A Primer on Agile Software Development for Cost Analysts

Presented to:

2016 ICEAA National Conference

Qualis Corporation:
Your Partner in Success!

Qualis Corporation ♦ 689 Discovery Drive ♦ Huntsville, AL 35806 ♦ 256-971-1707

Purpose

- To explain “the next big thing” in software development practices, Agile development
- To discuss why many people think Agile is the panacea for software cost/schedule growth woes, and then balance that dream with a discussion on the many implementation issues

Overview

Pro-Agile

- What is Agile software development and how does it differs from traditional methods?
- What are the promises of Agile with respect to cost, schedule, performance ?

Anti-Agile

- Agile implementation issues

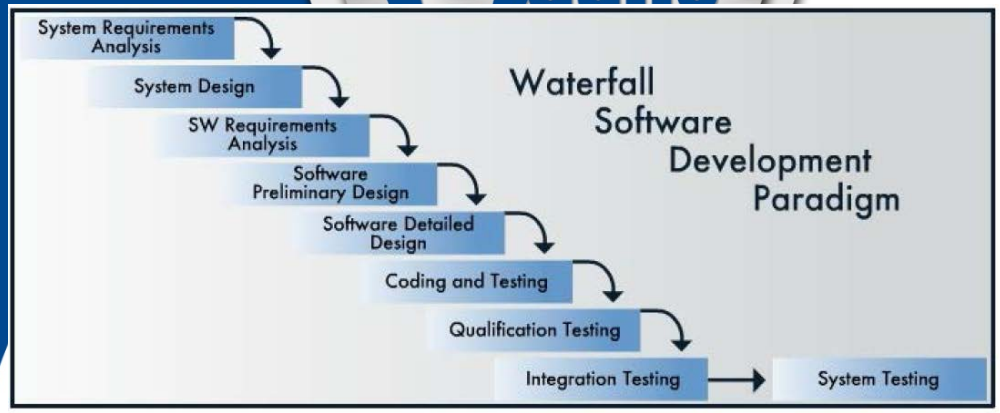
Agile Agnostic

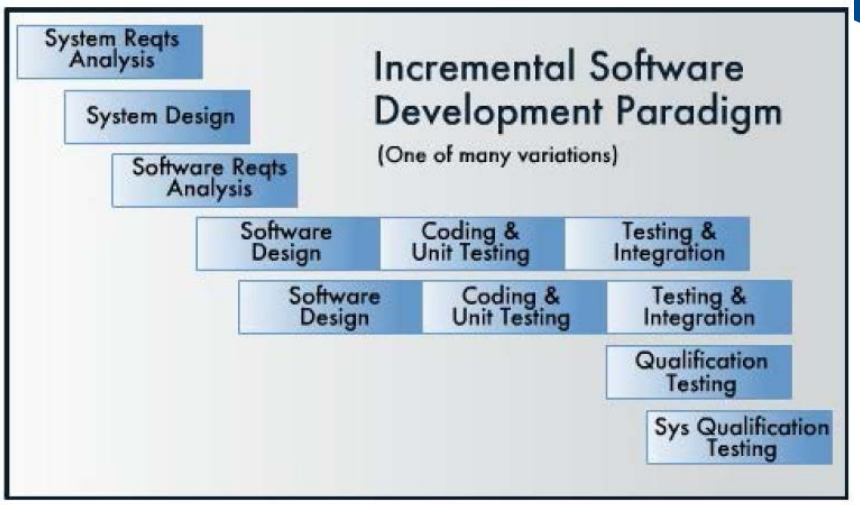
- Cost Estimating methods for Agile

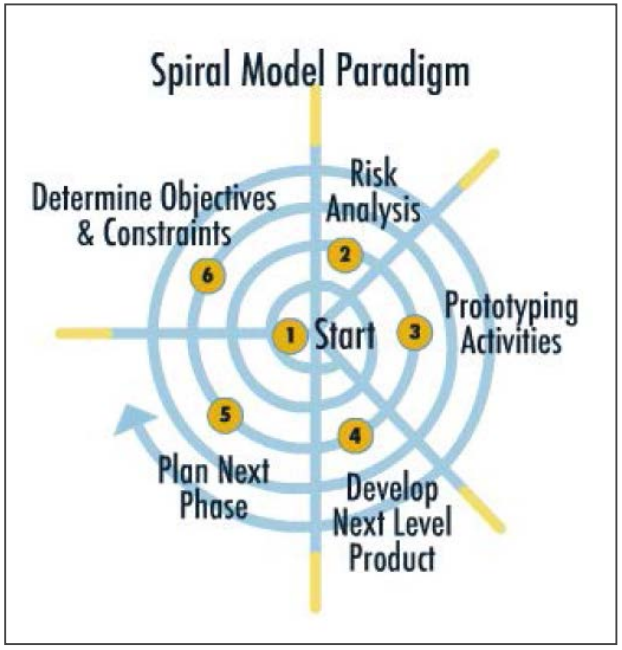
A Brief History of Agile

