Continuous Integration: Verification and Validation in an Agile Environment

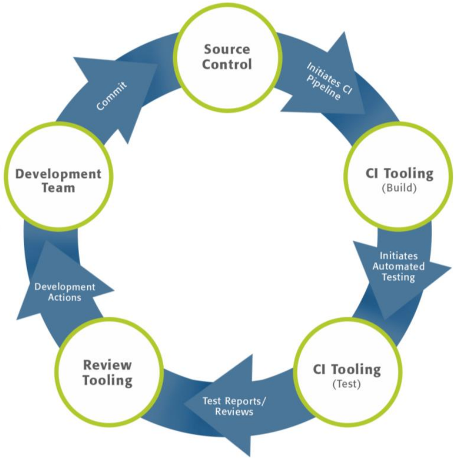
FAA Verification and Validation Summit 2014

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

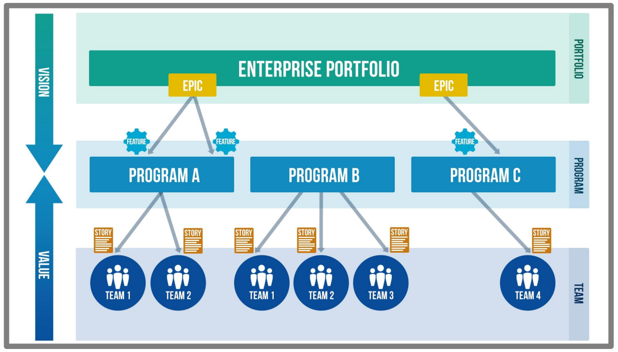
September 18th

noblis
For the best of reasons

Where this talk is headed



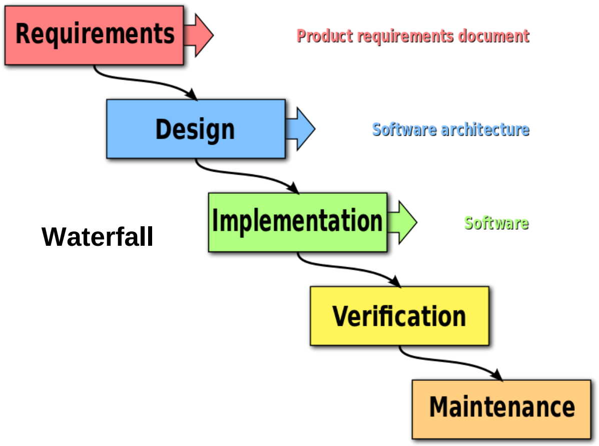
```
{P} C {Q}
---------

{a > b}

begin max(int a, int
b)

   if a >=b
      return a
   else
      return b
   end

end

{ result == a }
```



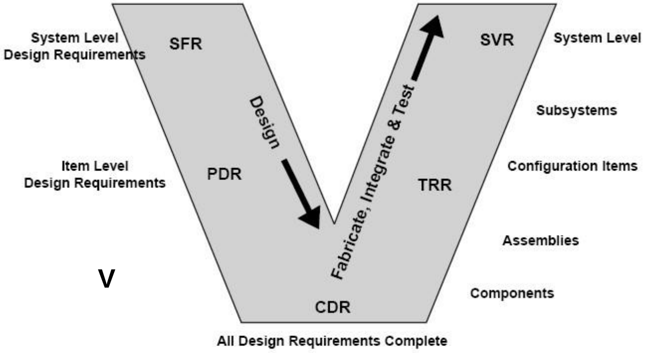
Some Context Setting on Development Models

Predictive Models

Waterfall



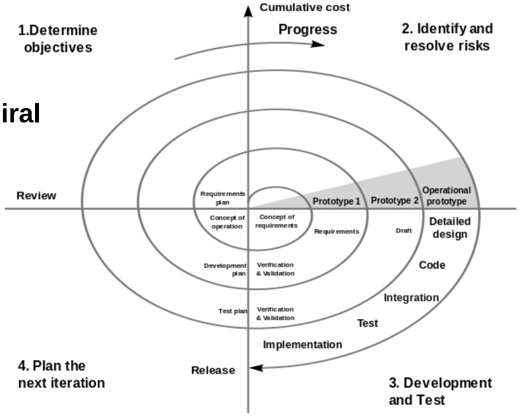
V



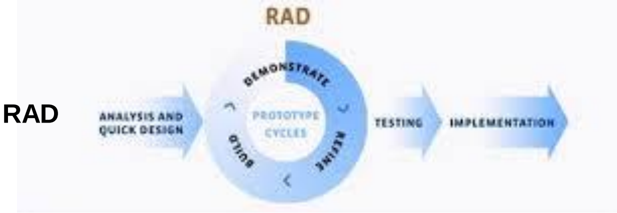
SFR = System Functional Review
PDR = Preliminary Design Review
CDR = Critical Design Review
TRR = Test Readiness Review
SVR = System Verification Review

