

An Innovative Approach to Lower the Risk of Software Intensive Development Programs

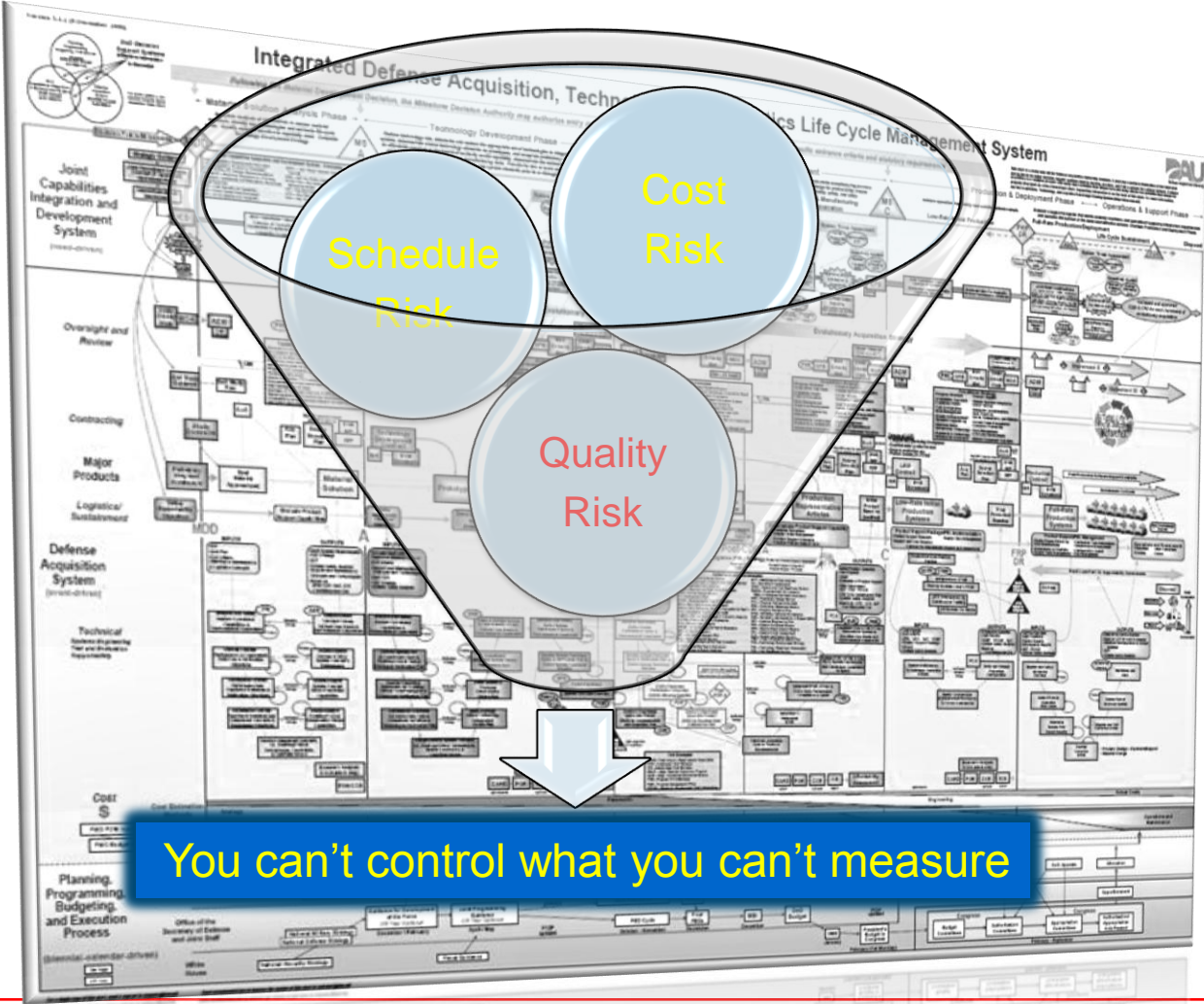
Jeff Dunlap

Director Navy C4ISR

Jeff.dunlap@baesystems.com



Assertion: Less risk = greater probability of success