- **In Feb 2001, 17 software developers met at the Snowbird resort in Utah to discuss 'lightweight' s/w dev methods**
  - They came from variety of s/w backgrounds (NASA, Chrysler, et al) and all believed in alternate coding approaches (e.g. eXtreme Programming, SRIM, DSDM, Adaptive Software Dev, and Crystal)
  - Goal was to uncover better ways of developing software after witnesses multiple projects fail due to schedule delays
  - The "Agile Alliance" published an "Agile Manifesto", a framework for doing what we now call Agile s/w dev at the end of the 2-day meeting
- **In 2005, Alistair Cockburn and Jim Highsmith wrote a guide for software project management according to agile methods**
- **In 2009, Robert Martin wrote 'The Software Craftsman Manifesto', further elaborating on what agile dev is**
- **Currently, there are over 100,000 "Certified ScrumMasters" and probably a million trying to do Agile**

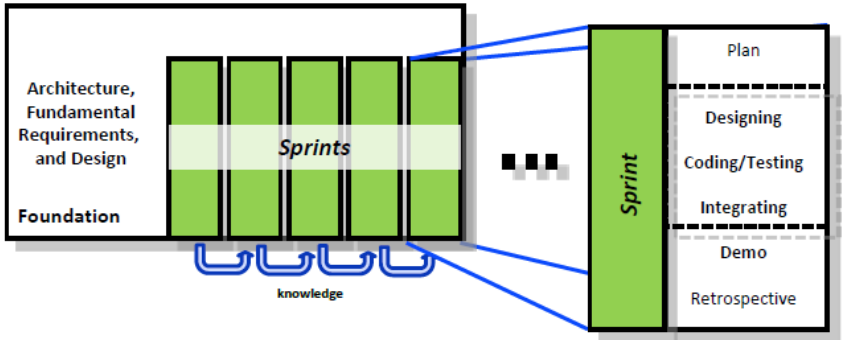
Four Common Software Development Methods