Iterative Models

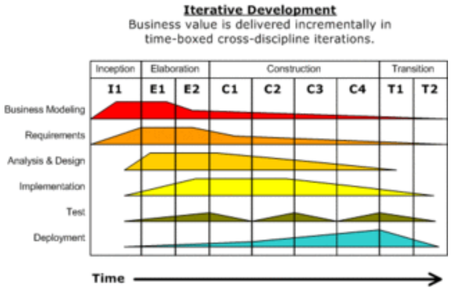
Spiral



RAD

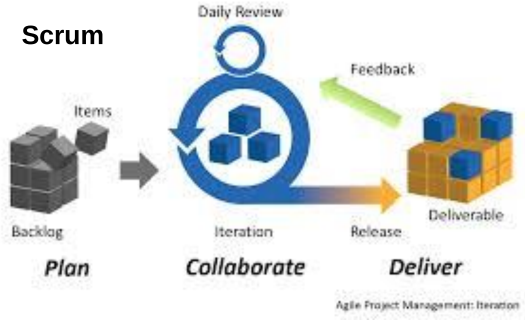


RUP

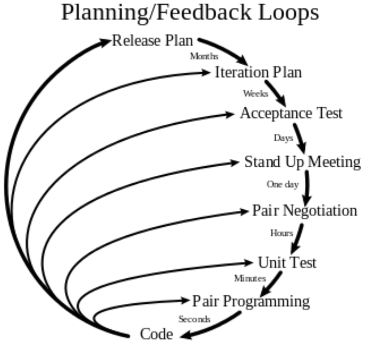


Agile / Adaptive Models

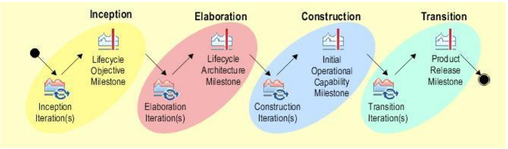
Scrum



XP

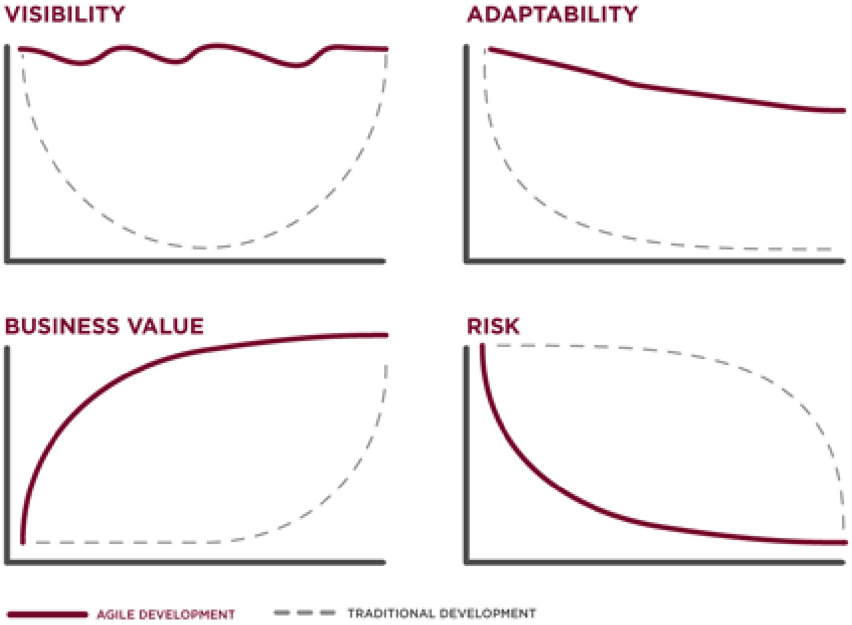


OpenUP


Inception
Elaboration
Construction
Transition
Lifecycle Objective Milestone
Lifecycle Architecture Milestone
Initial Operational Capability Milestone
Product Release Milestone
Inception Iteration(s)
Elaboration Iteration(s)
Construction Iteration(s)
Transition Iteration(s)

Comparison of Waterfall, Spiral, and Agile

Agile		Waterfall
Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan



Agile	Waterfall
Iterative Planning	Upfront Disciplined Planning
Necessary Documentation	Comprehensive Documentation
Emerging Evolving Requirements	Fixed Requirements Upfront
Minimum Analysis & Design Needed	Comprehensive Analysis & Design
Changes Expected/Acceptable	Changes Unexpected/Discouraged
Empowered Teams	Command & Control
Soft Control	Structured Control Methods
Integrated & Flexible	Linear & Rigid

