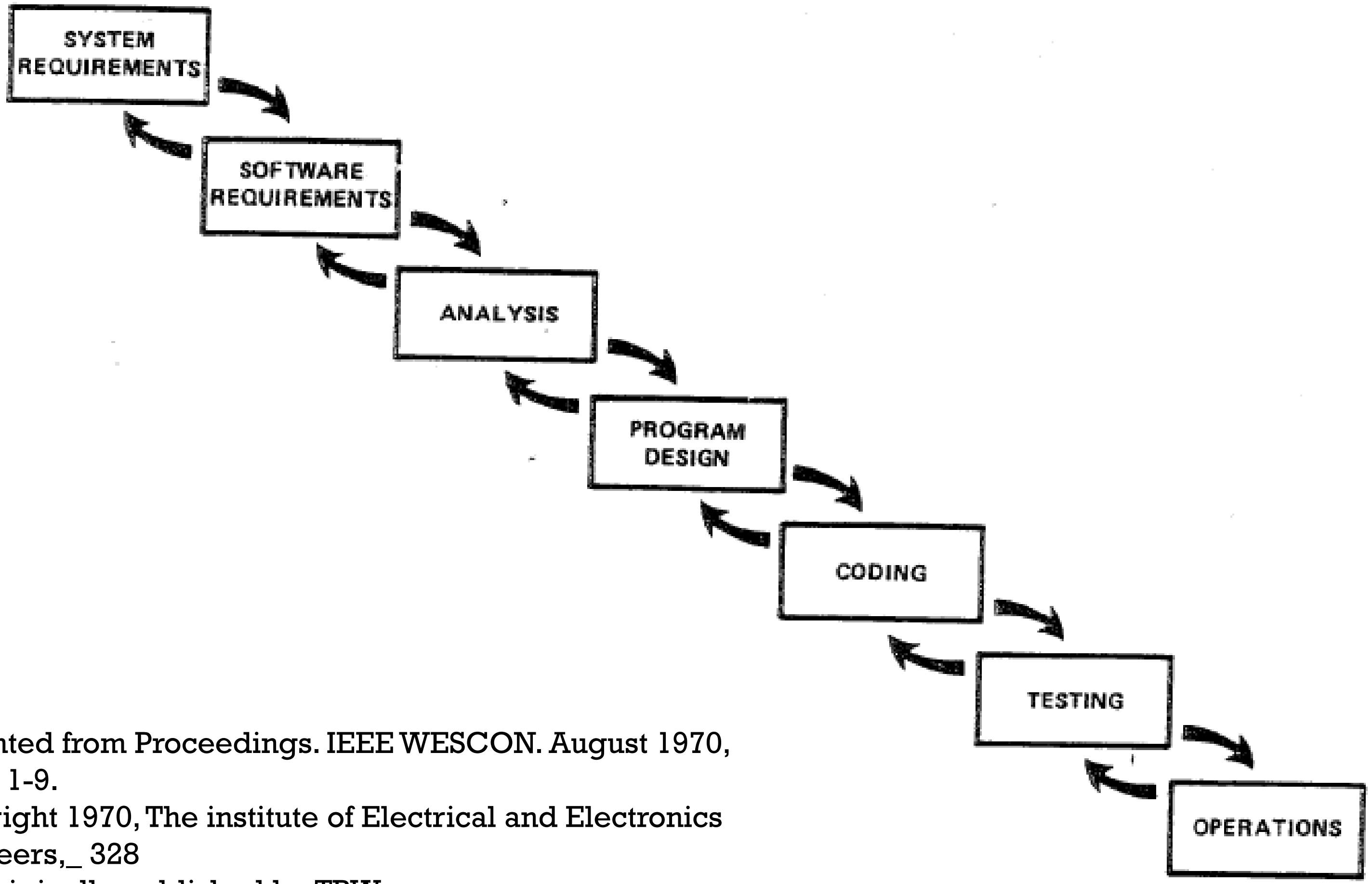
## Learning Objectives

- Understand different Software Life Cycle Development Models
- Understand how traditional parametric tools are not appropriate for agile-developed software
- Understand the cost and schedule drivers of an Agile-developed SW project
- Understand a proposed "nearest neighbor" analogy method of estimating Agile SW projects
- View minor revisions needed to DoD's Software Resource Data Report (SRDR) to support this method

## Software Development Life Cycle Models

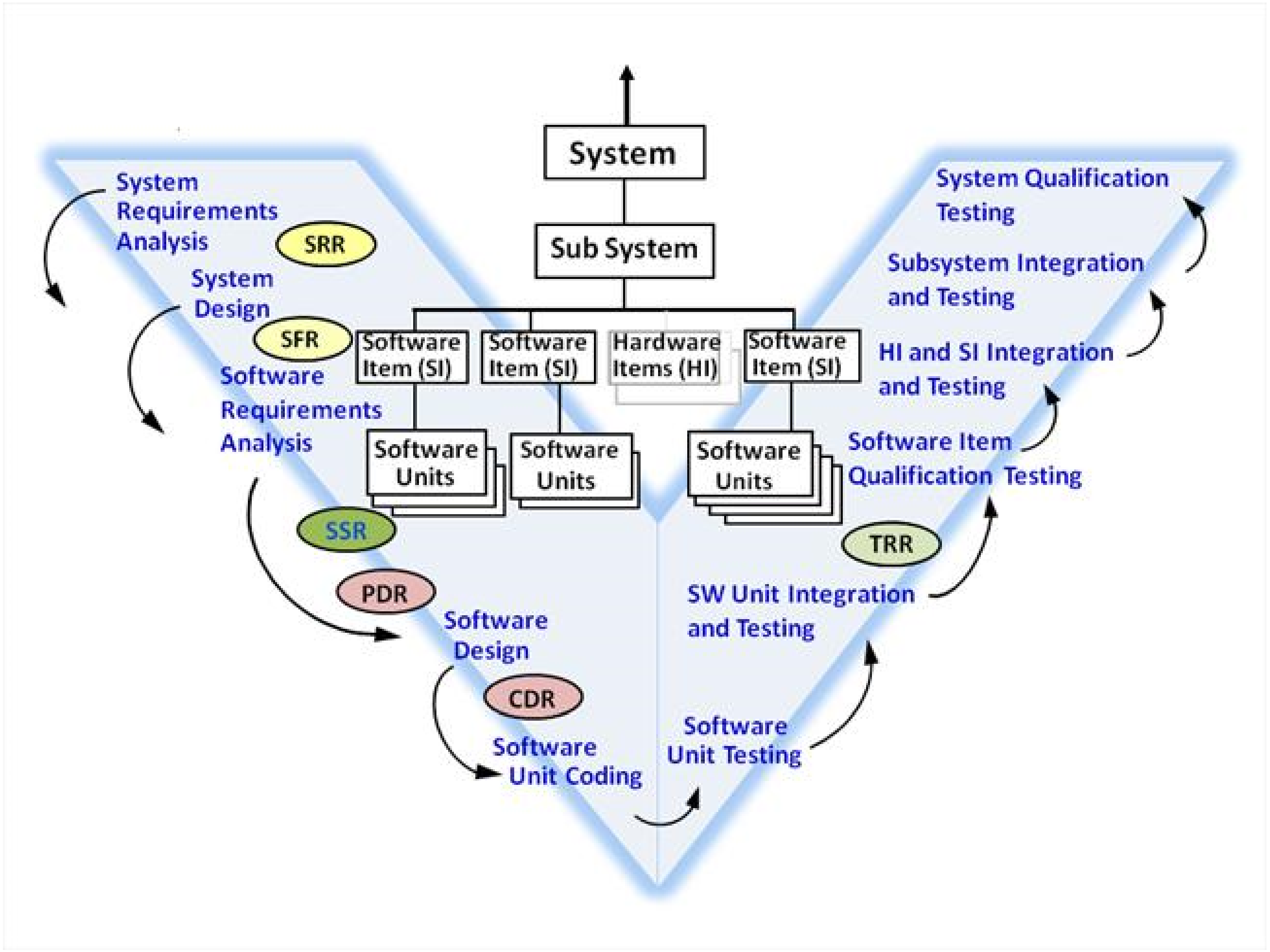
- Waterfall
- Incremental development;
- Evolutionary development;
- Rapid application development (RAD);
- Spiral development; and
- Agile development

