Continuous Integration: Verification and Validation in an Agile Environment

FAA Verification and Validation Summit 2014

Nick Bartlow, Ph.D.
Lead, Noblis Center of Digital Excellence

September 18th
Where this talk is headed

{(P) C {Q}}
{a > b}
begin max(int a, int b)
  if a >= b
    return a
  else
    return b
end
{ result == a }
Some Context Setting on Development Models

Predictive Models
- Requirements
- Design
- Implementation
- Verification
- Maintenance

Iterative Models
1. Determine objectives
2. Identify and resolve risks
3. Development and Test
4. Plan the next iteration

Agile / Adaptive Models
- Scrum
- XP
- OpenUP
### Comparison of Waterfall, Spiral, and Agile

#### Agile
- Individuals and interactions
- Working software
- Customer collaboration
- Responding to change

#### Waterfall
- Process and tools
- Comprehensive documentation
- Contract negotiation
- Following a plan

#### Agile vs Waterfall
<table>
<thead>
<tr>
<th>Agile</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterative Planning</td>
<td>Upfront Disciplined Planning</td>
</tr>
<tr>
<td>Necessary Documentation</td>
<td>Comprehensive Documentation</td>
</tr>
<tr>
<td>Emerging Evolving Requirements</td>
<td>Fixed Requirements Upfront</td>
</tr>
<tr>
<td>Minimum Analysis &amp; Design Needed</td>
<td>Comprehensive Analysis &amp; Design</td>
</tr>
<tr>
<td>Changes Expected/Acceptable</td>
<td>Changes Unexpected/Discouraged</td>
</tr>
<tr>
<td>Empowered Teams</td>
<td>Command &amp; Control</td>
</tr>
<tr>
<td>Soft Control</td>
<td>Structured Control Methods</td>
</tr>
<tr>
<td>Integrated &amp; Flexible</td>
<td>Linear &amp; Rigid</td>
</tr>
</tbody>
</table>

#### Agile* vs Spiral*

<table>
<thead>
<tr>
<th>Agile*</th>
<th>Spiral*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Each increment delivers functioning software</td>
<td>Increments sometimes deliver prototypes</td>
</tr>
<tr>
<td>Primary focus on delivery of value to end user</td>
<td>Primary focus on risk management</td>
</tr>
<tr>
<td>1-4 week increments</td>
<td>6 mo – 2 yr increments</td>
</tr>
<tr>
<td>Suitable for both large and small projects</td>
<td>Typically best for large projects</td>
</tr>
</tbody>
</table>

*Comparison incites much debate, mileage can vary based on specific implementations*
Agile and Scrum: Team-level Processes, Tools, and Methods

- Review product backlog
- Establish Sprint goals
- Collaboratively estimate sprint backlog items
- Commit

- Requirements Breakdown
- Client-prioritized product features
- User Stories / Use Cases

- Integrated sub-teams

- Features assigned to sprint
- Estimated by team
- Team Commitment

- Create value
- Report & Monitor Progress
- Continuous interaction
- Continuous integration and automated testing

- Potentially shippable, fully tested product increment
- Rapid delivery of business value

- Demo features to team, client and stakeholders for feedback
- Continuous process improvement
Testing and Quality Assurance

Goal: To build, integrate, test and deploy application software with production level quality in a production-like environment every single day.

- QA occurs throughout each sprint/iteration
  - Catch defects earlier
  - Reduce rework
  - Avoid surprises
  - Manage feature scope

- Testers are key members of cross-functional teams
  - Definition of Done - Ensure increment produced at sprint end is potentially shippable
  - Acceptance Criteria - What Product Owner expects; what team must accomplish

- Continuous Integration and Automated Testing enable Agile delivery
  - Regression and system testing activities grow with each sprint in new product development
  - Automate testing where possible to allow manual testing where needed
What is Continuous Integration (CI)?

- **Continuous Integration (CI)** - A development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early [Fowler, 2006].

- **Keys:**
  - Maintain a single source repository
  - Automate the build
  - Make your build self-testing
  - Everyone commits to the mainline every day*
  - Every commit should build the mainline on an integration machine*
  - Keep the build fast
  - Test in a clone of the production environment
  - Make it easy for anyone to get the latest executable
  - Everyone can see what’s happening

*The state of the art of combining CI and branching has changed in the last few years, no longer strictly a requirement*
Validation and Verification Opportunities in Agile

Each 2 week increment provides an opportunity for end user to assess whether we’re “Building the right product”

Within increments, each run of our automated CI pipeline can / should involve validation and verification activities
Continuous Verification and Validation Opportunities

Validation

Continuous deployment to QC / QA / Staging Environments can facilitate:

- Interactive testing
- FAT / SAT / etc.
- Stakeholder Demonstration

Example CI Tooling

Manual Verification

Automated Verification

Build Artifacts

Automated Test Execution

Static Analysis

Metrics / Reports

Code Reviews
TDD / ATDD / BDD – Preemptive Verification and Validation

- **Test Driven Development (TDD)**
  - Write tests first and establish failures
  - Write minimal functionality to allow tests to pass
  - Facilitates incremental, continuous growth of test suite