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

***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

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

VERSIONONE
Agile Made Easier

JIRA

Travis

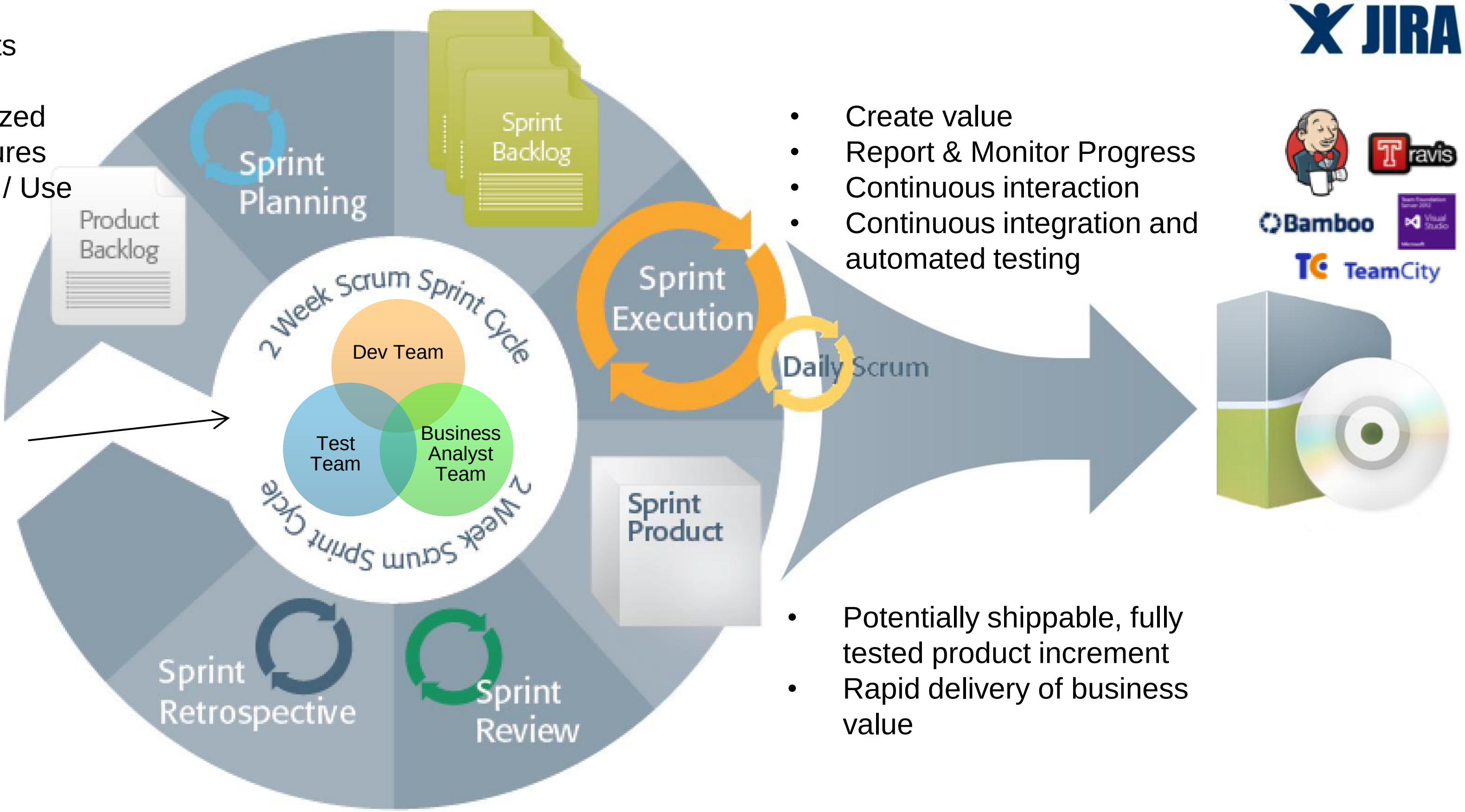
Bamboo

Visual Studio

TeamCity

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

- Integrated sub-teams



- 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

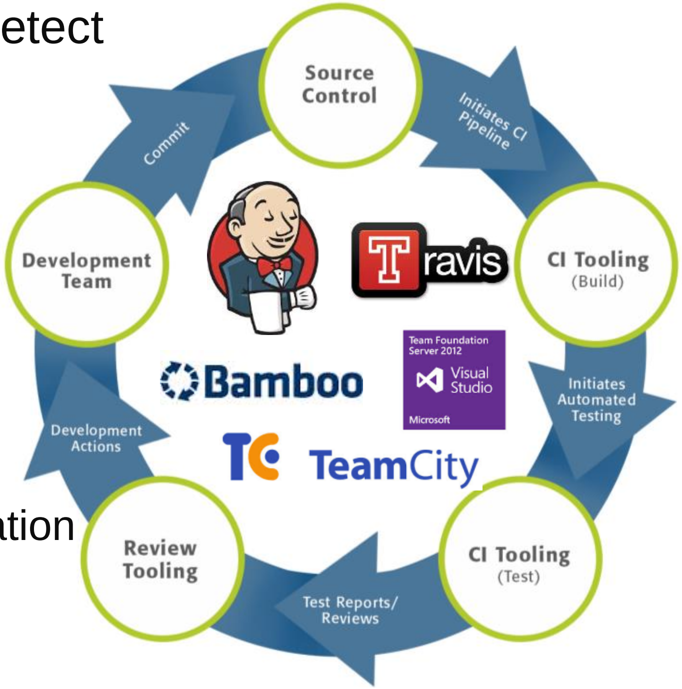
Goal: To build, integrate, test and deploy application software with production level quality in a production-like environment every single day.

- 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)?

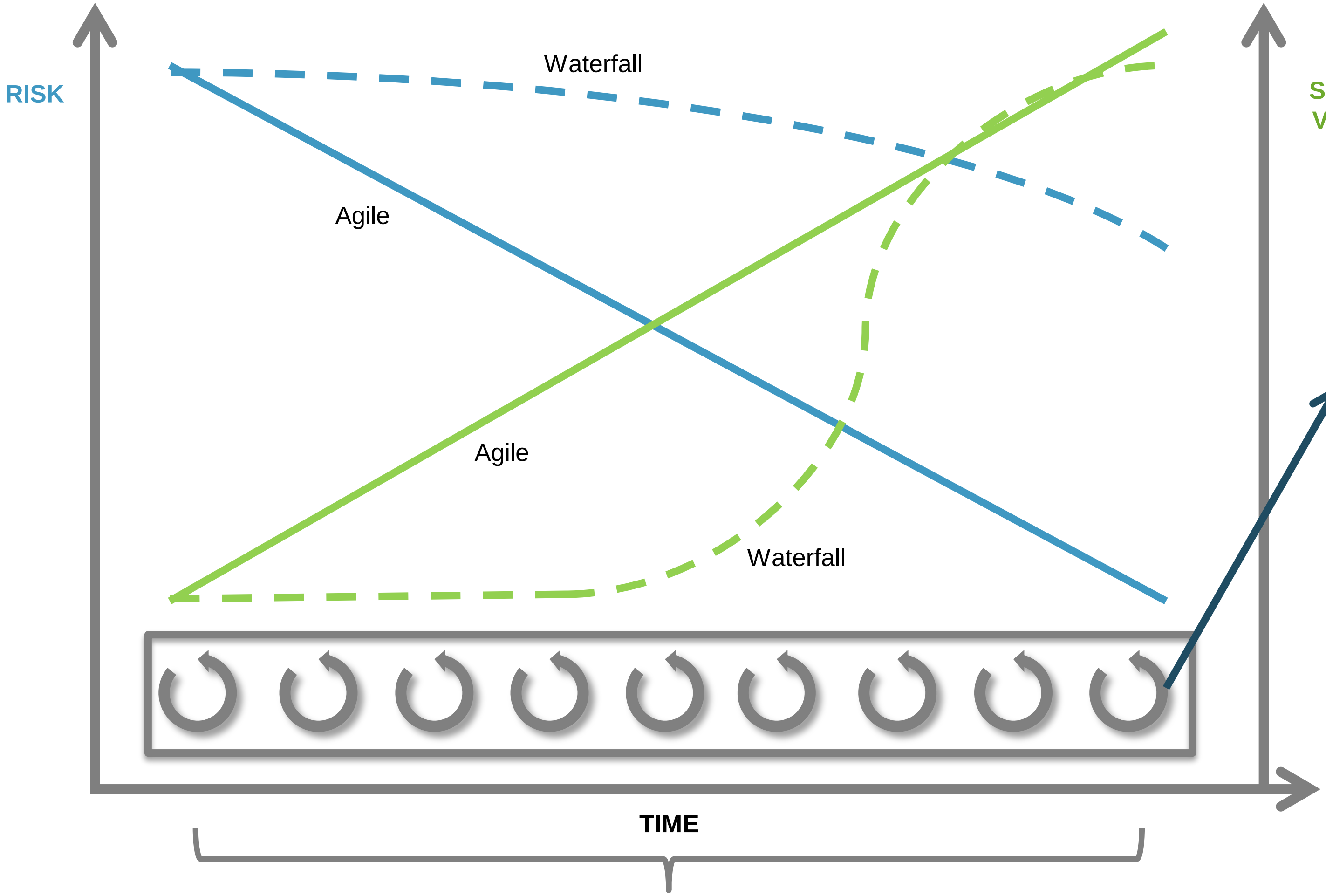
- **Continuous Integration (CI)** - A development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early [Fowler, 2006].