# Waterfall Software Development Model



Reprinted from Proceedings. IEEE WESCON. August 1970, pages 1-9.
Copyright 1970, The institute of Electrical and Electronics Engineers,_ 328
Inc. Originally published by TRW

# Traditional "Waterfall" Development Process


System
Sub System
Software Item (SI)
Software Item (SI)
Hardware Items (HI)
Software Item (SI)
Software Units
Software Units
Software Units
System Requirements Analysis
SRR
System Design
SFR
Software Requirements Analysis
SSR
PDR
Software Design
CDR
Software Unit Coding
Software Unit Testing
SW Unit Integration and Testing
TRR
Software Item Qualification Testing
HI and SI Integration and Testing
Subsystem Integration and Testing
System Qualification Testing

Source: ACC.dau.mil

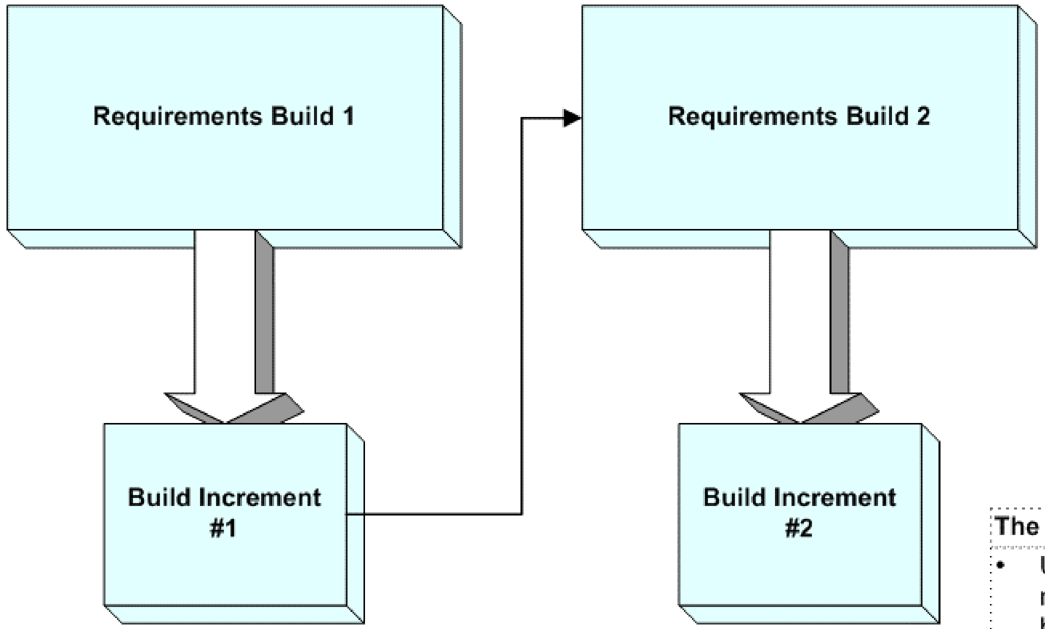
## Incremental Development Model

- The incremental build model is a method where the product is designed, implemented and tested incrementally (a little more is added each time) until the product is finished. It involves both development and maintenance. The product is defined as finished when it satisfies all of its requirements. This model combines the elements of the waterfall model with the iterative philosophy of prototyping.

Source: https://en.wikipedia.org/wiki/Incremental_build_model

# Evolutionary Development Model

Methodology: Software Life Cycle Process Management: The Evolutionary Model



The next logical step in life cycle development is the ***evolutionary*** model, which explicitly extends the incremental model to the requirements phase/ Figure illustrates this, showing that the first build increment is used to refine the requirement s for a second build increment.

**The advantages of the evolutionary model are as follows:**

- The model can be used when the requirements cannot or will not be specified.
- The user can experiment with the system to improve the requirements.
- Greater user/acquirer involvement is required than in the waterfall method.

**The disadvantages are:**

- Use of the method is exploratory in nature and therefore constitutes a high-risk endeavor. Strong management is required.
- This method is used as an excuse for hacking to avoid documenting the requirements or design, even if they are well understood.
- Users/acquirers do not understand the nature of the approach and can be disappointed when results are unsatisfactory.