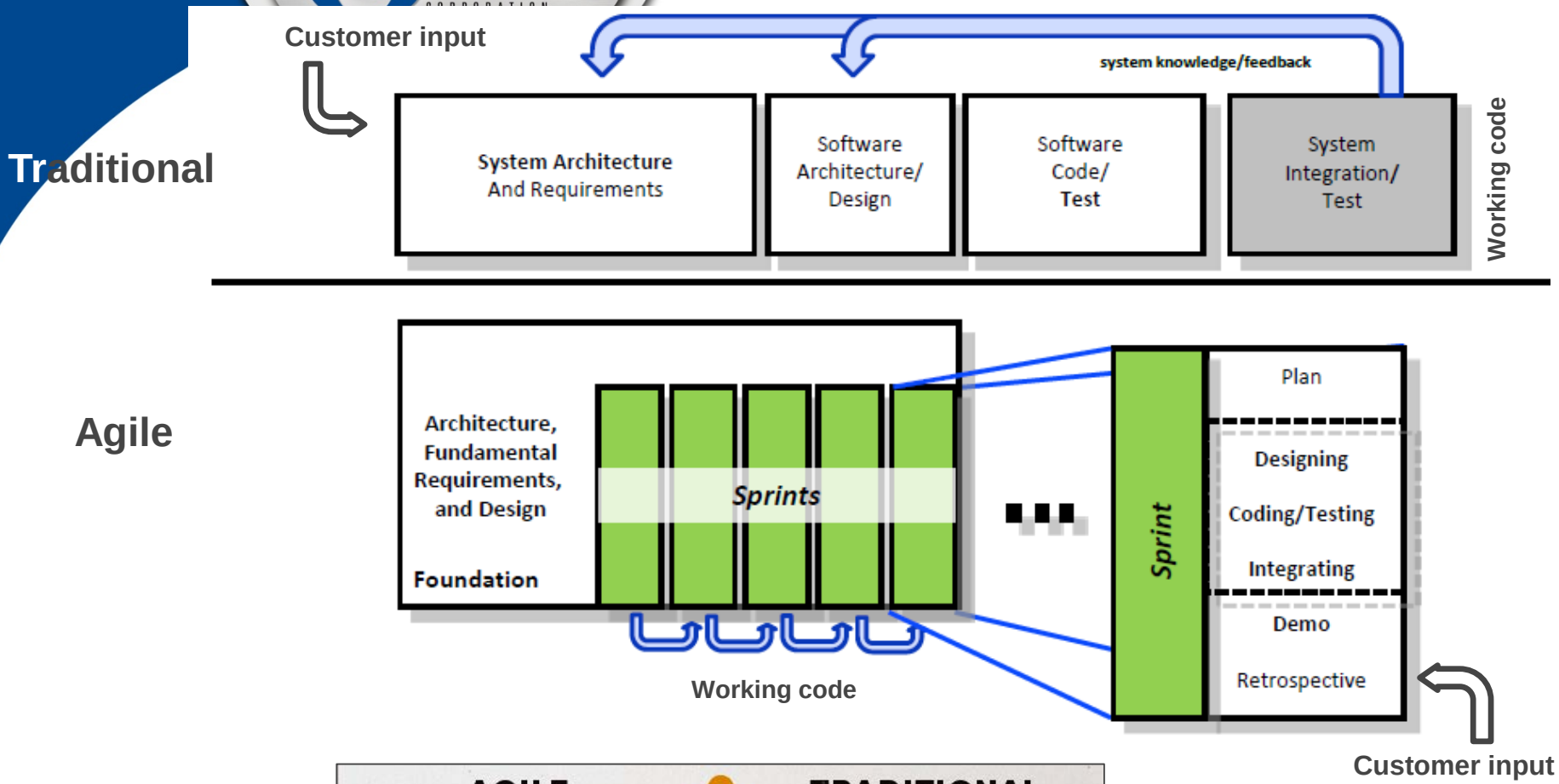


Agile (Scrum)