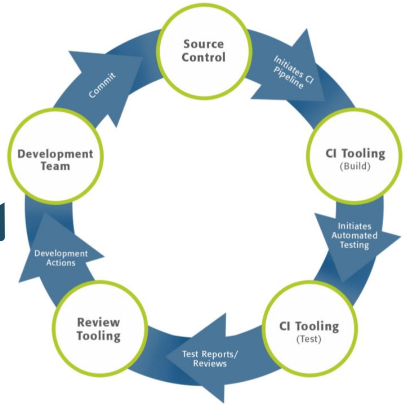
- **Keys**:
  - Maintain a single source repository
  - Automate the build
  - Make your build self-testing
  - Everyone commits to the mainline every day*
  - Every commit should build the mainline on an integration machine*
  - Keep the build fast
  - Test in a clone of the production environment
  - 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



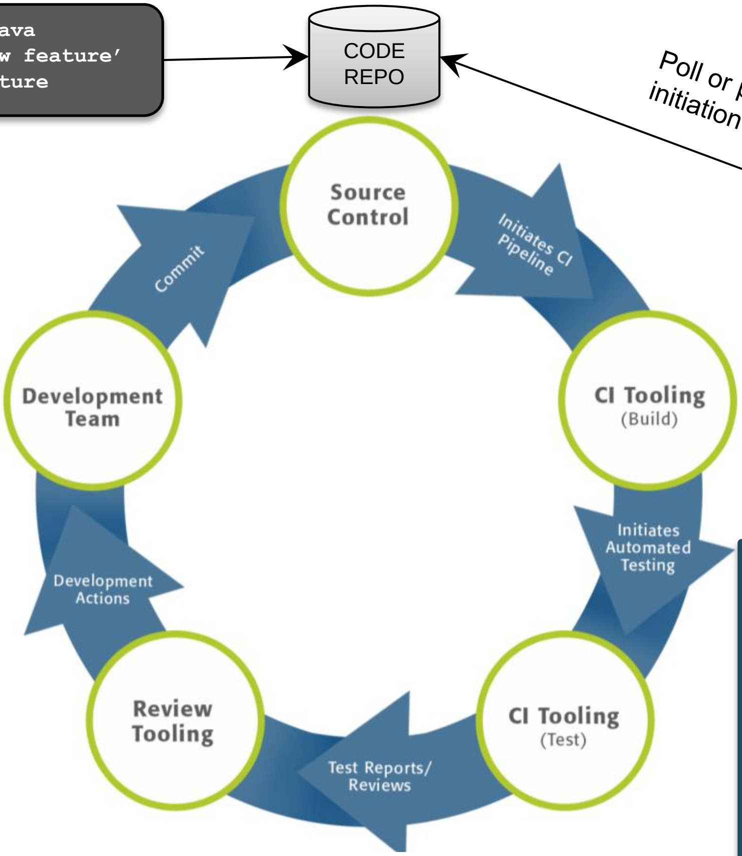


Within increments, each run of our automated CI pipeline can / should involve validation and verification activities

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

Continuous Verification and Validation Opportunities

```
git add some_new_file.java
git commit –am ‘some new feature’
git push origin new_feature
```



Poll or push-style initiation possible

Example CI Tooling

Jenkins, Travis, Bamboo, Team Foundation Server 2012 Visual Studio, TeamCity

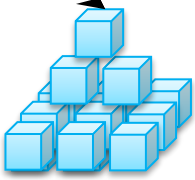
Validation

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

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

Automated Verification

Build Artifacts



Automated Test Execution

googletest Google C++ Testing Framework

JUnit

Se

Se

FEST

Static Analysis

sonarqube

FindBugs

MALPAS

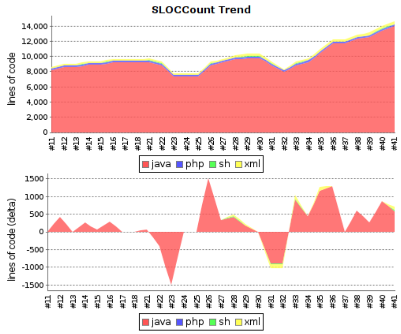
Manual Verification

Code Reviews

CRUCIBLE

Metrics / Reports

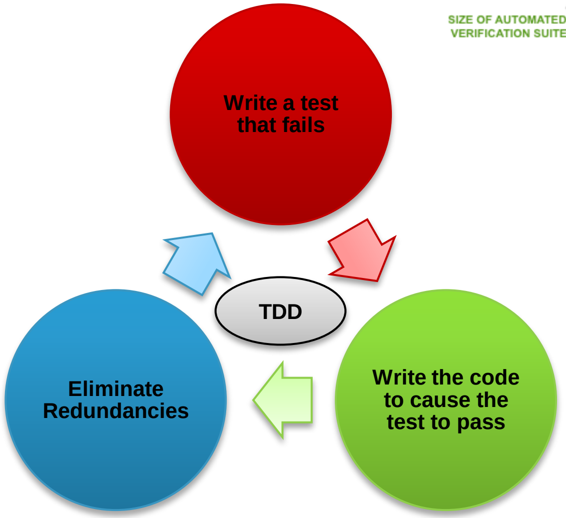
Package	# Classes	Line Coverage	Branch Coverage	Complexity
hudson.tools	6	80%	81%	0
hudson.views	15	75%	N/A	0
hudson.search	28	67%	60%	1.421
hudson.scheduler	7	67%	60%	2.294
hudson.node_monitors	28	61%	36%	0
hudson.slaves	50	56%	27%	1.6
hudson.model.listeners	5	54%	46%	1.667
hudson.triggers	18	52%	37%	1.5
hudson.widgets	5	50%	17%	1
hudson.matrix	27	50%	32%	1
hudson	125	49%	44%	2.845
hudson.model	303	48%	38%	1.66
hudson.tasks	68	46%	32%	1
hudson.security	84	45%	28%	1.724
hudson.tasks.junit	17	45%	36%	3
hudson.maven	76	43%	32%	1.486
hudson.util	167	35%	23%	1.832
hudson.scm	83	30%	20%	3
hudson.util.spring	10	34%	24%	1.238
hudson.diagnosis	5	31%	7%	0
hudson.maven.reporters	37	20%	13%	2.182
hudson.tasks.test	15	18%	6%	0
hudson.scm.browsers	20	8%	2%	2
hudson.logging	6	8%	4%	0
hudson.os.solaris	16	6%	3%	0
hudson.os.windows	4	5%	0%	0
hudson.lifecycle	11	2%	2%	0
hudson.util.jna	20	0%	0%	1
hudson.tasks.labelers	5	0%	0%	0
hudson.org.apache.tools.ant.taskdefs.cvslib	8	0%	0%	1.867
hudson.org.apache.tools.ant.taskdefs	1	0%	0%	0
hudson.fsp	5	0%	0%	0



