Software Acquisition

Facing the Challenge, Valuing Velocity!

NDIA SE Division Meeting

Washington DC

Mr. James Thompson
Director, Major Program Support
OUSD (R&E) / DASD (Systems Engineering)
Department of Defense
August 15, 2018
Code of Hammurabi (~1754 BC)

If a builder build a house for some one, and does not construct it properly, and the house which he built fall in and kill its owner, then that builder shall be put to death.


De Architectura on Cost/Schedule Overruns (~15 BC)

• [...] When an architect [in Ephesus] was entrusted with the execution of a public work, an estimate thereof being lodged in the hands of a magistrate, his property was held, as security, until the work was finished. [...] **But when more than one-fourth of the estimate was exceeded, he was required to pay the excess out of his own pocket.** [...]  

• Would to God that such a law existed among the Roman people, not only in respect of their public, but also of their private buildings, for then the unskillful could not commit their depredations with impunity, and those who were the most skillful in the intricacies of the art would follow the profession. Proprietors would not be led into an extravagant expenditure so as to cause ruin; architects themselves, from the dread of punishment, would be more careful in their calculations, and the proprietor would complete his building for that sum [...]

Software Is Everywhere

- DoD relies on software to provide decisive advantages to our forces
- The complexity required to achieve this advantage demands specific capabilities and tight coupling
- Partial solutions are inadequate
  - Because they don’t fit the schedule
  - Because it simplifies refactoring
DoD Software Failures in the Press

Air Force May Delay F-35 Combat Readiness Unless Software Improves
~ Investor's Business Daily

Billion-Dollar Flop: Air Force Stumbles on (ECSS) Software Plan
~ The New York Times

'Strongarissing to defend': US general blasts Raytheon's GPS control system a 'disaster'
~ Reuters

Senate Condemns US Air Force ECSS Program Management's Incompetence
~ IEEE Spectrum

Software Hurdles May Follow KC-46A Wiring Woes
~ DoD Buzz

Also, many successful software programs
Software Risk Assessed by DoD Program Offices

<table>
<thead>
<tr>
<th>Likelihood</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td><img src="#" alt="Green" /></td>
<td><img src="#" alt="Yellow" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="#" alt="Green" /></td>
<td><img src="#" alt="Yellow" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="#" alt="Green" /></td>
<td><img src="#" alt="Yellow" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="#" alt="Green" /></td>
<td><img src="#" alt="Yellow" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="#" alt="Green" /></td>
<td><img src="#" alt="Yellow" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
<td><img src="#" alt="Red" /></td>
</tr>
</tbody>
</table>

Software not in top program risks

Software among most frequent and most critical challenges, driving program risk on ~ 60% of acquisition programs
Historical Approaches to Address DoD’s Software Challenges

1984
Software Engineering Institute established

1987
Ada as single, common programming language DoDD 3409.1

1990

1994
Software Development and Documentation MIL-STD-498

1996
DoD CIO established; DoD 5000 required compliance Clinger-Cohen Act

2000

2001
GAO Report on Software / Systems Process Improvement (SPI)

2002
Implements CMM level 3 as goal DoD 5000.02-R

2003
Software / Systems Process Improvement (SPI) FY03 NDAA

2007
Time-Certain Development & MAIS Critical Change Reporting FY07 NDAA

2009
2 DSB Reports on Acquisition issues

2009
GAO Report: DOD Needs to Strengthen Management of its SSPI Efforts

2010
Implementation of New Acquisition Process For IT Systems FY10 NDAA

2011
2012
2015
GAO Reports on Acquisition & Agile

2015
Addition of SW-centric Models DoDI 5000.02

2017
DSB Reports on Software Acquisition

2017
MOSA, et al. FY17 NDAA

2018
FY18NDAA ??

LEGEND

Policy & Guidance
Congressional Actions
Reports
Historical Approaches to Address DoD’s Software Challenges

1984
Software Engineering Institute established

1987
Ada as single, common programming language DoDD 3408.1

1990
DoD SW shall be in Ada NDAA’s

1994
Software Development and Documentation MIL-STD-498

1996
DoD CIO established; DoDI 5000 required compliance Clinger-Cohen Act

2000
Time-Certain Development & MAIS Critical Change Reporting FY07 NDAA

2001
GAO Report on Software / Systems Process Improvement (SPI)

2002
Implements CMM level 3 as goal DoD 5000.02-R

2003
Software / Systems Process Improvement (SPI) FY03 NDAA

2007
2 DSB Reports on Acquisition issues

2009
Addition of SW-centric Models DoDI 5000.02

2010
GAO Report: DOD Needs to Strengthen Management of its SSPI Efforts

2011
GAO Reports on Acquisition & Agile

2012
Implementation of New Acquisition Process For IT Systems FY10 NDAA

2015
MOSA, et al. FY17 NDAA

2017
FY18NDAA ??

LEGEND
 Policy & Guidance  Congressional Actions  Reports
DSB on Software Acquisition

- “Software Factory”
- Continuous Iterative Development
  - Requirements/MVP
- Risk Reduction & Metrics
  - Competition
  - Cost/schedule scoping estimation techniques
  - Execution metrics framework
- Transition to Current and Legacy Programs
- Workforce improvement
- Software is Immortal - Sustainment
- Machine Learning – IV&V, Cyber

Software Cycle Time for Recent Programs

David M. Tate, Software Development May Drive Future Acquisition Cycle Times, (Revised) IDA Document NS D-8053 (Revised) October 2016
Log: H 16-000790 - Approved for public release; distribution is unlimited.
Estimated Schedule Durations for a Software Development Effort

Productivity required to meet planned 32-month schedule is substantially higher than industry averages and developer’s history on Programs A and B.