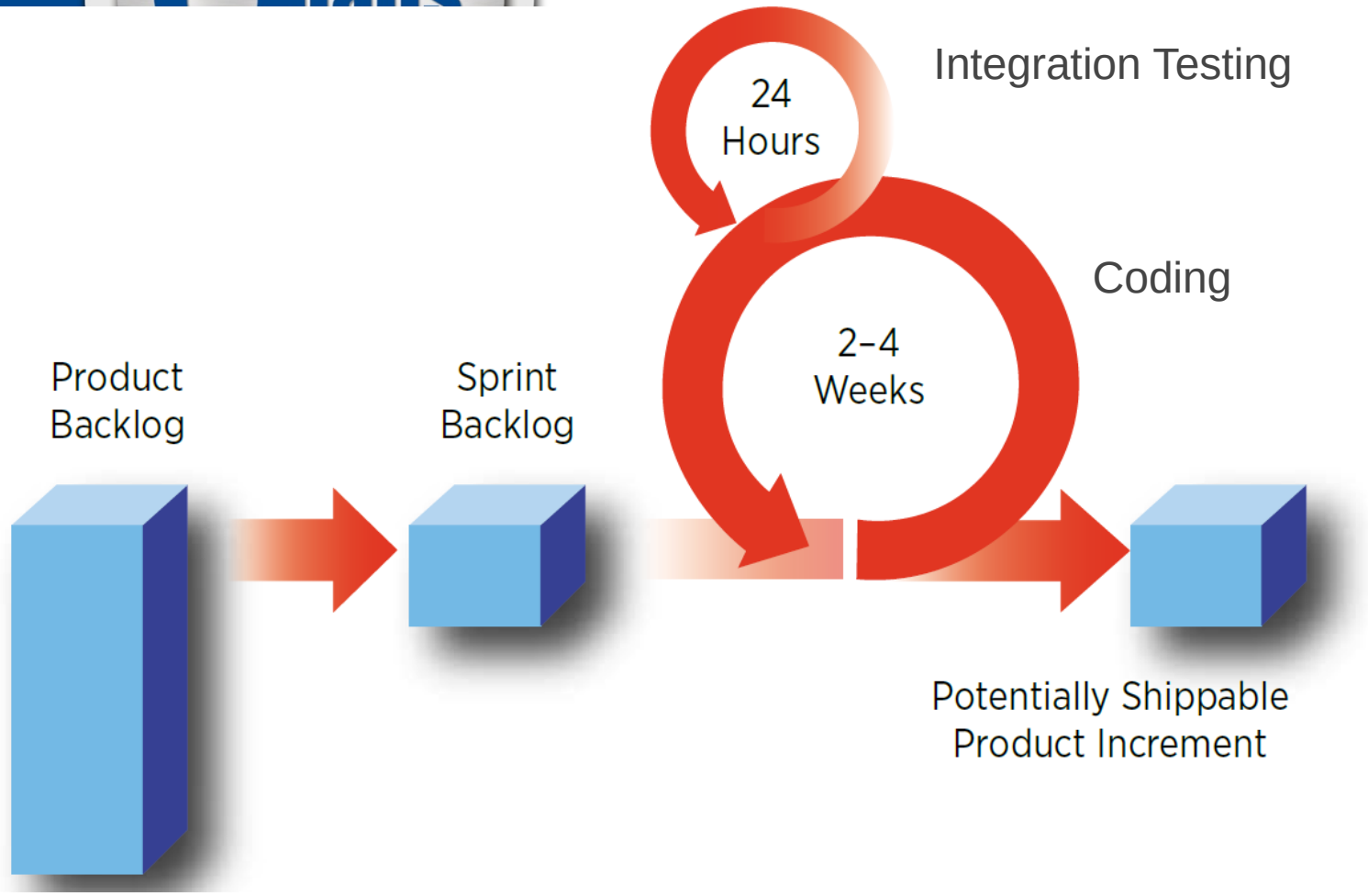


Note: Graphics taken from DAU website

Traditional and Agile



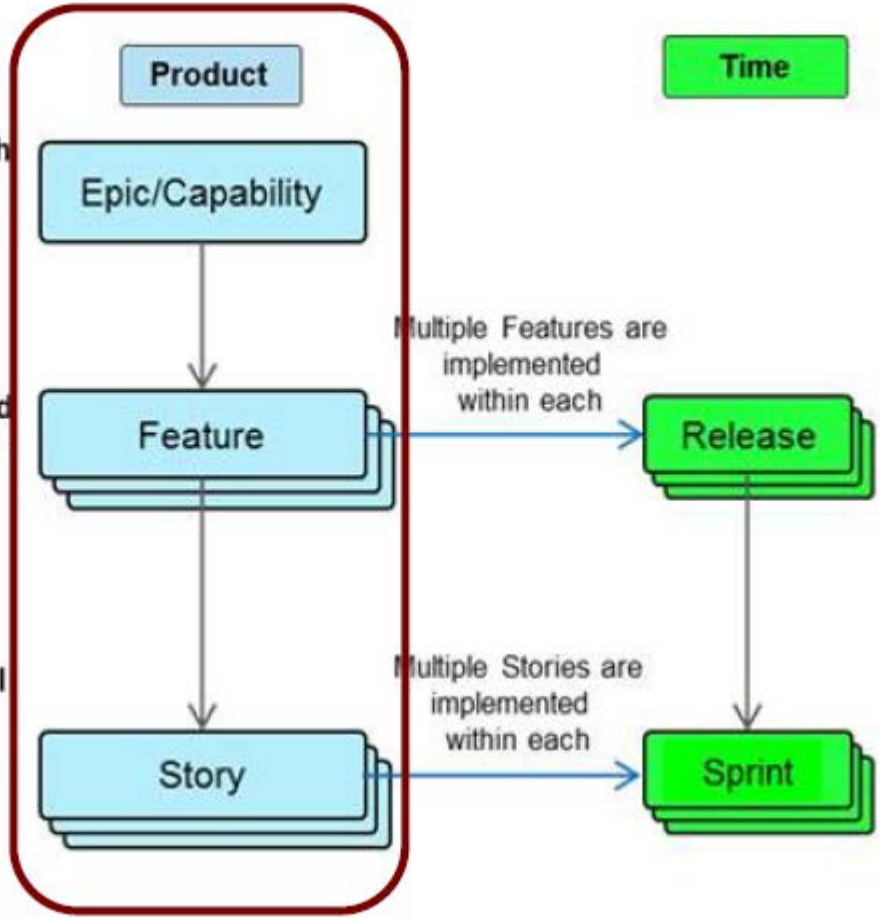
Agile Sprint Process



Note. Adapted from *The SCRUM process/SCRUM framework* [Web page], by Expert Program Management (n.d.) at http://www.expertprogrammanagement.com/2010/08/the-scrum-process/.

Agile Products

- Epic/Capability: A high level major system function typically completed across multiple releases.
- Feature: A well defined system function to be completed within a release.
- Story: A small but well defined system function that can be developed within one iteration.



- Cadence Release: A fixed time cadence for development (not to be confused with a product delivery milestone); nominally 2-4 months, comprising a fixed number of complete iterations.
- Iteration: Recurring, non overlapping, cadence for development, 1-6 weeks in duration.

Agile uses a product-oriented WBS

Uniquely Agile

Agile Values		
'Individuals and Interactions'	Over	'Processes and Tools'
'Working Software'	Over	'Comprehensive Documentation'
'Customer Collaboration'	Over	'Contract Negotiation'
'Responding to Change'	Over	'Following a Plan'

Agile Practices

Incremental Development	Small Teams
Iterative Development	Time Boxing
Short Time-lines	Lean Initiatives
Retrospectives (Lessons Learned)	Prototyping
Empowered/Self-organizing/ Managing Teams	Continuous User Involvement
Prioritized Product Backlog (Requirements)	Co-located Teams

Impact of Going Agile

- Very little qualitative data exists to support business case for implementing Agile
  - Most of the impetus for change is coming from Industry who recognizes that this is a better way to do s/w development and is asking govt to adapt their polices and reqts (especially EVM)

- Intuitively, Agile should:
  - Reduce integration cost due to continuous testing
