Agile Verification and Validation

Dr. Supannika Mobasser
Stephen Blanchette

Systems Engineering Forum
Agility in Acquisition
12 April 2022

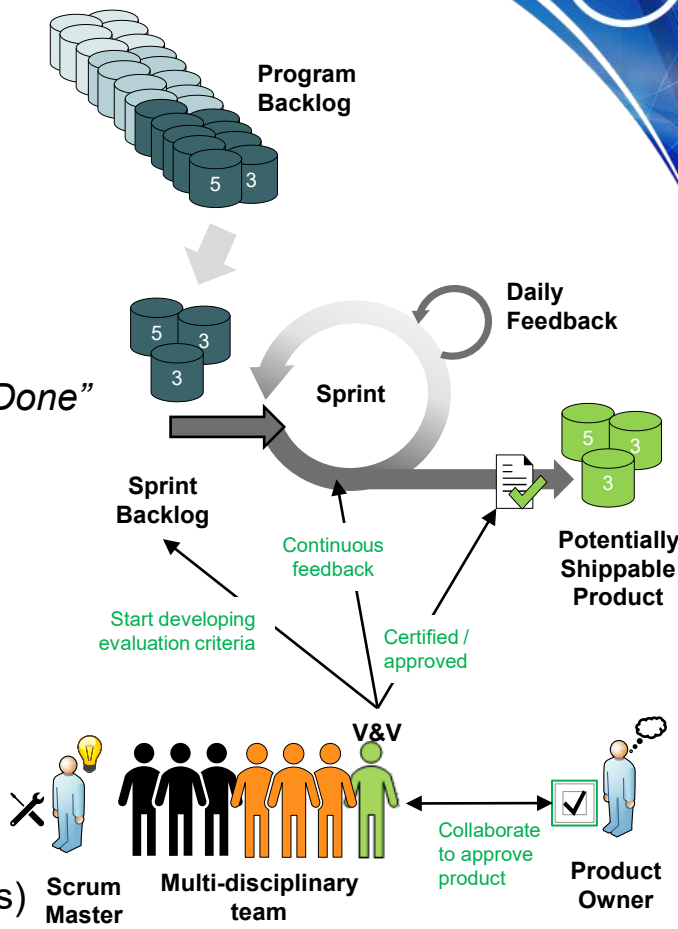
© 2022 The Aerospace Corporation

Agile Verification and Validation (V&V)

- Alternative V&V approaches for Agile
  - *Integrated V&V (not independent)*
    - V&V team is part of the Agile team
  - *Staggered V&V Sprints*
    - V&V team is running one Sprint behind to certify / evaluate the incrementally developed products
  - *Kanban*
    - No timebox, continuous flow of works; support varying sizes of tasks
  - *V&V-Driven-Development*
    - Develop metrics, threshold, or run test scripts to establish baseline, then incorporate continuous monitoring
  - *V&V Escrow*
    - Provide continuous oversight and performs periodic (quarterly) code inspection
  - *V&V in DevSecOps*
    - Automate the evaluation as much as possible to support continuous delivery

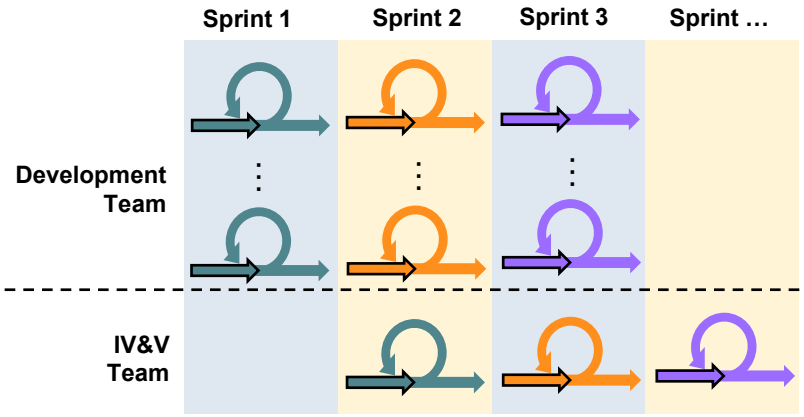
Integrated V&V (not independent)

- Practices
  - *V&V personnel is part of the development team*
  - *Provide instant feedback through tools or activities*
    - *Tools : acceptance criteria, test scripts, code quality tools*
    - *Activities : peer review, support Product Owner to declare "Done"*
- Pros
  - *Insights on all software development steps*
  - *In-sync with development team, open exchange information*
  - *Real-time feedback, less cost of rework*
- Cons
  - *No longer "independent"*
  - *Need adjustment on team structure, culture, battle rhythm*
    - V&V personnel may come from different organizations
    - Ideally at least one dedicated V&V per team (5-9 developers)
  - *Very fast pace*
    - Need to code, test, certify within a (typically 2-week) Sprint
- Resources
  - *~1 full-time person per development team*