Planned duration appears unrealistic compared to historic data for Programs A and B, and to Industry and AT&L trend estimates.

Software benchmarks promote credibility in scoping, enable data driven decisions.
Sample Metrics Testing Optimism

2 April 2014: Testing Begins

Forecasted Completion
Sample Metrics Testing Optimism

- Test closure doesn’t go as expected
- Forecasted completion re-planned

Test Groups

- Total Test Groups
- Free-play Actuals
- Free-play Plans

Forecasted Completion Shifts Right
Sample Metrics Testing Optimism

- Still not as expected
- Forecast re-planned again

Test Groups

<table>
<thead>
<tr>
<th></th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
<th>Jul</th>
<th>Aug</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
<th>Jun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Test Groups</td>
<td>60</td>
<td>50</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free-play Actuals</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Free-play Plans</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Forecasted Completion Shifts Right again
Sample Metrics Testing Optimism

- Still not as expected

But Optimism starts to set in that schedule can be maintained

Test Groups

Total Test Groups
Free-play Actuals
Free-play Plans

Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec  Jan  Feb  Mar  Apr  May  Jun
Sample Metrics Testing Optimism

- Summit achieved – “Now we have full understanding”

Optimism fully realized – we can accelerate closure and push schedule left

Test Groups

- Total Test Groups
- Free-play Actuals
- Free-play Plans

Apr  | May  | Jun  | Jul  | Aug  | Sep  | Oct  | Nov  | Dec  | Jan  | Feb  | Mar  | Apr  | May  | Jun

0     | 10   | 20   | 30   | 40   | 50   | 60   | 50   | 40   | 30   | 20   | 10   | 0    | 0    | 0
Sample Metrics Testing Optimism

- Closing test cases but not on plan
- Blinding Optimism -- “Closure Cliff”
- Shift in forecast back to original
Sample Metrics Testing Optimism

Test Groups

- Total Test Groups
- Free-play Actuals
- Free-play Plans

15 Dec 14: Initial Completion Date Not Obtained
Contractor Surges to minimize delay
Forecast Re-planned

The graph shows the trend of test groups over time, with notations for initial completion date, contractor surge, and forecast re-planned.
Sample Metrics Testing Optimism

- Total Test Groups
- Free-play Actuals
- Free-play Plans

15 Dec 14: Initial Completion Date Not Obtained

...And again – 13 Mar 15 Completion
Sample Metrics Testing Optimism

Summary

- **Total Test Groups**
- **Free-play Actuals**
- **Free-play Plans**

- **15 Dec 14:** Initial Completion Date Not Obtained
- **13 Mar 15:** Completion
- **Contractor Surges to minimize delay**
Opportunities

**AUTONOMY**
- DSB: Autonomous solutions mitigate mission challenges
  - Enable rapid decision making
  - Manage a high volume of data
  - Coordinate complex actions
  - Ensure persistence and endurance

**ARTIFICIAL INTELLIGENCE**
- Improve on-board sensing
- Exploit time-critical intelligence from seized media
- Manage a dynamic spectrum for protection missions

**MACHINE LEARNING**
- Adapt acquisition and sustainment to rapid deployment
- Attain predictive logistics and adaptive planning

**CONTINUOUS CAPABILITY DELIVERY**
- Software “immortality”
- Iterative development and sustainment
  - Agile
Opportunities

MODELING & SIMULATION
• Enable warfighting capability and acquisition
  – Reduce risk
  – Accelerate programs
  – Save lives and $

MODEL BASED SYSTEMS ENGINEERING AND TEST
• System requirements, design, analysis, verification and validation across the life cycle
  – Improve communication
  – Manage increased complexity
  – Improve quality

EXPANDING SOURCES
• National SW resources and capabilities
  – Embrace U.S. leadership in SW
  – Utilize Silicon Valley
Challenges

Cyber-attack

- Increased SW provides a vulnerability path to a master OFF switch
- Need to protect key mission components from malicious activity
- As SW increase the potential for vulnerabilities increases exponentially
- Ensure key data is protected from adversary collection
- Strengthen Supply Chain Activities

SW is rate controlling step in acquisition

DoD acquisition program cycle time is dictated by amount of SW

Inadequate insight – performance to plan

Never quite sure where we are during development (inadequate data)

Always optimistic… until the end (can’t believe data we have)

David M. Tate, Software Development May Drive Future Acquisition Cycle Times, (Revised) IDA Document NS D-8053 (Revised) October 2016; Log: H 16000790 - Approved for public release; distribution is unlimited.
Challenges

Defects / unintended consequences

Impossible to exhaustively test systems

How do we establish and maintain trust in SW systems?