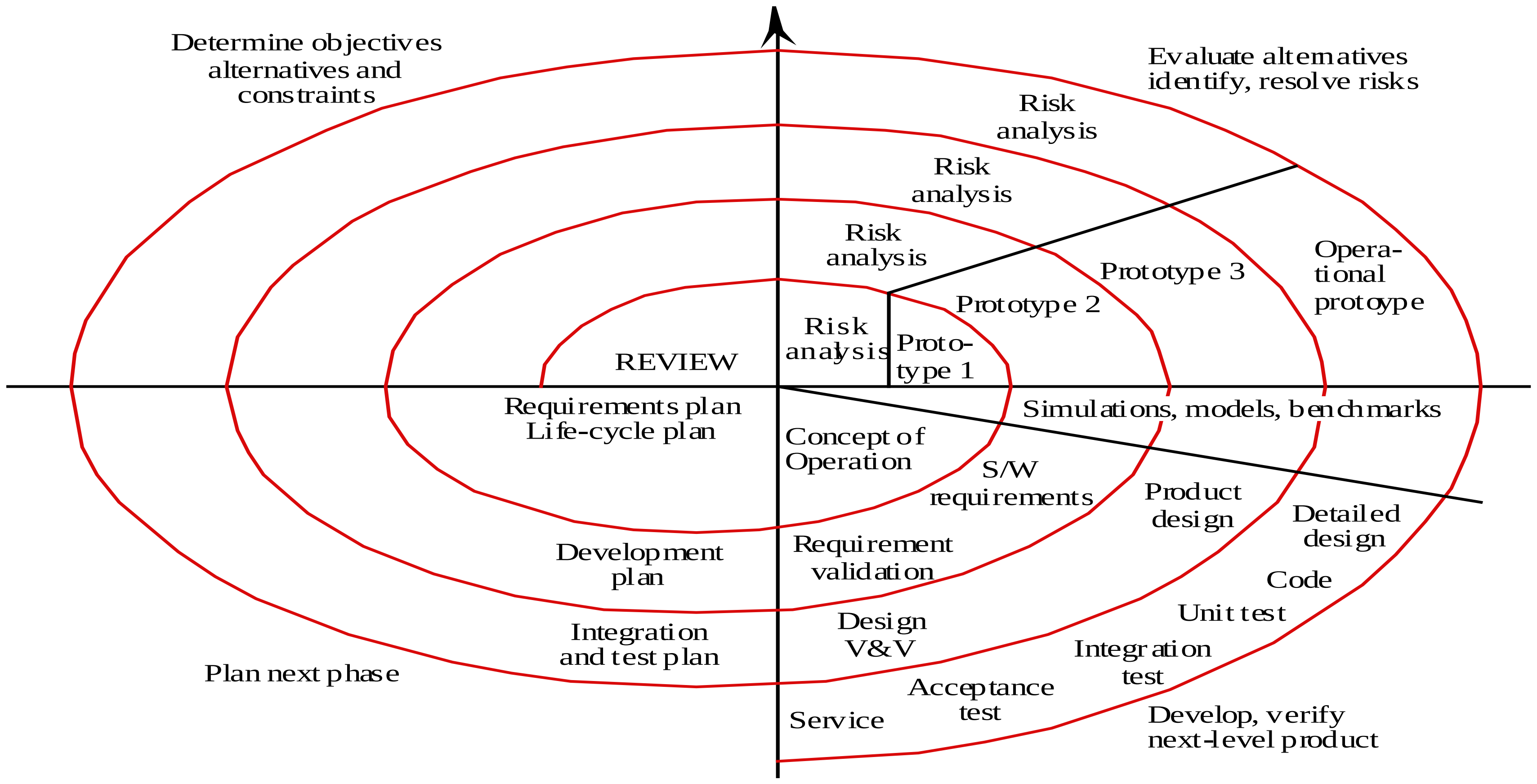
Source: http://myprojects.kostigoff.net/methodology/development_models/pages/evolutionary.htm

## Rapid Application Development Model

- Rapid Application Development model is a type of incremental model. In RAD model, the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype. This can quickly give the customer something to see and use and to provide feedback regarding the delivery <u>and their requirements</u>.

https://en.wikipedia.org/wiki/Software_development_process

# Spiral Development Model



The spiral model starts with an initial pass through a standard waterfall life cycle, using a <u>subset of the total requirements</u> to develop a robust prototype. The theory is that the set of requirements is hierarchical in nature, with additional functionality building on the first efforts.

Source: *The Incremental Commitment Spiral Model*, Barry Boehm and Jo Ann Lane, Addison Wesley, 2014

## Agile Development Model

- Agile development model is also a type of Incremental model. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications.

http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/

## Ways to Estimate Traditional "Waterfall Developed" SW Projects

- Available Methods
  - Analogy
  - Parametric
  - Engineering (Bottom Up)
  - Extrapolation of Actual Cost

- Analogy and parametric methods usually used during early phases
  - Analogy requires
    - Capability descriptions of projects
    - Final effort and durations
    - Monthly average full time equivalent personnel
    - Final software lines of code (SLOC) and defect data (optional)
  - Parametric requires size and scaling factors:
    - Effort = A × SizeB × C
    - Where
      - Effort is in person-months;
      - A is a calibrated constant;
      - B is a size scale factor;
      - C is an additional set of factors that influence effort; and
      - Size is in terms of SLOC, Function Points, Object-Oriented metrics or other

## Using Parametric Estimating Tools

- Need to know†
  - Targeted Operating Environment
  - Estimates of
    - New, reused, modified, generated Source Lines of Code (SLOC) that will be delivered
    - Percent of adopted software that must be re-designed
    - Percent of the reused and modified code that must modified to adapt it to the targeted objectives and environment
    - Requirements volatility
    - Team productivity attributes

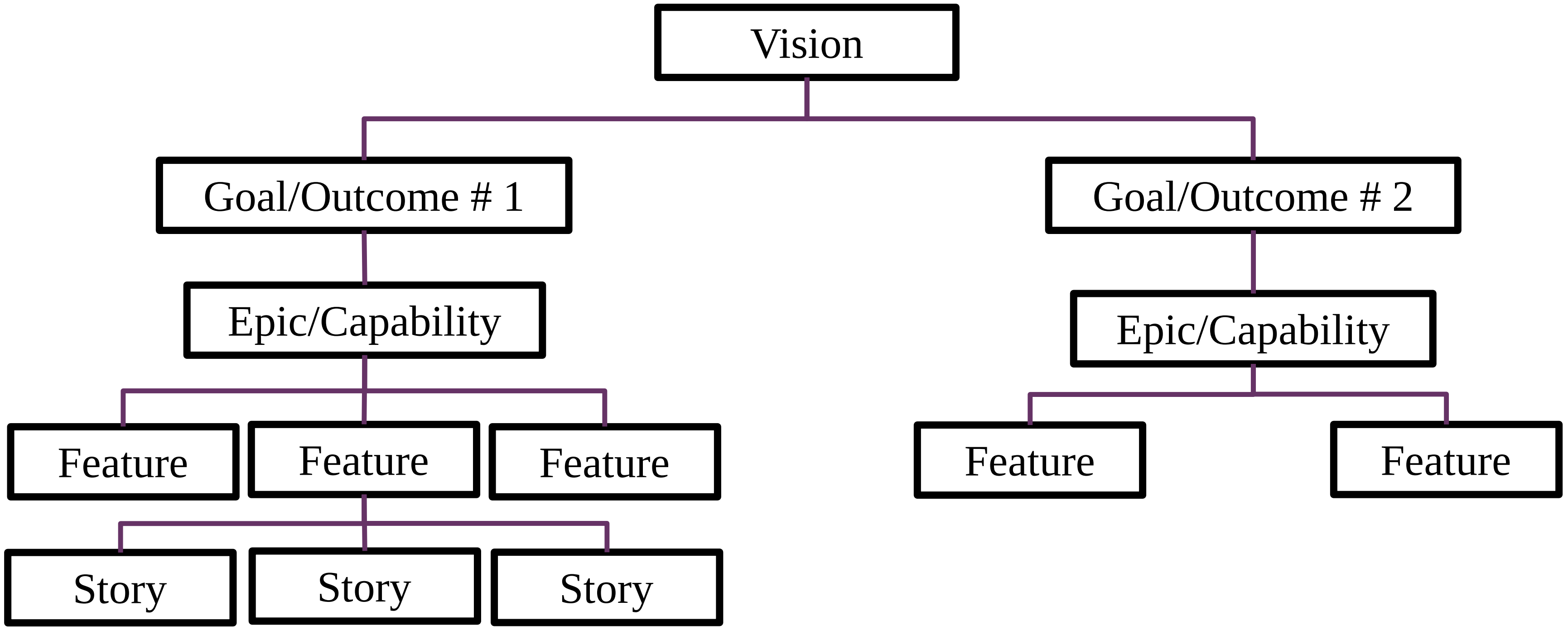
† Abstracted from "Software Cost Estimation Metrics Manual", Systems Engineering Research Center (SERC), 2016