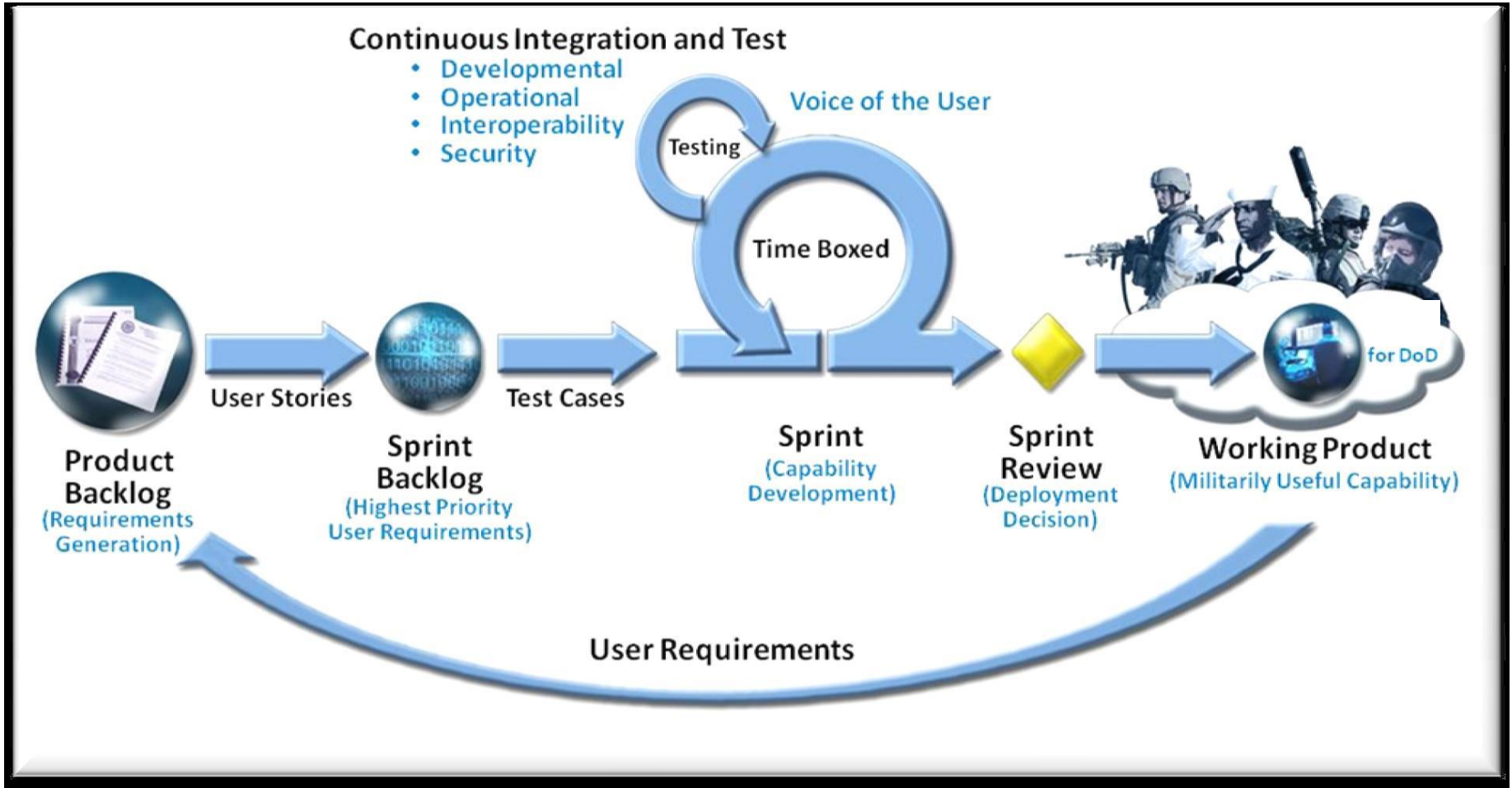
Elements needed to control risk

- ☑ constant measure of schedule achieved
- ☑ constant measure of actual cost incurred
- ☑ constant measure of quality tested

