Lessons Learned Applying EVMS on Agile Programs

Alexandria, VA
February 19, 2015

Ron Terbush  LM EVM SME
ronald.j.terbush@lmco.com
301-240-6573

Rob Eisenberg  LM Fellow
robert.j.eisenberg@lmco.com
301-240-5510

@2015 LOCKHEED MARTIN CORPORATION. ALL RIGHTS RESERVED
Agile Terminology

- Epic/Capability: A high level major system function.

- 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.

- Release: Release content has clear goals and objectives and occurs on a regular cadence (not to be confused with a program milestone).

- Iterations: Recurring, non overlapping, cadence for development (nominally 2 – 3 weeks).

“SPRINT” is overtaking “iteration” at LM
The WBS organizes the project deliverables into product based manageable units of work. The Agile WBS will nominally follow one of two basic structures, referred to here as “release centric” or “capability centric”. Which variation is employed is primarily driven by how the customer views the product to be delivered.

**RELEASE CENTRIC**
The customer views the product in terms of release. An example of this might be a large satellite ground system where the releases are based around major system events such as launch support, initial calibration, initial operations, and full system operations.

**CAPABILITY CENTRIC**
The customer views the product in terms of a set of discrete capabilities, where the releases are primarily viewed as time boxes for the ongoing and sustained delivery of Features. The release content may change greatly over time based upon changing priorities.
Work Breakdown Structure

- **Challenges**
  - Customer requires MIL-STD-881C WBS
  - Definition of “Release” was ambiguous (cadence vs. customer milestone/event)
    - WBS based on release cadence drives Control Account proliferation & administration
    - Defining the WBS based on customer milestone/event is suboptimal
  - Transitioning to agile from a waterfall WBS
  - Segregation by CLIN/Funding source

- **Lessons Learned**
  - Utilize capability based WBS combined with customer milestone/event based IMP
  - Work with customers to change traditional WBS practices
Agile Program Planning

**DEFINE THE WORK**

SOW Requirements are mapped to Epics and Features in the Program Backlog.

**PLAN THE WORK**

The Program Plan is reflected in the Release Roadmap, which is an initial allocation of Features and Epics from the Program Backlog to releases based on the objectives and goals of each release.

**SCHEDULE THE WORK**

Features are prioritized using the Release Roadmap and planned in the IMS.

Cross release planning occurs before the first release begins, later releases will be less well-defined.
Agile Program Planning

• Challenges
  - Mapping requirements (scope and budget) to Epics and Capabilities
    - Bid waterfall … executing agile
    - Transitioning from functional BOEs to Epics & Capabilities
  - Agile programs with undefined scope (bid as capacity)
  - Culture including roles and responsibilities

• Lessons Learned
  - Transitioning from bidding work in a waterfall fashion to Agile took some time
  - Overall agile approach to planning is working well
  - Agile programs with undefined scope do not accommodate EVM easily (and the same is true for waterfall)
  - Cultural changes are harder than technical changes
IMS and Critical Path

• The IMS should only go down to the level of Features (not story level)

• Utilize Rolling Wave Planning at Release Points

• Feature completion criteria and interdependencies are clearly defined
IMS and Critical Path

• Challenges
  – Just-in-time rolling wave planning (Change control period)
  – Release cadence span too long
  – Feature Acceptance Criteria drives completion (not completion of planned story points)
  – Feature duration greater than 40 days
  – Traditional Schedule Risk Analysis (Monte Carlo)

• Lessons Learned
  – Leverage customer direction to bypass change control period
  – IMS should provide critical path at high level (e.g. Features)
    • Story interdependencies can be modeled in agile tool
  – IMS tasks at Feature level allows freedom to prioritize/update stories within the feature without impacting the IMS. Stories provide QBD.
  – Use capacity, backlog and velocity for Schedule Risk Analysis
  – Incorporate agile metrics into customer reviews and status meetings (replaces detailed IMS metrics – LS/LF)
Features are comprised of stories. Each story is assigned a weighted story point (SP) value. SP’s are claimed at the completion of a story!
Control Account Hierarchy & EV

- **Challenges**
  - How to compute APC when stories change (added or deleted)?
    - Is scope the number of planned story points or feature acceptance?
      - Originally defined scope as number of SPs in order to manage change (prevent scope creep). Solved one problem but created another.
  - Story credit (0/100) is not given until story acceptance at iteration demo. Iteration that spans accounting month causes roller-coaster SV/CV spikes.
  - What happens to unfinished work at iteration and release points?

- **Lessons Learned**
  - Objective Criteria (completion of stories at weighted SP value) is easy and objective.
  - Agile team discipline (daily & iteration assessments) supports EV status & forecasting extremely well – better than non-agile programs.
  - Clearly defined completion criteria allows the stories within a Feature to evolve without a change to budget.
  - Consider taking 100% credit when Product Owner approves story (prior to demo). If other stakeholder involvement in approval is deemed critical take partial credit for stories when Product Owner approves, but pending demo (e.g., 80% at PO approval, 100% at demo acceptance).
  - Iteration and release boundaries have no impact on unfinished work.
  - Customer partnership and two way trust is critical for change management.
**Estimate To Complete (forecasting)**

