Driving Efficiencies into the Software Life Cycle for Army Systems

Stephen Blanchette Jr.

Presented to the CECOM Software Solarium
7 September 2016
[Distribution Statement A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM-0003950
Where behavior counts most…

...much is required of software
The Army’s software challenges are many

Huge system/software engineering endeavors in weapon systems, command and control systems, enterprise systems, simulators, even ammunition

- Several million SLOC programs; hybrid systems combining legacy re-use, COTS, new development
- Multi-contractor teams using different processes; dispersed engineering, development & operational locations
- New technologies creating opportunities/challenges; products change/evolve, corporations mutate
- Business/operational needs change - often faster than full system capability can be implemented
- Skillset shortfalls; cost and schedule constraints
- Increased demands for integration, info/cyber-security, interoperability, system of system capabilities
- Software increasingly connects other systems (manned/unmanned teaming, swarming, etc.)
- Enterprise perspectives/requirements
- Growing (but late) concerns about sustainment
We tend to make the same mistakes…

Requirements

• Undocumented assumptions
• Creep, instability
• Functionality – System Qualities = Incomplete Specs

Architecture – for software???

Assume testing yields quality

Ignoring and/or marginalizing the role of software from program outset

We can do better!
We can use incremental methods to embrace change...

...but also control it

Examples include Spiral development, Incremental Commitment Model, Agile methods such as Scrum and XP, etc.
We can employ software architecture to provide a foundation 1 of 2

To help plan out how the pieces go together…

Most modern systems are hybrids
We can employ software architecture to provide a foundation 2 of 2

...and to reason about changes

What if I remove... this one?
We can take a more holistic view of V&V
We can take a more holistic view of V&V.
We can employ evidence-based reviews
We can employ evidence-based reviews
Most important – we need to start making changes!

Stop pretending software doesn’t matter!
  • Include software experts in project planning

Pay attention to long-term sustainment drivers
  • Plan for licensing and data rights
  • Buy the work products necessary for owning the technical baseline

Stop being afraid of software
  • We know there are challenges, so let’s face them head on!

Reduced complexity is not in the cards; we need to be better engineers & managers