How to be Successfully Agile

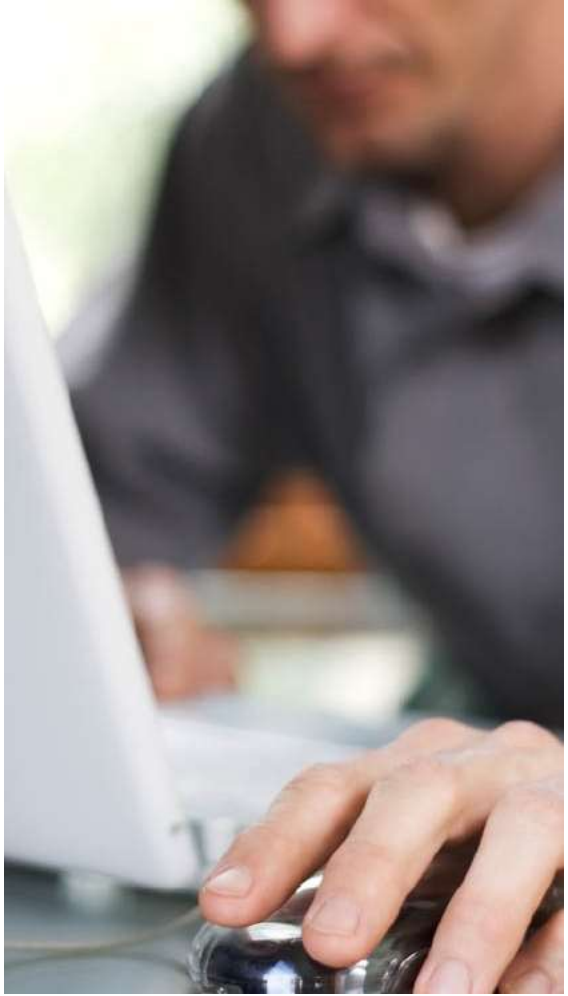
Keith Korzec

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM23-0015

Lessons Learned



Leadership/Sponsorship

- Need 100% leadership and organizational support
  - Continuous involvement in the program (to keep aware of needed executive-level decisions)
- Reinforce both sides have “skin in the game”
- Have an agile champion (of high enough grade to affect change) and an organic agile support team
  - Consider an Agile Executive Support Office (AESO) concept (specifically *not* a Center of Excellence)

Stakeholder Management

- Think about how the acquisition/development program plans to engage with operational users/consumers early and often
  - A User Agreement (from 5000.87) is a great way to start.

Quarterly Planning Cadence & Events

- PI planning events are expensive, but worth it
  - Cannot overstress this point: Otherwise, where would teams *ever* have such an opportunity to share critical information and get a better comprehension of the context
- Begin with a PI Zero to test the system and practice what was learned in training
- Don’t gloss over or skip key agenda items (architecture, context, vision briefs) during PI planning. This refreshes the constraints of existing team and helps new members understand the context.

Dealing with Requirements

- Use functional / mission threads to define and refine requirements
- No requirement started without an approved acceptance criteria
- Prioritize features based on mission value/need
- Perform trade studies of performance that make the case for investing in a particular capability
- Create naming convention before starting so that epics can be linked all the way to tasks and vice versa
  - Require bi-directional traceability
  - Prohibit orphaned stories/features
  - Enforce size requirements to aid flow

Training and Adoption Support (incl tooling)

- Train **everyone** on the new way you are doing business
- Fund a development environment, test facility, integration environment and integration tools before beginning any work on the system
- Plan for changing roles & responsibilities within the government program office
  - Staff appropriately within the program office
  - Agile roles cannot be additional duties
  - Include IA/Cyber/fielding/OUDTE as members of the architecture/engineering team
- Include the Definition of Done in the RFP, work with developer to refine the Definition of Done immediately after contract award
- Take advantage of the ROAM process in addition to the contractor's normal risk management system
- Avoid accumulating technical debt in order to "show" progress
- There should be planning from day one as to "continuity of care" for the system

Integration and Testing 1

- Start integrating the components on day one and never stop integration
- Automate **all** testing
  - Regression test points grow in number and complexity with each new release
- Build strategies that enable verification at component and subsystem levels
- Use modelling as much as possible
  - Simulation-intensive tests that exercise a thread of functionality relevant to the user
  - Models that give a proof of concept that a target level of performance is achievable/affordable
  - Incorporate system and test lab (such as the ESIL) models into the developer's workflow to increase quality

Integration and Testing 2

- Use operational test threads (open air sorties) that target the functionality being added
- Allocation of test points to a unit level or to a specific component-to-component or component-to-system interface
- Rapidly test as often as possible from the outset
  - Testing the integration and performance of the software in-house is the area that bears the most fruit
  - Can be a significant upfront cost but reduces the total lifecycle costs

Contact Information

Keith Korzec
Senior Engineer
SSD/CTSD
Email: kkorzec@sei.cmu.edu

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800