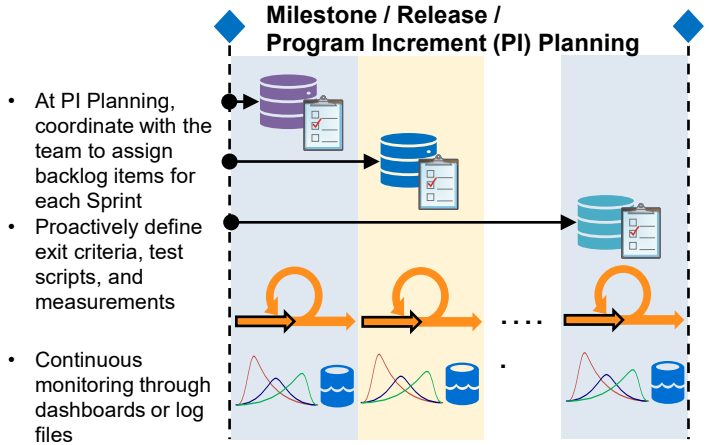
Staggered V&V Sprints

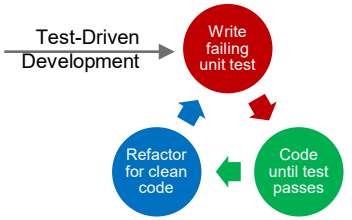
- Practices
  - *V&V team trails the development team(s) by one Sprint*
    - Evaluate or certify products delivered from development team every sprint, provide feedback every Sprint
- Pros
  - *Low cost of rework; only take a month to rework (2-week Sprint)*
  - *One V&V team could support multiple development teams*
  - *V&V personnel are independent but frequent synchronizations*
    - Clear synchronization tempo between teams
  - *More time to evaluation*
    - No need to rush code-test-evaluate in the same Sprint
- Cons
  - *V&V works highly dependent on Development team works*
  - *May be perceived as outsiders or "us vs them" mindset*
  - *Late identification of defects, potentially costly rework*
- Resources
  - *~0.5 to 1 full-time person per development team*



V&V-Driven Development

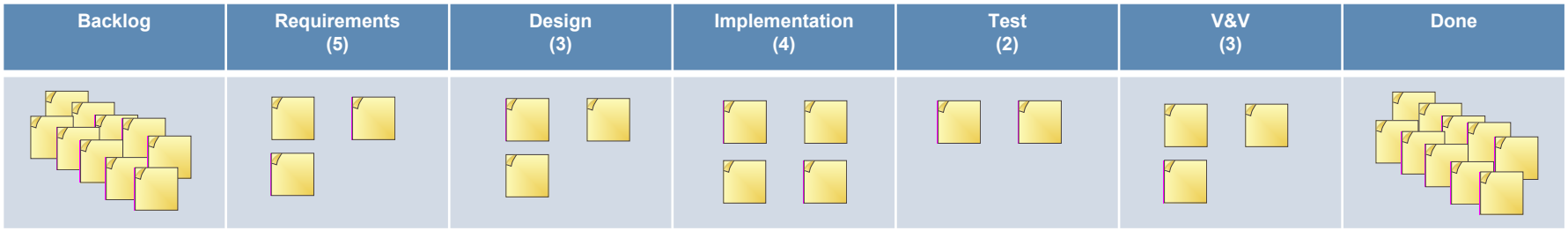
- Practices
  - *Similar to Test-Driven Development, define acceptable thresholds and metrics upfront, or prior to development*
  - *Run test scripts to establish baseline, incorporate continuous monitoring*
  - *Share visible results daily with V&V personnel*
- Pros
  - *Proactively set the goal, metrics, dashboard*
  - *Don't know the whole, but know what the team is working on*
- Cons
  - *High upfront collaboration to set up expectations*
  - *Need infrastructure in place; monitoring dashboard, tools, environments*
    - Be on the same page on clarification
  - *May be perceived as giving the answers upfront*
- Resources
  - *~1 full-time person for a small program (10 – 20 developers or low program complexity)*
  - *~2 to 3 full-time people for a medium program (20 – 50 developers or moderate program complexity)*
  - *~3 to 5 full-time people for a large program (more than 50 developers or high program complexity)*