noblis

TDD / ATDD / BDD – Preemptive Verification and Validation

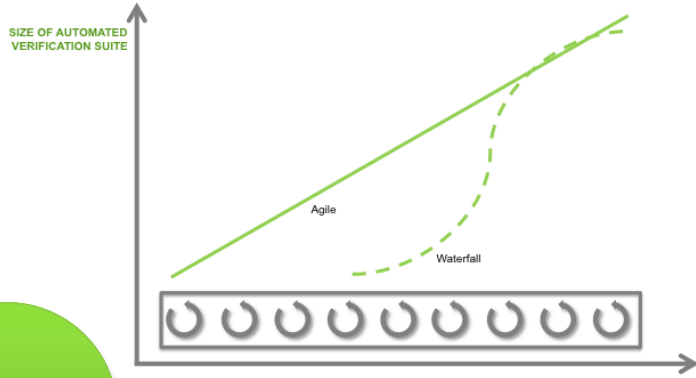
- **Test Driven Development (TDD)**
  - Write tests first and establish failures
  - Write minimal functionality to allow tests to pass
  - Facilitates incremental, continuous growth of test suite
- **Acceptance Test Driven Development (ATDD)**
  - Emphasizes developer-tester-business customer collaboration
    - 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


Write a test that fails

TDD

Write the code to cause the test to pass

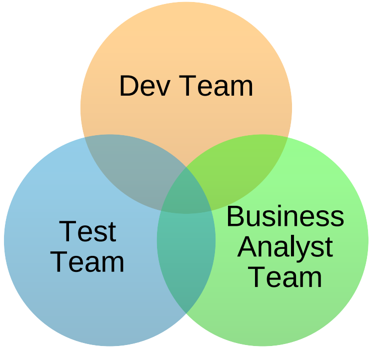
Eliminate Redundancies



Cucumber

specflow

Cucumber for .NET

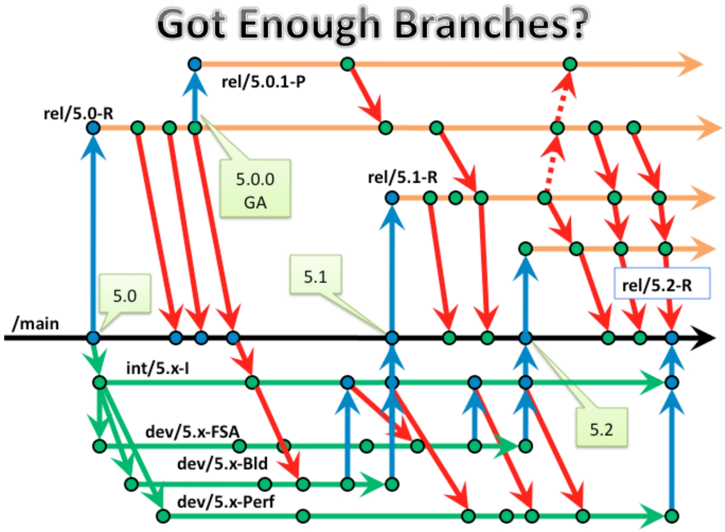

Dev Team

Test Team

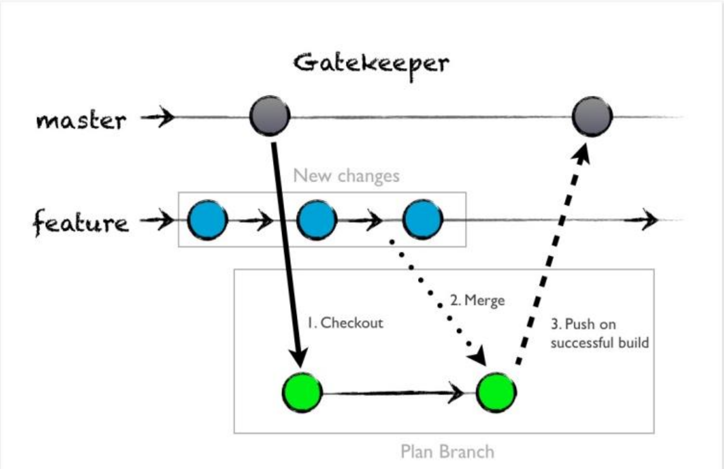
Business Analyst Team

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#Usingplanbra ches-Gatekeeper

Mission Criticality and Provable Correctness

Brooks

“No Silver Bullet”

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

"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”’