  - Reduce incidents of massive cost growth as working code is delivered sooner (scope is traded instead of cost or schedule)
  - Reduce sustainment cost as defect rate is decreased (quality code)
  - Reduce code re-work as customers are integrated into teams
  - Customers validate value of new features in near real-time and influence code direction

- Rally Software Development company study agile implementation across 50,000 Agile teams and estimated:
  - Responsiveness: Agile processes cut time to market in half
  - Quality: Teams doing Full Scrum have 250% better quality
  - Productivity: Stable Agile teams are twice as productive

Implementation Complications

- The transfer of this approach to DoD acquisition strategies has not been straight-forward
  - There is no DoD-level guidance on how to implement Agile
  - In some cases, industry is pushing agile dev methods to govt
  - In other cases, govt is pushing agile dev methods to industry

- Common agile pitfalls experienced in industry include:
  - Lack of overall project design
  - Adding stores or sprints indefinitely
  - Insufficient training (Agile not taught in universities)

- Changing methods in DoD requires potentially writing new Acquisition Law (DoD 5000 series), changing OSD/Services Acquisition Policy, and/or changing values/culture

- Eventually, Agile could follow EVMs path and become a standard fixture in DoD acquisition with a set of common tools/policy/data/vocabulary …. Not there yet

DoD Implementation Issues

- **Can Agile address the complexity of DoD Systems?**
  - Can we decompose tightly-coupled technical requirements into Agile user stories and controlled interfaces?
  - Can we identify authoritative customers – among many diverse stakeholders, including the Adversary – for feedback and iteration?
  - Can we learn from small, agile teams and scale to complex projects?
  - Can we support formal, independent testing over long test cycles?