Akin to Test-Driven Development, which is a popular Agile technique

Kanban

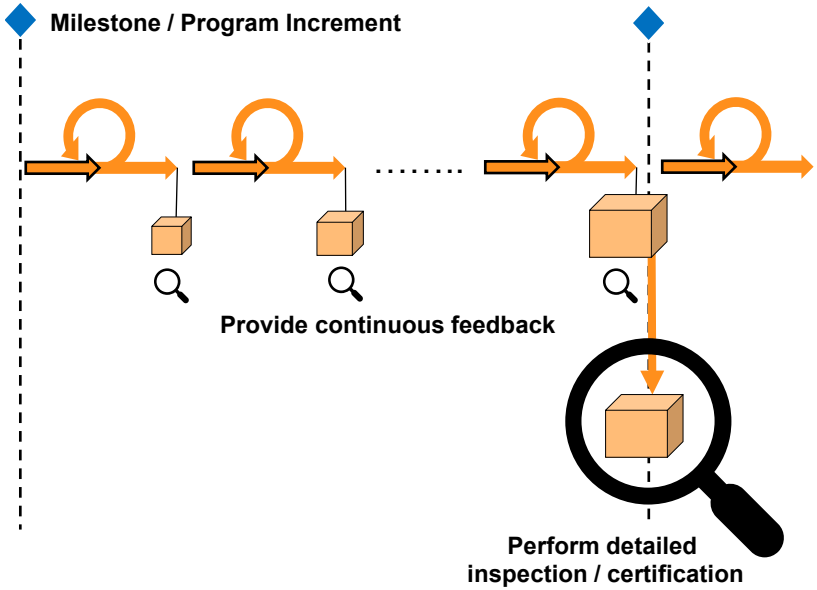


- Practices
  - *Assembly line; Continuous flow of tasks, V&V as a final step before "Done"*
  - *Use "Pull" system to pull task in only if available (under "Work-In-Process" (WIP) limit); one piece of work at a time*
  - *More appropriate for sustainment / enhancement programs*
- Pros
  - *No timebox, work on your own speed, especially when task durations are not the same*
  - *Easy to see the bottlenecks*
- Cons
  - *Need to manage Work-in-Progress (WIP) carefully, otherwise it could block the flow; traditional IV&V takes a long time*
  - *Late feedback loop, higher cost of rework*
  - *IV&V could be throwing wrenches into development with late findings*
- Resources
  - *Proportional to amount and speed of the workflow*

Few programs use Kanban as their main method, which may limit applicability of this approach

V&V Escrow

- Practices
  - *Like escrow agent; ensure that the system is working properly*
  - *Participate to provide feedback at sprint review, system demo*
  - *Perform detailed inspection at each major milestone*
    - *Quarterly or semi-annually*
- Pros
  - *Has up-to-date knowledge on incrementally developed product*
  - *Able to provide continuous feedback with detailed inspection*
- Cons
  - *Batch detailed review, medium loop feedback*
  - *Require additional resources to participate team reviews*
  - *V&V could be throwing wrenches with late finding; schedule could be fall behind and out-of-sync*

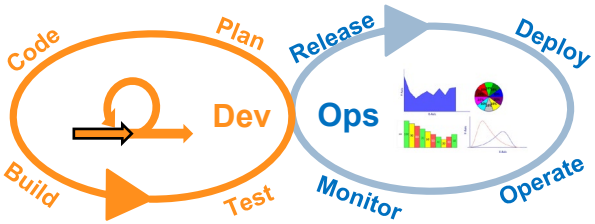


- Resources
  - *~1 full-time person for a small program (10 – 20 developers or low program complexity)*
  - *~2 to 3 full-time people for a medium program (20 – 50 developers or moderate program complexity)*
  - *~3 to 5 full-time people for a large program (more than 50 developers or high program complexity)*

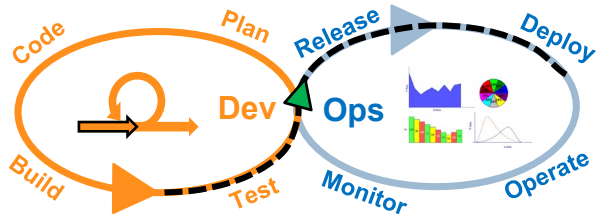
A balanced combination of quick feedback and detailed inspection

V&V and DevSecOps

- Practices
  - *Automate testing and evaluation steps as much as possible*
  - *May cover from testing in test environment, release to staging environment and deploy to Ops-like environment*
  - *Easier to start with sustainment / enhancement programs*
- Pros
  - *Support programs that require continuous delivery*
    - Either to Ops or Ops-like environment
- Cons
  - *Requires highly automated processes to support continuous integration and continuous delivery; Manual evaluation could slow down the process*
  - *Requires first-class configuration management system*
  - *Potentially challenging for traditional IV&V*
  - *Product is never finished*
- Resources
  - *1 full-time person for a small program (10 – 20 developers or low program complexity)*
  - *2 to 3 full-time person for a medium program (20 – 50 developers or moderate program complexity)*
  - *3 to 5 full-time person for a large program (more than 50 developers or high program complexity)*



DevOps or DevSecOps



IV&V with DevOps or DevSecOps

DevSecOps is a popular approach, which may make this solution especially attractive

Summary

Alternative V&V approaches for Agile

- Integrated V&V (not independent)
  - *V&V team is part of the Agile team*
- Staggered V&V Sprints
  - *V&V team is running one Sprint behind to certify / evaluate the incrementally developed products*
- Kanban
  - *No timebox, continuous flow of works; support varying sizes of tasks*
- V&V-Driven-Development
  - *Develop metrics, threshold, or run test scripts to establish baseline, then incorporate continuous monitoring*
- V&V Escrow
  - *Provide continuous oversight and performs periodic (quarterly) code inspection*
- V&V in DevSecOps
  - *Automate the evaluation as much as possible to support continuous delivery*


Questions