Dijkstra

- 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

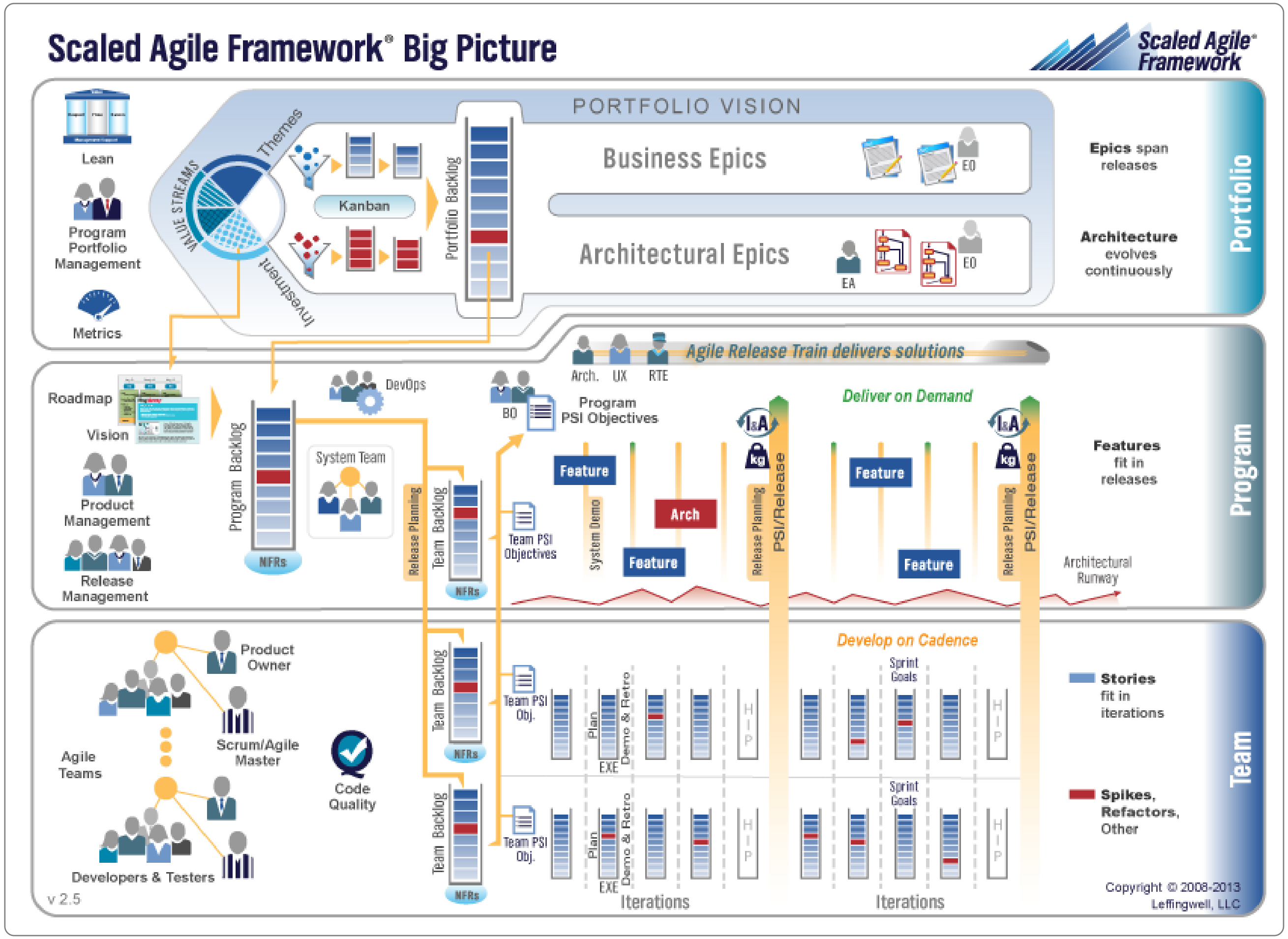
- Formal methods lite
- Run-time verification
- Automated test generation from formal specifications

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

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

Increasing level of Criticality

Scaling Agile – Scaled Agile Framework (SAFe)