## Estimating Challenge on Agile Projects

- The traditional parametric approach fails us because we do not have way of scaling the software application
  - *Is this system bigger than a bread box?*
  - No sizing possible since  requirements are not fully specified at the outset
  - All we have from the customer is a vision statement, desired capabilities and features, typically included in the customers Concept of Operations (ConOps) document

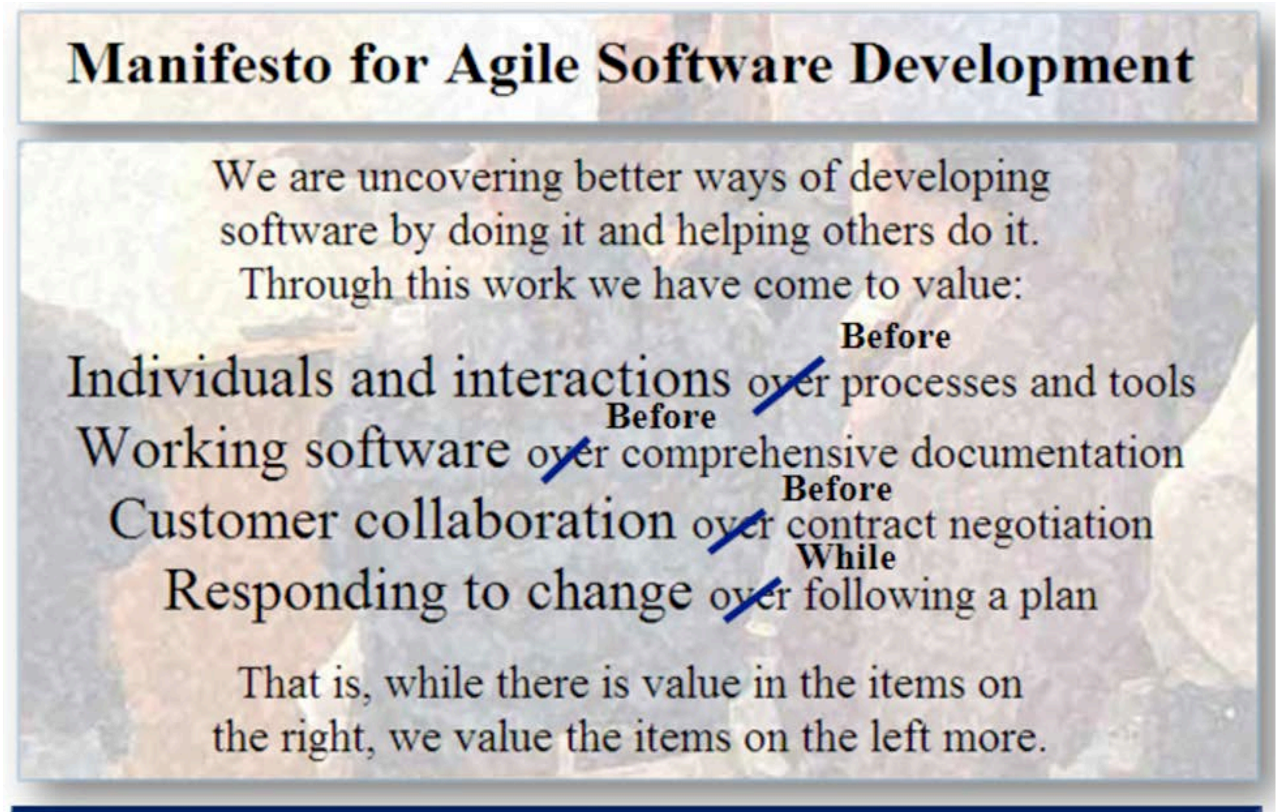
- Even though **specific capabilities and features** may not be finally delivered, we still must develop a credible estimate of cost and schedule for **those planned**

# Decomposing Vision into Capabilities and Features



Source: David Bulkin, Lithespeed.com

## + The Agile Manifesto does not fit well with FAR Part 15 Procurement



**Manifesto for Agile Software Development**

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions ~~over~~ **Before** processes and tools
Working software ~~over~~ **Before** comprehensive documentation
Customer collaboration ~~over~~ **Before** contract negotiation
Responding to change ~~over~~ **While** following a plan

That is, while there is value in the items on the right, we value the items on the left more.