Grand architectures

Grand scales and unified architectures hamper timely solutions

Need to modify acquisition processes for “Minimal Viable Product”

SW – HW coupling

Current: HW is static in a system; SW evolution is limited by installed HW

Next Gen:
- Value modularity, consider decoupling/loosening integration
- HW abundance (cpu, bandwidth, etc.) needs to be factored into iterative acquisition

Software / data / skills are critical resources

DoD workforce is dated

Mean age: ~ 46

DoD workforce not sufficiently knowledgeable to SW challenges and solutions
### Agile/Classic Concepts

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Agile</th>
<th>Classic DoD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Involvement</td>
<td>Stakeholder Involvement</td>
<td>Integrated Product Teams</td>
</tr>
<tr>
<td>Approach</td>
<td>Continuous Iterative Development</td>
<td>Build a little, test a little, learn a lot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pre-Planned Product Improvement</td>
</tr>
<tr>
<td>Modularity</td>
<td>Incremental, “vertical slice” products</td>
<td>Chunk the problem</td>
</tr>
<tr>
<td></td>
<td>Incremental developments have value, and customers can use them.</td>
<td>Customers cannot start testing parts of the product until everything is complete.</td>
</tr>
<tr>
<td></td>
<td>Work can be broken into parts and conducted in rapid, iterative cycles.</td>
<td>Late changes are expensive or impossible</td>
</tr>
<tr>
<td></td>
<td>Late changes are manageable.</td>
<td>MOSA</td>
</tr>
</tbody>
</table>

- In-phase defect containment
- Total Quality Management

---

Incomplete data on agile at scale, with some exaggeration – However…
Empirical data and strong industry movement to agile development across all domains strongly motivates DoD to move to agile development
# Addressing the Challenge!

## Valuing velocity

<table>
<thead>
<tr>
<th>Challenge/Opportunity</th>
<th>Approach</th>
</tr>
</thead>
</table>
| Software/data are critical assets, software/related fields are critical skills | • Cultivate workforce, centers of excellence, capabilities, base  
• Software Factory, repository  
• Outreach to industry, academia |
| Software has become rate controlling step in acquisition   | • Credibility in planning & execution – improve scoping & metrics framework  
• Establish/demonstrate Software Factory early (TMRR), use in evaluations  
• Common production/mission representative development and test environments  
• Harness technology (e.g., automation (build, test), machine learning, etc.) |
| Grand scales and unified architectures are challenging     | • Modify requirements process - “Big R” - enable MVP / P3I  
• Modify acquisition processes - right-size programs and approaches  
• Value modularity, consider decoupling/loosening integration  
• Reduce customization (enterprise optimization)  
• Promote multi-functionality |
| Software enables opportunity for continuous capability delivery | • Software is immortal  
• Iterative development – initial development and sustainment |
| Hardware limitations are barriers                         | • Hardware abundance (cpu, bandwidth, etc.) supporting iterative acquisition  
• Promote hardware/software independence |
| Cyber/program protection / resilience add layers to challenge | • Build in capability  
• Refresh opportunity  
• Manage risk  

• “Software by the pound”

**Implement on current and legacy programs**
Conclusion

- **Unrelenting demand for SW in DoD**
- **SW is the fuel for innovation and future capabilities**

- **Action needs to be taken to meet challenges**
  - Diversify the Department’s approach and sources
  - Embrace opportunity in a change-rich environment (threats, technology, process improvements)

**Shift to threat-based acquisition demands enhanced velocity**

Shift SW acquisition to an enabler of speeding capability
Fearless workers who stood with their back to a TANK roaring towards them to prove its stopping power

Link: http://video.dailymail.co.uk/video/1418450360/2014/01/1418450360_3121005887001_tank-brake-test.mp4
Systems Engineering: Critical to Defense Acquisition

Innovation, Speed, Agility

http://www.acq.osd.mil/se
What About Agile?

• **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 we deliver **capabilities**?

• **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?

DoD Systems tend to be complex, with independently-developed, highly-coupled components
CMMI Thoughts

• Previous experience with CMMI “good in theory, but not so much in practice”
  – Chasing maturity levels vs. continuous performance improvement.

• Concept of the “Software Factory” (consistent, realistic outcomes for schedule, cost, quality and functionality performance of SW)
  – Make “Software boring”

• Agile also not without its issues – still comes down to discipline, if the discipline is missing, still poor quality of code, no matter how soon it can be done.

• Move beyond Agile / DevOps – e.g., Software Lifecycle Automation

• Acquisition Challenges: Greater reliability and consistency (and insight) into contractor’s abilities to meet the mission requirements for SW, Systems, etc.
  – Being able to distinguish the real differentiators between contractors re: performance