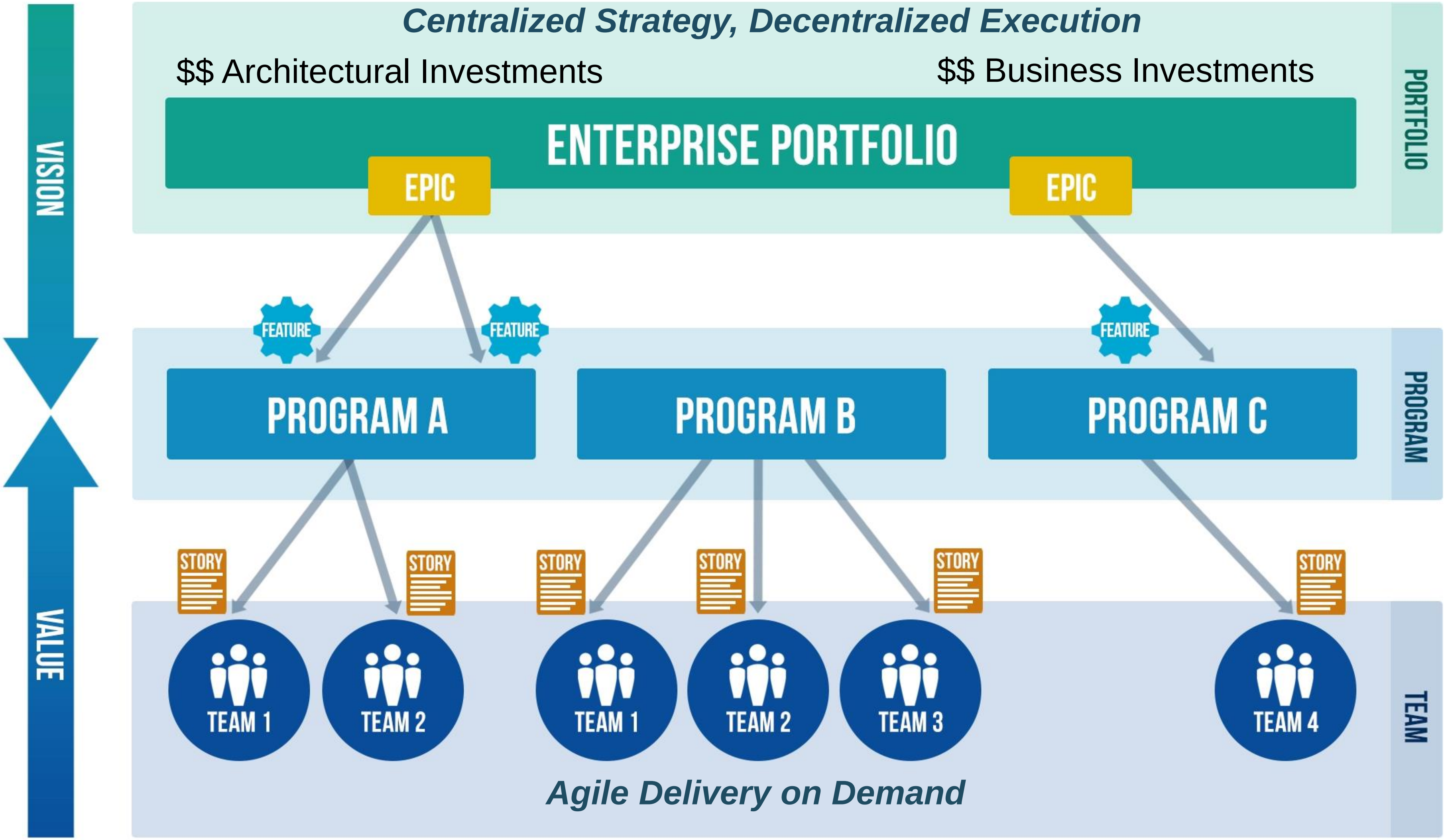
http://ScaledAgileFramework.com

- 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:

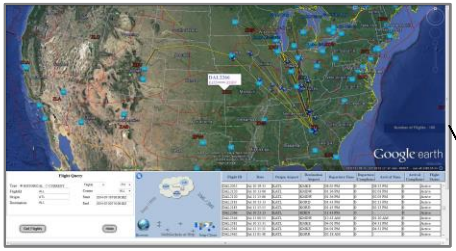
1. **Code Quality**
2. **Program Execution**
3. **Alignment**
4. **Transparency**

A Scaled Agile Portfolio Approach ≠ "Top Down"

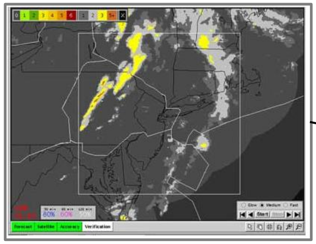


How Scaled Agile Approaches Relate to Complex Systems

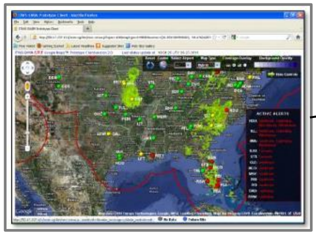
An Oversimplified Hypothetical Related to SWIMS