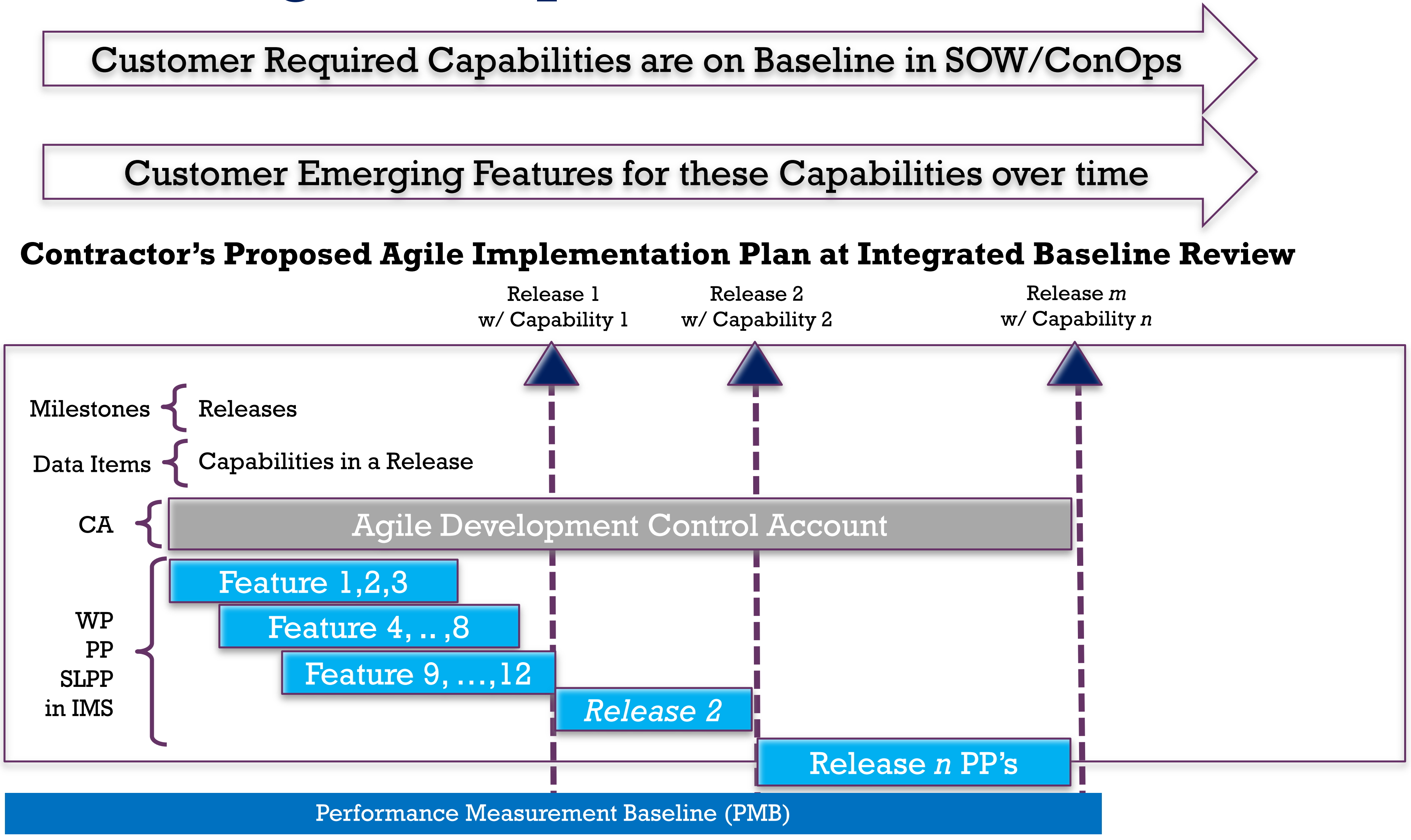
Nice for small self directed teams working on commercial projects with their customer in the same room

But we still need to deliver on the customer *Vision* with emerging requirements

## The 12 Principles of the Agile Manifesto

| | |
|---|---|
| 1. Customer satisfaction by early and continuous delivery of valuable software | 7. Working software is the principal measure of progress |
| 2. Welcome changing requirements, even in late development | 8. Sustainable development, able to maintain a constant pace |
| 3. Working software is delivered frequently (weeks rather than months) | 9. Continuous attention to technical excellence and good design |
| 4. Close, daily cooperation between business people and developers | 10. Simplicity—the art of maximizing the amount of work not done—is essential |
| 5. Projects are built around motivated individuals, who should be trusted | 11. Best architectures, requirements, and designs emerge from self-organizing teams |
| 6. Face-to-face conversation is the best form of communication (co-location) | 12. Regularly, the team reflects on how to become more effective, and adjusts accordingly |

# DoD Agile Acquisition Process



Customer Required Capabilities are on Baseline in SOW/ConOps
Customer Emerging Features for these Capabilities over time
Contractor's Proposed Agile Implementation Plan at Integrated Baseline Review
Release 1 w/ Capability 1
Release 2 w/ Capability 2
Release m w/ Capability n
Milestones
Releases
Data Items
Capabilities in a Release
CA
Agile Development Control Account
WP
PP
SLPP
in IMS
Feature 1,2,3
Feature 4, .. ,8
Feature 9, …,12
Release 2
Release n PP's
Performance Measurement Baseline (PMB)

## Cost Drivers of Agile Developed Software

- Vision, Capabilities, and Features that implement that Capabilities
- Same attributes needed using parametric tools
  - Operating Environment, e.g. Unmanned Aircraft, Sea Systems, Space Systems, etc.
  - Application Domain, e.g., Communication, $C^2$, Enterprise Information System, etc.
  - Productivity Factors
    - Team experience
    - Extent use of existing code (common libraries or existing code)
    - Extent use of automated tools

# Proposal: Estimate Agile Software Projects Using the Nearest Neighbor Analogy Technique

- For a proposed future agile application for a given operating environment and application domain and stated features, the estimator should be able to query on a standardize feature list and obtain:
  - Actual staff hours to produce the feature
  - Actual duration to produce the feature
  - Extent of software reuse and sources
  - Extent of use of automated tools
  - Team experience
- For each feature, estimator would array historical dependent effort and durations and other attributes and compare to targeted feature
- For proposed new feature with given planned library and automated tool use, find nearest neighbor with similar reuse and tool use and extract effort and duration
- If no knowledge of reuse and automated tool use, use the means