- **Can Agile address regulatory challenges?**
  - Can we provide enough up-front cost, schedule, and risk analysis to satisfy DoD regulatory and statutory requirements?
  - Can we support the persistent oversight and management requirements of DoD acquisitions?
  - Can we mix contractual negotiation with customer collaboration?

Implementation Issues by Functional

Contracting Issues

- How do evaluate proposals when one company uses Agile and the other doesn't?
- How do you understand and control change, preventing unauthorized changes to scope? What constitutes a contract change?
- Do we create RFP (CDRL, SOW/SOO) template for Agile?

Program Management

- Culture change with customer (Warfighter) working directly with industry and making changes to requirements on the fly.
- Need a Agile Integrated Program Management (IPM) guide and implementation training for everyone in Program Office (Institutionalize common understanding of Agile in all functional areas)

Implementation Issues by Functional

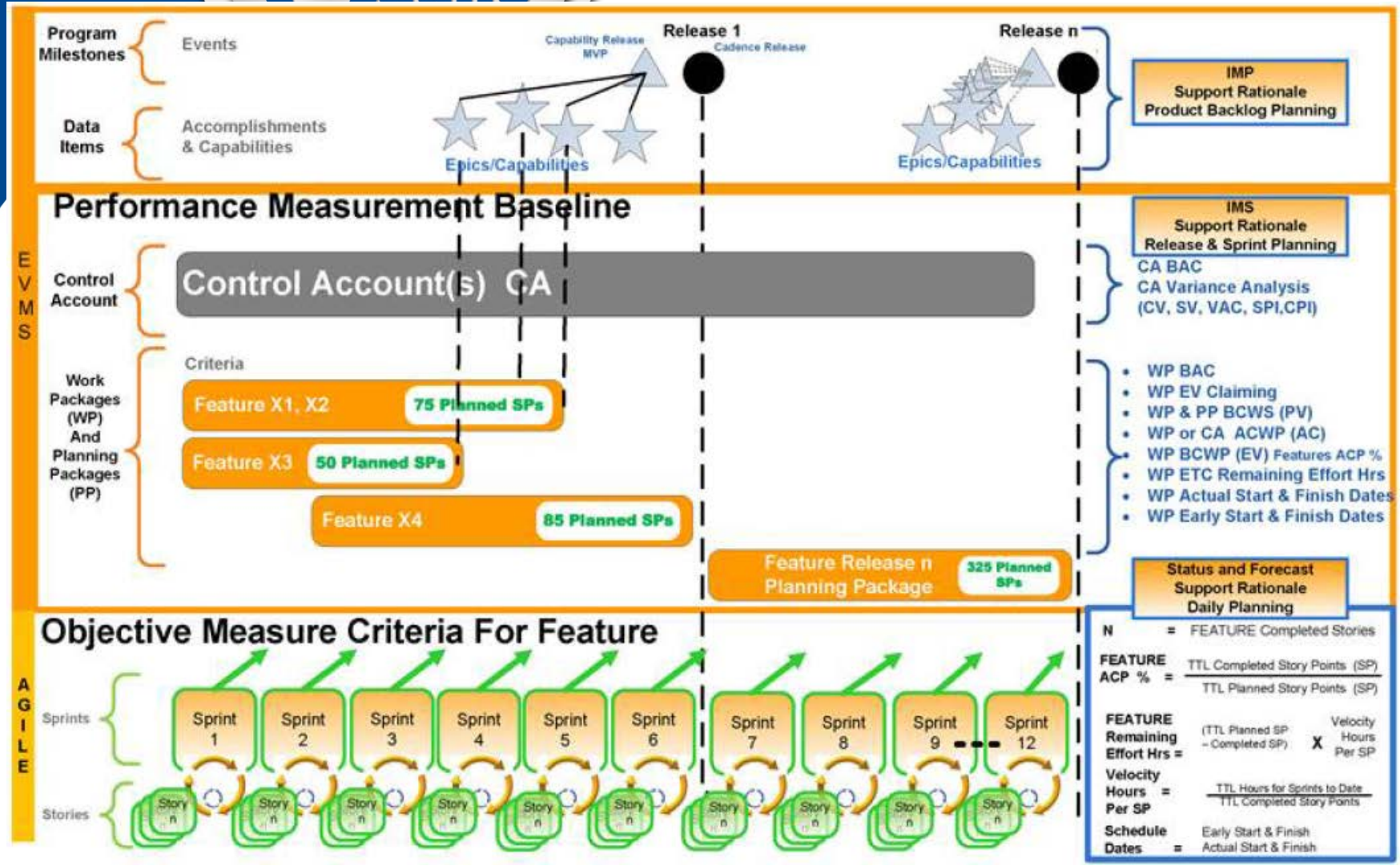
System Engineering

- Defining “just enough reqts” for design
- What do design reviews (PDR, CDR,…) look like in Agile?
- How do we do baseline management?
- We still need some documentation, right?

Earned Value Management

- No fixed scope to measure baseline against
- Common Agile framework needs to be developed within construct of EVMS
- How does the contractor’s processes/documents integrate Agile and EVMS into contract execution?
- What products within the contractor Agile execution can be used to support demonstration of EVMS compliance?

EVM and the Agile Framework



Cost Estimating Issues

Open questions for cost community:

- Do we develop a standard WBS for Agile programs?
- How do we evaluate proposals where one is Agile and one is not?
- Are existing cost estimating metrics and models going to work with Agile?
- How valid are historical software databases and productivity rates in an Agile environment?
- Is there a cost for implementation or a learning curve for the Program Office?
- Do we move from SLOC-based cost estimating to ‘Story Point / Velocity’ method?
- Do we need to modify cost reporting CDRLs for Agile?