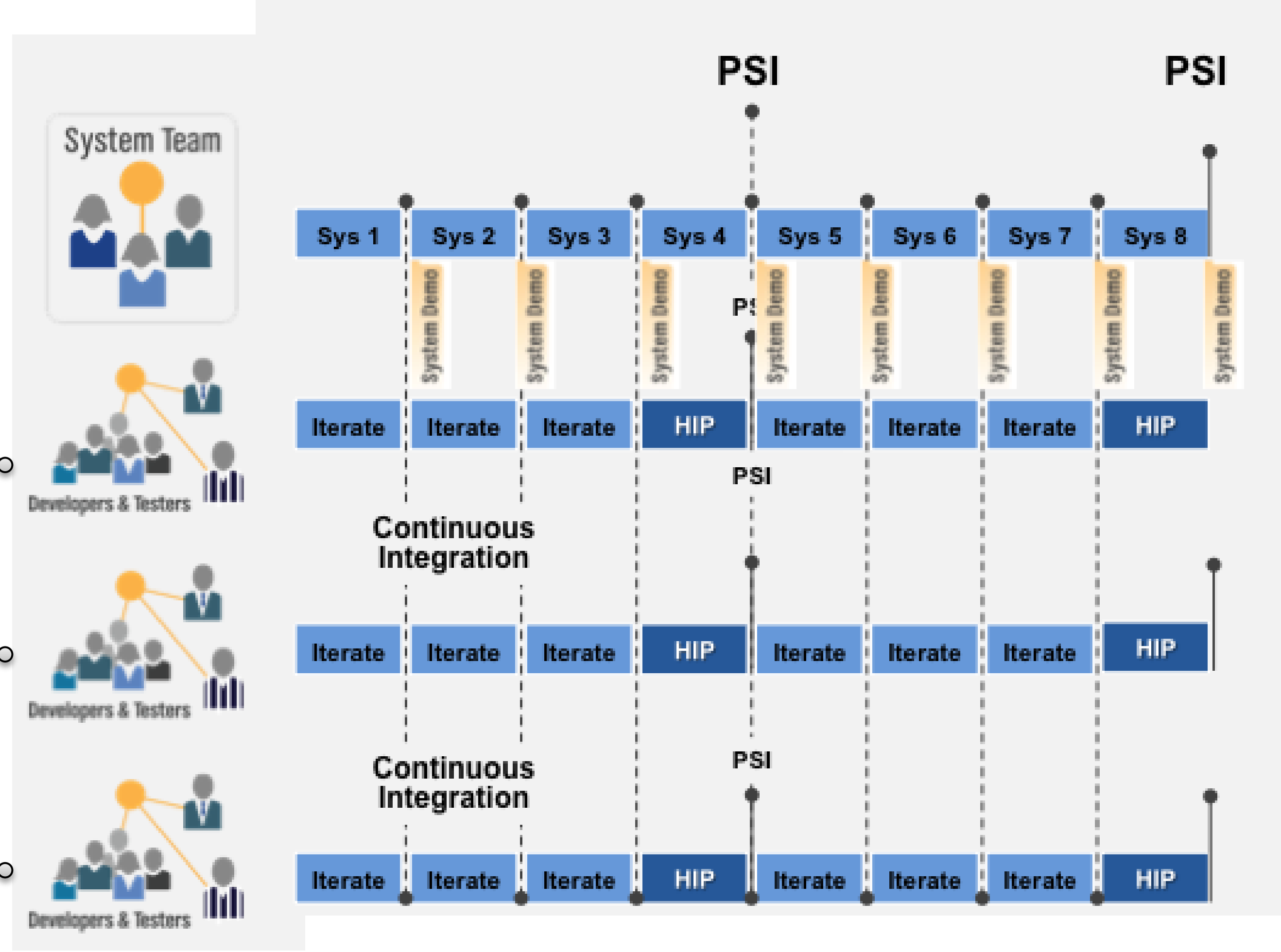
SFDPS



CIWS

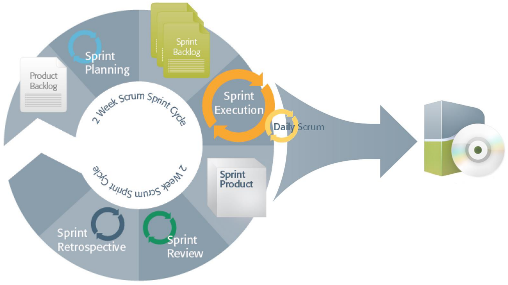


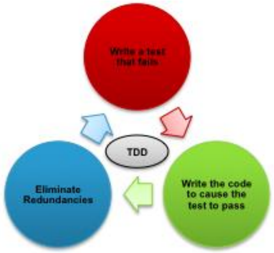
ITWS

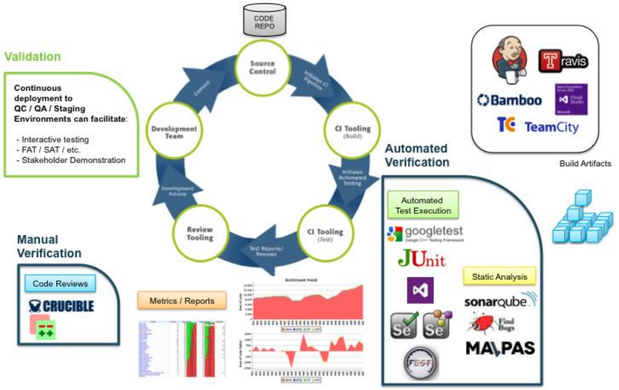


This probably isn’t news…

- 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.”

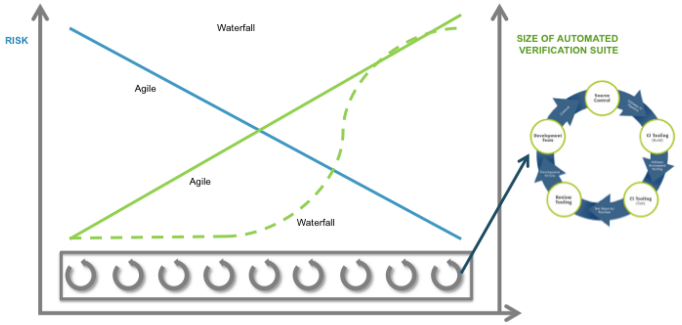


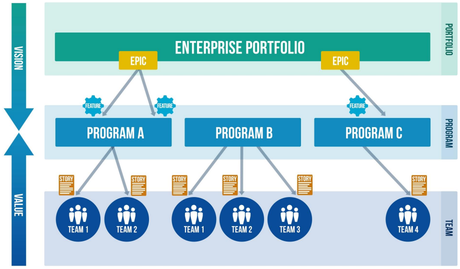


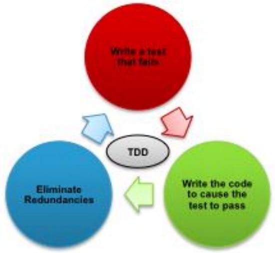


This probably isn't news…

- 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."





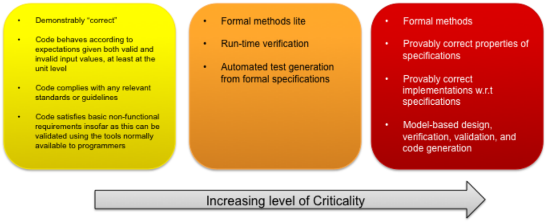


Cucumber

specflow

Cucumber for .NET

Dev Team
Test Team
Business Analyst Team



- 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

Increasing level of Criticality

References & Resources

- Effective Practices and Federal Challenges in Applying Agile Methods, GAO (27 Jul 2012). http://www.gao.gov/products/GAO-12-681

- 25 Point Implementation Plan to Reform Federal IT Management, OMB (09 Dec 2010). https://cio.gov/wp-content/uploads/downloads/2012/09/25-Point-Implementation-Plan-to-Reform-Federal-IT.pdf

- Planning for Success: Agile Software Development in Federal Agencies, ACT-IAC (Aug 2013). https://actiac.org/sites/default/files/Agile%20Software%20Development%20in%20Federal%20Agencies%20-%20ET%20SIG%2008-2013.pdf

- An Introduction to Scrum, Mountain Goat Software LLC (06 June 2014). http://www.mountaingoatsoftware.com/agile/scrum/a-reusable-scrum-presentation

- How the FBI Proves Agile Works for Government Agencies, CIO.com, Jason Bloomberg (22 Aug 2012). http://www.cio.com/article/2392970/agile-development/how-the-fbi-proves-agile-works-for-government-agencies.html

- Delivering Provably Correct Code, Dave Nicollete (23 Oct 2012) http://davenicolette.wordpress.com/2012/10/23/delivering-provably-correct-code/

- Can New Software Testing Frameworks Bring Us to Provably Correct Software?, Matthew Heusser (13 Feb 2013), http://www.cio.com/article/2388410/agile-development/can-new-software-testing-frameworks-bring-us-to-provably-correct-software-.html

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

- Software Development Process, Wikipedia, http://en.wikipedia.org/wiki/Software_development_process

- Cucumber – Making BDD Fun, http://cukes.info/

- Federal Aviation Administration System Wide Information Management (SWIM) Program Overview and Status Update, Jim Robb (24 April 2014), http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/atc_comms_services/swim/documentation/media/briefings/Day_2_SWIM_Jim_Robb-Final.pdf

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

noblis®
For the best of reasons