<table>
<thead>
<tr>
<th>Date</th>
<th>Iteration #</th>
<th>Planned SPs per Iteration</th>
<th>Actual SPs per Iteration</th>
<th>Remaining SPs (Best Case)</th>
<th>Remaining SPs (Most Likely Case)</th>
<th>Remaining SPs (Worst Case)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AVG 32</td>
<td>Act AVG 30</td>
<td>AVG 36</td>
<td>AVG 30</td>
<td>AVG 24</td>
</tr>
<tr>
<td>10-Feb</td>
<td>1</td>
<td>29</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24-Feb</td>
<td>2</td>
<td>32</td>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9-Mar</td>
<td>3</td>
<td>33</td>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23-Mar</td>
<td>4</td>
<td>33</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6-Apr</td>
<td>5</td>
<td>32</td>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td>174</td>
<td>174</td>
<td>174</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOW</td>
<td></td>
<td>322</td>
<td>148</td>
<td>0</td>
<td>30</td>
<td>6</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ASSUMPTIONS**

- **BEST CASE**: Avg SPs of highest 3 iterations
- **MOST LIKELY**: Avg SPs of all 5 iterations
- **WORST CASE**: Avg SPs of lowest 3 iterations

**SCHEDULE FORECAST (TO GO)**

- **BEST CASE**: 5 Iterations
- **MOST LIKELY**: 6 Iterations
- **WORST CASE**: 8 Iterations

---

**Agile team performance to date (velocity) provides a basis for forecasting estimate to complete (ETC) for the remaining work**
Estimate To Complete (Forecasting)

• Challenges
  – Determining ETC beyond current release.
  – New and immature agile teams may have inconsistent velocity.

• Lessons Learned
  – Program Backlog should be “coarse sized” to allow forecasting across releases.
  – New teams will need a few iterations before accurate forecasting using velocity can be performed.
  – Burn Down Charts (agile metrics) expose unfinished work. Gives insight into schedule and cost growth.
  • PM feedback “Objective status of completed stories provided real progress and translated into early & fairly accurate ETC projections. ETC growth was quickly identified.”
Do’s and Don’ts

- **DO**
  - Leverage agile metrics and planning practices to support EVM planning, status, forecasting and analysis
  - Have a product centric WBS
  - Have a feature based IMS
  - Use Feature completion criteria to define scope
  - Use Rolling Wave Planning
  - Size all Epics and Features in the program backlog

- **DON’T**
  - Establish a release based WBS
  - Put stories or iterations in the IMS
  - Follow EVM or agile rules blindly
Remaining Challenges

• Change Management
  – Managing change that may involve a change in scope

• Culture
  – Agile is undisciplined and other myths
  – Changing roles and responsibilities

• Contracts & RFPs
  – Require traditional milestones (PDR, CDR)
  – Require traditional documentation (artifacts)
  – Require WBS that is not accommodating to agile
**PLATINUM CARD**

**EVM For Agile Development**

**AGILE DEVELOPMENT**

**EVM HIERARCHY**

**Variances**: Positive is Favorable, Negative is Unfavorable

- **Cost Variance**: CV = Burn Up Status - Actual Cost (BCWP - ACWP)
- **Schedule Variance**: SV = Burn Up Status - Release Plan (BCWP - BCWS)
- **Variance at Completion**: VAC = BAC - EAC

**DoD Metrics**

- **Favorable is > 1.0, Unfavorable is < 1.0**
- **Cost Efficiency**: CPI = Burn Up Status / Actual Cost (BCWP / ACWP)
- **Schedule Efficiency**: SPI = Burn Up Status / Release Plan (BCWP / BCWS)

**Program Agile Team Estimate @ Completion**

- **ETC** = Velocity x Remaining Backlog
- **EAC** = Actual Cost + (Velocity x Remaining Backlog)

**Independent Estimate @ Completion #**

- **EAC_{PLM}** = ACWP_{PLM} + [(Release Plan - Remaining Backlog) / CPI_{PLM}]
- **EAC_{Composite}** = ACWP_{Composite} + [(Release Plan - Remaining Backlog) / (CPI_{Composite} + SPI_{Composite})]

**To Complete Performance Index (TCPi) = Work Remaining / Cost Remaining**

- **TCPi_{PLM}** = Remaining Backlog (Velocity x Remaining Backlog)
  - (BAC - BCWP_{PLM}) / (EAC - ACWP_{PLM})

**Agile Terminology**

- **Backlog**: Collection of user stories the agile team will work on at some point in the future
- **Burn Up Chart**: Representation of the amount of user stories completed
- **Feature**: Cohesive business function or attribute of a software product or system. A single feature typically is implemented through many stories. Features provide the basis for organizing stories
- **Iteration (Sprint)**: Time period of fixed length during which the agile development team produces an increment of completed software
- **Release Plan**: Schedule for releasing software into productive use, made up of features and user stories
- **Story Points**: Characteristics of a user story, relative size measurement used by agile teams for work product estimation
- **Velocity**: Measured amount of work a team can complete in an iteration, typically in Story Points, used to measure how long it will take a particular team to deliver future outcomes by extrapolating on the basis of prior performance

**Acronyms**

- **ACWP**: Actual Cost of Work Performed
- **APC**: Actual Percent Complete (BCWP) / BCWP claiming criteria (completed/plan Story Points)
- **BAC**: Budget At Completion
- **BCWP**: Budgeted Cost for Work Performed
- **BCWS**: Budgeted Cost for Work Scheduled
- **CA**: Control Account
- **EAC**: Estimate At Completion
- **ETC**: Estimate To Complete
- **MR**: Management Reserve
- **PMB**: Performance Measurement Baseline
- **PP**: Planning Package
- **SPs**: Story Points
- **TAB**: Total Allocated Budget
- **TCPi**: To Complete Performance Index
- **WP**: Work Package

©2015 Lockheed Martin Corporation. All Rights Reserved