## Nearest Neighbor Notional Example for Autonomous UAV Flight Control Feature

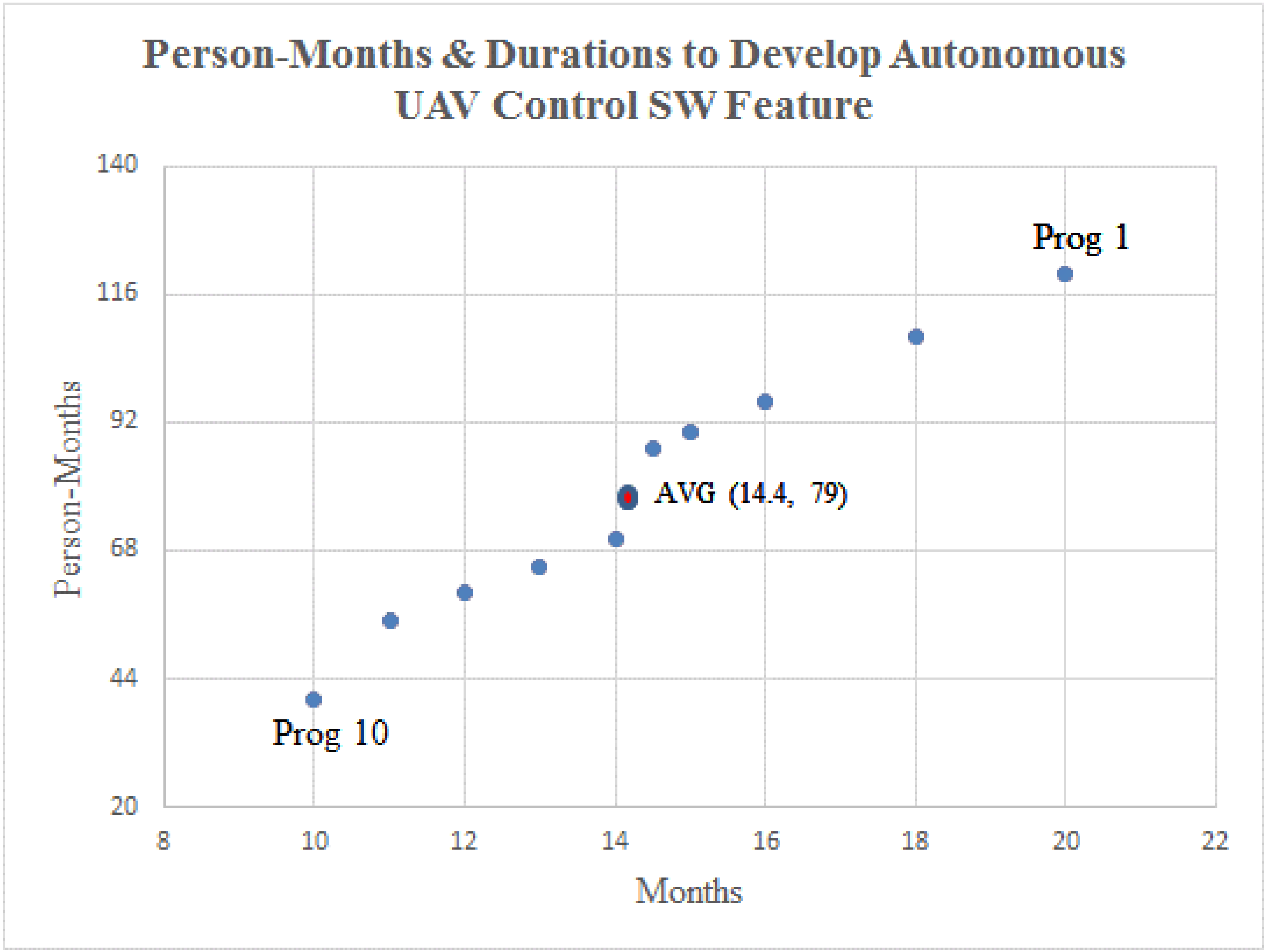
| Prog ID | Feature | DevProc | Lib Use | Tool Use | Durr | PM |
|---|---|---|---|---|---|---|
| Prog 1 | Auto Exc Pilot Mission | Agile | None | None | 20 | 120 |
| Prog 2 | Auto Exc Pilot Mission | Agile | Min | Min | 18 | 108 |
| Prog 3 | Auto Exc Pilot Mission | Agile | Min | Mod | 16 | 96 |
| Prog 4 | Auto Exc Pilot Mission | Agile | Min | Heavy | 15 | 90 |
| Prog 5 | Auto Exc Pilot Mission | Agile | Mod | Min | 14.5 | 87 |
| Prog 6 | Auto Exc Pilot Mission | Agile | Mod | Mod | 14 | 70 |
| Prog 7 | Auto Exc Pilot Mission | Agile | Mod | Heavy | 13 | 65 |
| Prog 8 | Auto Exc Pilot Mission | Agile | Heavy | Min | 12 | 60 |
| Prog 9 | Auto Exc Pilot Mission | Agile | Heavy | Mod | 11 | 55 |
| Prog 10 | Auto Exc Pilot Mission | Agile | Heavy | Heavy | 10 | 40 |

| | | |
|---|---|---|
| Mean | 14.4 | 79 |
| Std | 3.1 | 25.2 |

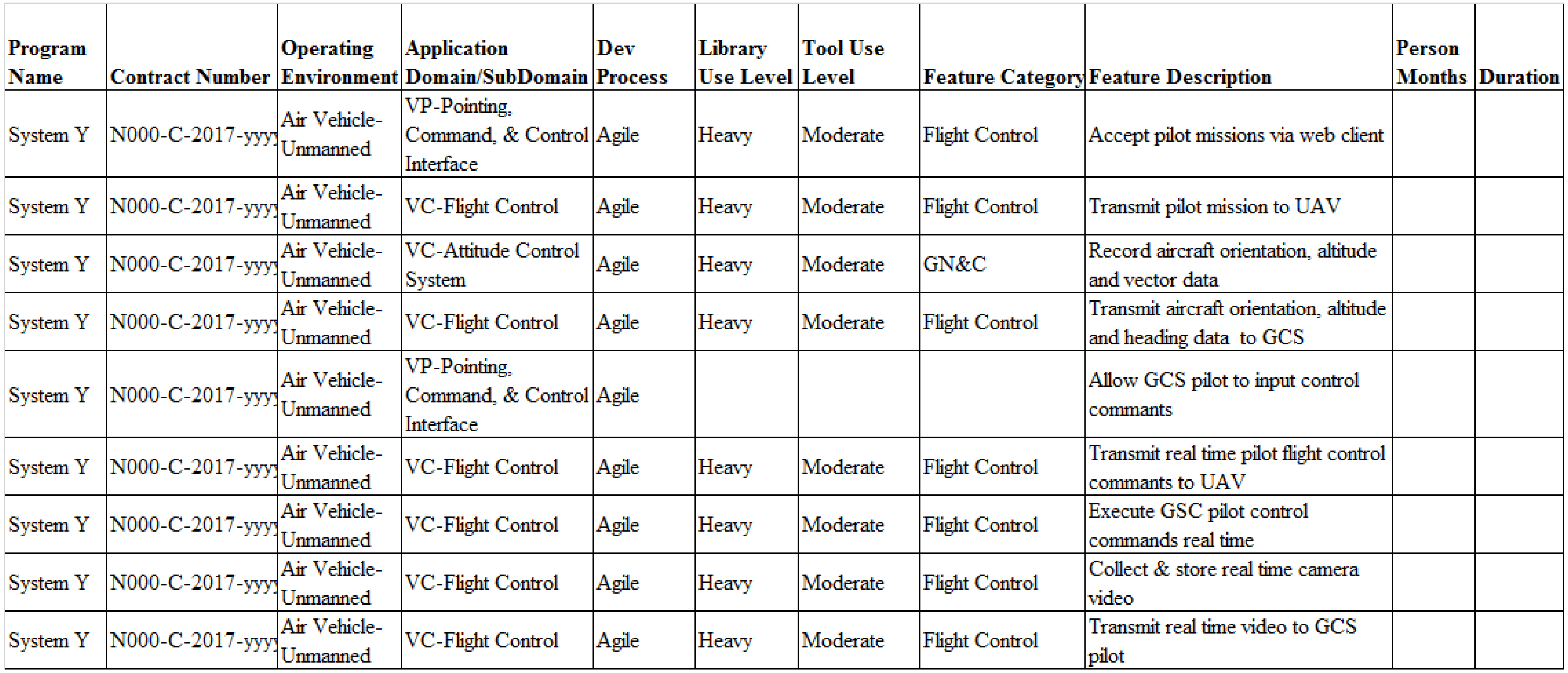
To estimate the effort and duration of a future UAV Autonomous Flight Execution Feature, select data point nearest attributes of future system.
If no knowledge of future attributes, select means