- **Acceptance Test Driven Development (ATDD)**
  - Emphasizes developer-tester-business customer collaboration
    - Given (setup)
    - When (trigger)
    - Then (verification)

- **Behavior Driven Development (BDD)**
  - Specify tests in business readable domain-specific language
  - Generate automated tests stubs from plain text behavior
Validated Merging – Maintaining Releaseability to Facilitate Delivery on Demand

- Develop on branches
- CI tooling executes full suite of automated testing and analysis
- Criteria is established that determines whether a changeset is considered healthy
  - **IF** criteria is met
    - CI tooling automatically merges branch into mainline
  - **ELSE**
    - CI tooling fails the build and the mainline is insulated from harmful changes
- Benefits
  - Maintains stability consistent with quality of your test / analysis suite
  - Prevents wasted time debugging non-problems in large and diverse teams
  - Mitigates risks associated with complex integrations

Using Branch Diagrams to Scare your CEO!

https://confluence.atlassian.com/display/BAMBOO/Using+plan+branches#Usingplanbranches-Gatekeeper
Mission Criticality and Provable Correctness

The principles of Continuous Integration are applicable no matter what Design Assurance Level (DAL) is required of the software you are building.

- Demonstrably “correct"
- Code behaves according to expectations given both valid and invalid input values, at least at the unit level
- Code complies with any relevant standards or guidelines
- Code satisfies basic non-functional requirements insofar as this can be validated using the tools normally available to programmers

- Formal methods lite
- Run-time verification
- Automated test generation from formal specifications

- Formal methods
- Provably correct properties of specifications
- Provably correct implementations w.r.t specifications
- Model-based design, verification, validation, and code generation

"No Silver Bullet"

"Testing can show the presence of errors, but not their absence."

"One can never guarantee that a proof is correct, the best one can say is: 'I have not discovered any mistakes'"

http://davenicolette.wordpress.com/2012/10/23/delivering-provably-correct-code/

Increasing level of Criticality
Scaling Agile – Scaled Agile Framework (SAFe)

- Synchronizes alignment, collaboration, delivery
- Scales successfully to large numbers of teams
- Agile Release Train Size
  - 5-12 agile teams
  - 50-125 individuals planning, committing, and executing together

Core values:
1. Code Quality
2. Program Execution
3. Alignment
4. Transparency

http://ScaledAgileFramework.com
A Scaled Agile Portfolio Approach ≠ “Top Down”

Centralized Strategy, Decentralized Execution

- $$$ Architectural Investments
- $$$ Business Investments

ENTERPRISE PORTFOLIO

VISION

Program A

Program B

Program C

TEAM 1

TEAM 2

TEAM 1

TEAM 2

TEAM 3

TEAM 4

Agile Delivery on Demand
How Scaled Agile Approaches Relate to Complex Systems

An Oversimplified Hypothetical Related to SWIMS
This probably isn’t news…

From *Development, Validation, and Demonstration of Technologies for the Health and Usage Monitoring Systems Airborne- and Ground-Based Automated Testing and System Functionality Partition*, December 2009

- “Best practices (and the supporting tools) discussed in this report include short iterations, test-driven development, customer tests, documentation techniques, collective code ownership, and continuous integration”
- “Even highly structured processes and certifications such as Capability Maturity Model Integration (CMMI) (popular in aerospace settings) have found compatibility with agile methods”
- “DO-178B is clearly compatible with iterative methods.“
- “…iterative fashion would likely limit the overall cost of certification assessment and prevent the cost of incorporating requirement changes after certification”
- “TDD [Test-driven Development] is compatible with DO-178B.”
- “A static analysis tool can easily be added to an automated build system that executes both dynamic and static analysis tests before releasing compiled artifacts”
- “Continuous integration is the practice of rebuilding and testing an application frequently.”
- “Continuous integration allows a system to be built, tested, and packaged at moment’s notice. As such, the most recent working system is always at hand.”
- “Within the context of DO-178B, an automated build tool enables the practice of continuous integration…it is the automated build tool that runs all unit and static analysis tests, performs automated acceptance testing, generates automated documentation, and builds the release executables.”
From **FAA Test and Evaluation Process Guidelines**, April 2014

- “Early testing is crucial.”
- “Avoid last minute test procedure changes”
- “A system is not truly tested until it is used by its target audience in an operational setting.”
- “Encourage and facilitate communication and interaction between the FAA and the developers”
- “Use an integrated test team approach”
- “Air Traffic and Technical Operations test teams need to communicate.”
- “Facilitate a team effort”

From **FAA Test and Evaluation Handbook**, September 2013

- “V&V is performed in varying degrees on a continuous basis by many entities involved in the acquisition.”
References & Resources


- Continuous Integration, Martin Fowler (01 May 2006), [http://martinfowler.com/articles/continuousIntegration.html](http://martinfowler.com/articles/continuousIntegration.html)


Thank You

Questions? Please contact:
   Nick Bartlow, Ph.D.
   Lead of Center of Digital Excellence (CoDE)
   Email: nicholas.bartlow@noblis.org
   Office: 304.848.3916