- How do we estimate the layers of management on top of the development team?

Estimating via Story Points and Velocity

- Story points are a way to size the s/w
- The size of the story includes design, unit code, testing, and integration
- Velocity = Story points complete per Sprint

Notional Example:

Sprint Team size x Sprint Length x Total Story points / Velocity = Effort

10 FTE/Sprint x 3 Weeks x 120 SP / 20 SP/Sprint = 180 man-weeks = 7,200 hrs

- This method is not SLOC based
  - Typically FTE per Sprint is in the 7 to 9 range
  - Timebox for each sprint is typically 2-3 weeks, can be 3 months
  - Velocity largely depends on type of software developing

Conclusion and Path Forward

- Agile is the “next big thing” in software development

- The cost community needs to adapt to the changing environment by:
  - Developing a Standard ‘Agile’ WBS
  - Collect cost data on completed Agile programs using DoD-wide data call
  - Using this data, compare predictive power of different estimating methods using standard WBS
  - Update DIDs to include Agile metrics if Velocity methods prove more robust then SLOC-based methods

Questions?

Agile Terms

- **User Stories**: A description of a user-oriented capability containing just enough information so that developers can have a reasonable understanding of how they would need to implement
- **Complexity Points**: A relative measure of complexity that enables the team to quantify the scope or effort for an issue/artifact (e.g. story)
- **Sprint**: Fixed time-box in which development occurs (usually 2 - 4 weeks)
- **Iteration**: Collection of sprints that result in prototype software demoed/tested with users
- **Release**: Multiple iterations that fulfill a major subset of requirements constituting an operational product released to the users
- **Velocity**: Performance / productivity measure that indicates progress toward capability delivery (i.e., Complexity Points completed per sprint)
- **Project / Sprint Backlog**: A prioritized database that summarizes the issues/artifacts yet to be completed for a sprint or the entire project
- **Burndown**: The concept, often shown as a graph over time, of working off or “earning” Complexity Points toward iteration or delivery completion