Note: Historical Data was pre-filtered for Operating Environment and Application Domain/Subdomain

## Nearest Neighbor UAV Flight Control Feature



# Extract of Notional Feature Repository



| Program Name | Contract Number | Operating Environment | Application Domain/SubDomain | Dev Process | Library Use Level | Tool Use Level | Feature Category | Feature Description | Person Months | Duration |
|---|---|---|---|---|---|---|---|---|---|---|
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VP-Pointing, Command, & Control Interface | Agile | Heavy | Moderate | Flight Control | Accept pilot missions via web client | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Transmit pilot mission to UAV | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Attitude Control System | Agile | Heavy | Moderate | GN&C | Record aircraft orientation, altitude and vector data | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Transmit aircraft orientation, altitude and heading data to GCS | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VP-Pointing, Command, & Control Interface | Agile | | | | Allow GCS pilot to input control commants | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Transmit real time pilot flight control commants to UAV | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Execute GSC pilot control commands real time | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Collect & store real time camera video | | |
| System Y | N000-C-2017-yyy | Air Vehicle-Unmanned | VC-Flight Control | Agile | Heavy | Moderate | Flight Control | Transmit real time video to GCS pilot | | |

# Data Needed to Use the Nearest Neighbor Technique

- **Feature Description**
- Operating Environment, e.g. Air Vehicle, Space Systems, Ordnance Systems, etc.
- Application Domain, e.g., Vehicle Payload, Vehicle Control, Command and Control, Enterprise Information System
- **Extent of software reuse (use of common SW Libraries)**
- **Extent use of Automated Tools**
- **Feature Category**
- Actual staff hours to produce the feature
- Actual duration to produce the feature

## Source of Feature Data

- OSD CAPE has proposed revisions to Software Resource Data Report (SRDR) to collect software data that are developed using the Agile model
  - Most of the data needed is contained in the SRDR
  - Only items bolded in black on previous page appear to be missing **(Action)**
- Feature Category would need to be added after the community created a Feature Breakdown Structure from the submitted data **(Future Action)**

## Some Conclusions

- Agile-developed software estimating is challenging because we don't know all the requirements up front so we can't generate traditional size measures and use parametric tools
- For Agile efforts, we only know the customer's vision, desired capabilities and stated features
- Analogy-based nearest neighbor estimating approach is a feasible technique if we have requisite data from historical projects
- This presentation explains and demonstrates this estimating approach, identifies the data needed and offers suggestions to SRDR to make this estimating approach possible

## Wrap Up

Let’s not get all wrapped up in the notion of Agile Software Development

We still have to apply the core principles of cost estimating in the presence of uncertainty

Agile let’s us focus on modeling the cost of the Capabilities and Features

The Result is a cost model from the *Capabilities Breakdown Structure* for the Feature Costs

## Questions???

# Backup

## Software Resource Data Report (SRDR) Background

- The SRDR is DoD's record of contractor or government entities' planned and actual software resources embedded within Major Defense Acquisition Programs (MDAPs) or Major Automated Information Systems (MAIS)

- Applicability
  - The SRDR is required on contracts and subcontracts regardless of contract type valued in excess of $20 Million
  - Reported resources are required on individual Work Breakdown Structure (WBS) elements (or group of WBS elements) within Major Defense Acquisition Programs (MDAPs) and Major Automated Information System (MAIS) programs as specified in approved Cost and Software Resource Data Report (CSDR) Plans
  - Initial, Interim and Final Reports for each SW Release is required throughout the complete life cycle to include the Operating and Support (O&S) phase of a program

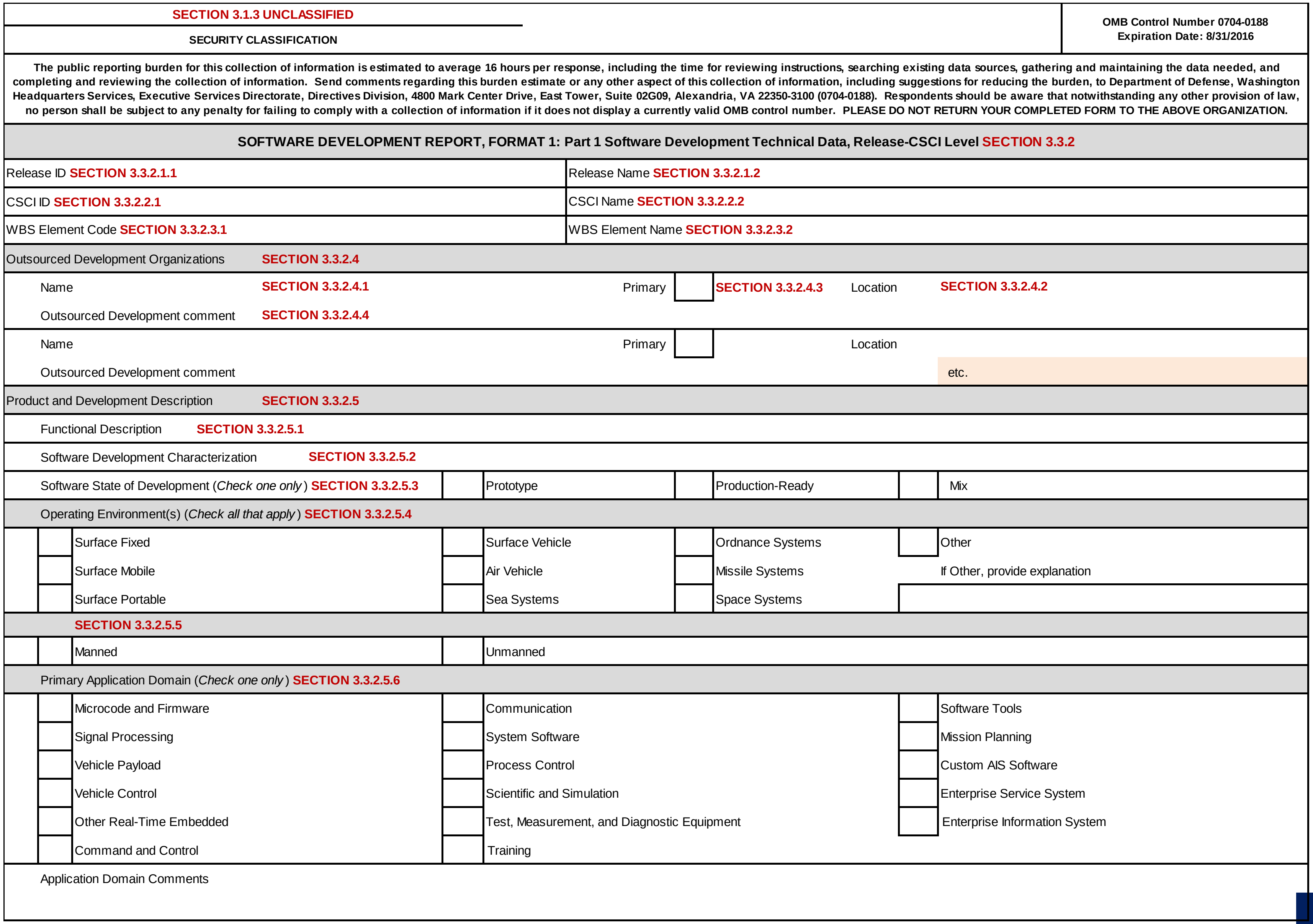
- Is used to
  - Build credible size, cost, and schedule estimates of future software-intensive systems
  - Support Analysis of Alternatives
  - Perform Cost Research
  - Assist in Program Progress Reviews

# SRDR (Concluded)

- Report comes in three formats
  - DD Form 3026-1 – Software Development Report
    - Part 1: Planned and actual software development size, context, and technical information, e.g. defects, at the Release and CSCI level of detail
    - Part 2: Actual hours and dollars expended by month for each Release and CSCI
  - DD Form 3026-2 – Software Maintenance Report
    - Part 1: Actual software development size, context, and technical information at the Release level
    - Part 2: Actual hours for each WBS element within a release by ISO 12207 SW maintenance activity categories
  - DD Form 3026-3 – Enterprise Resource Planning (ERP) Software Development Report
    - Part 1: Planned and actual software development size, context, and technical information for a release, including planned and actual agile metrics, e.g. epics, features, stories and story points*
    - Part 2: Actual hours per release to plan & analyze, design/build, test, deploy support the system, and provide other direct labor, e.g. PM, ST&E and data

* A "Release" in an agile project is a time-box covering a fixed calendar period (for example 90 days), and not a specifically designed software end item.

# The New SRDR Provides Some of the Data Needed



| | |
|---|---|
| SECTION 3.1.3 UNCLASSIFIED<br>SECURITY CLASSIFICATION | OMB Control Number 0704-0188<br>Expiration Date: 8/31/2016 |

The public reporting burden for this collection of information is estimated to average 16 hours per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Executive Services Directorate, Directives Division, 4800 Mark Center Drive, East Tower, Suite 02G09, Alexandria, VA 22350-3100 (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR COMPLETED FORM TO THE ABOVE ORGANIZATION.

**SOFTWARE DEVELOPMENT REPORT, FORMAT 1: Part 1 Software Development Technical Data, Release-CSCI Level SECTION 3.3.2**

| | |
|---|---|
| Release ID **SECTION 3.3.2.1.1** | Release Name **SECTION 3.3.2.1.2** |
| CSCI ID **SECTION 3.3.2.2.1** | CSCI Name **SECTION 3.3.2.2.2** |
| WBS Element Code **SECTION 3.3.2.3.1** | WBS Element Name **SECTION 3.3.2.3.2** |

Outsourced Development Organizations **SECTION 3.3.2.4**

| | | | | | | |
|---|---|---|---|---|---|---|
| Name | **SECTION 3.3.2.4.1** | Primary | ☐ | **SECTION 3.3.2.4.3** | Location | **SECTION 3.3.2.4.2** |
| Outsourced Development comment | **SECTION 3.3.2.4.4** | | | | | |
| Name | | Primary | ☐ | | Location | |
| Outsourced Development comment | | | | | | etc. |

Product and Development Description **SECTION 3.3.2.5**

Functional Description **SECTION 3.3.2.5.1**

Software Development Characterization **SECTION 3.3.2.5.2**

| | | | | | | |
|---|---|---|---|---|---|---|
| Software State of Development (*Check one only*) **SECTION 3.3.2.5.3** | ☐ | Prototype | ☐ | Production-Ready | ☐ | Mix |

Operating Environment(s) (*Check all that apply*) **SECTION 3.3.2.5.4**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ☐ | Surface Fixed | ☐ | Surface Vehicle | ☐ | Ordnance Systems | ☐ | Other |
| ☐ | Surface Mobile | ☐ | Air Vehicle | ☐ | Missile Systems | | If Other, provide explanation |
| ☐ | Surface Portable | ☐ | Sea Systems | ☐ | Space Systems | | |

**SECTION 3.3.2.5.5**

| | | | |
|---|---|---|---|
| ☐ | Manned | ☐ | Unmanned |

Primary Application Domain (*Check one only*) **SECTION 3.3.2.5.6**

| | | | | | |
|---|---|---|---|---|---|
| ☐ | Microcode and Firmware | ☐ | Communication | ☐ | Software Tools |
| ☐ | Signal Processing | ☐ | System Software | ☐ | Mission Planning |
| ☐ | Vehicle Payload | ☐ | Process Control | ☐ | Custom AIS Software |
| ☐ | Vehicle Control | ☐ | Scientific and Simulation | ☐ | Enterprise Service System |
| ☐ | Other Real-Time Embedded | ☐ | Test, Measurement, and Diagnostic Equipment | ☐ | Enterprise Information System |
| ☐ | Command and Control | ☐ | Training | | |

Application Domain Comments

DD Form 3026-1, Part 1, contains required operating environments and application domains

# SRDR Planned Agile Data Elements

**ENTERPRISE RESOURCE PLANNING SOFTWARE RESOURCES DATA REPORTING, FORMAT 3:**
**PART 1: Software Development Technical Data (Other Sizing) SECTION 3.3**

**D.2 - Alternative Product Size Reporting 3.3.5**

| | | |
|---|---|---|
| **35. Agile Measures: 3.3.5.3** | **Days per Release:** | **Days per Sprint:** |

| Release Map 3.3.5.3.1 | | Planned and Achieved Development (by Feature per Epic) 3.3.5.3.2 | | | | | |
|---|---|---|---|---|---|---|---|
| Epic/ Capability Identifier | Feature Identifier | Planned Stories per Feature by Epic | Actual Stories | Planned Story Points per Feature by Epic | Actual Story Points | Planned Hours per Feature by Epic | Actual Feature Hours |
| *ex: AAAA* | *1.x.x* | *5* | *3* | *15* | *10* | *200* | *180* |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

(NOTE: Insert rows as needed to account for all Features and Epics mapped to this Release.)

**Summary Totals for This Release 3.3.5.3.3**
**(Sum of all rows with data by Feature by Epic)**

| Item | Quantity Planned | Quantity Actually Developed |
|---|---|---|
| Total Features | | |
| Total Epics/ Capabilities | | |
| Total Stories | | |
| Total Story Points | | |
| Total Feature Hours | | |
| Total Sprints | | |

## Observations on SRDR Planned Agile Metrics

- In Agile development Capabilities are baseline, Requirements in agile development *emerge* as the project progresses.

- Collecting story points does not seem to help the estimator
  - Story points represent the relative (UN–calibrated) effort involved to deliver a Product Backlog Item, when it is selected for development
  - Story points are not scope, cost or duration
  - Story points counting is not standard between development teams much less between development entities
  - Story points are not known before the contract begins