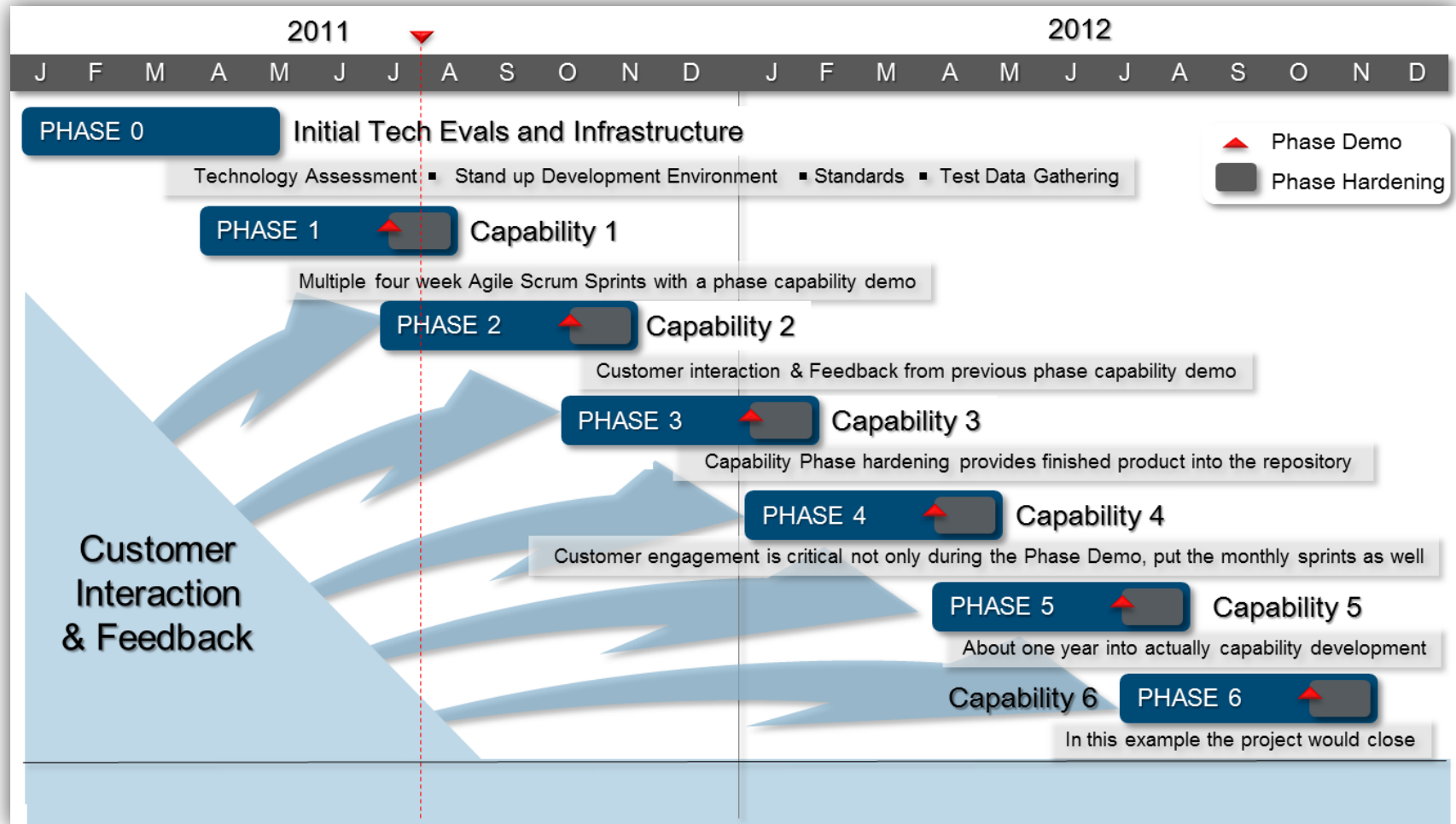
- ☑ Agile Earned Value



Agile Development



Industry example of a System Capability Roadmap



Agile four week Scrum Sprint within a capability phase

Week 1	Mon	Tues	Wed	Thrs		
Team			daily SCRUM	daily SCRUM	Off Friday	
Leads & CE	Story Prioritization		SCRUM of SCRUMS	SCRUM of SCRUMS		
Leads	Move Stories into JIRA					
		Move Tasks into JIRA				
Team	Bugs & Improvements					
	Design, Deliver, Accept - Update JIRA Status daily					
	Backlog prep for next Sprint					
All			Weekly Team Meeting			
PM	Determine Velocity / Feedback Loop to Planning		Status GreenHopper Tasks & Stories			
Week 2	Mon	Tues	Wed	Thrs		Fri
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
	Bugs & Improvements					
Team	Design, Deliver, Accept - Update JIRA Status daily					
	Backlog prep for next Sprint					
All			Weekly Team Meeting			
PM	Status GreenHopper Tasks & Stories					
Week 3	Mon	Tues	Wed	Thrs	Fri	
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	Off Friday	
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS		
	Bugs & Improvements					
Team	Design, Deliver, Accept - Update JIRA Status daily					
	Backlog prep for next Sprint					
All			Weekly Team Meeting			
PM	Status GreenHopper Tasks & Stories					
Week 4	Mon	Tues	Wed	Thrs		Fri
Team	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM		
Leads & CE	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS		
Leads & CE	Propose Next-Sprint Stories, Review Backlog					
	Bugs & Improvements					
	Design, Deliver, Accept - Update JIRA Status daily			JIRA Close-out		
	Backlog prep for next Sprint			Sprint Release / Design Review / Demo		
			Weekly Team Meeting	Retrospective		
	Status GreenHopper Tasks & Stories			Run Final Reports / Update Velocity		

Constant measure of schedule achieved

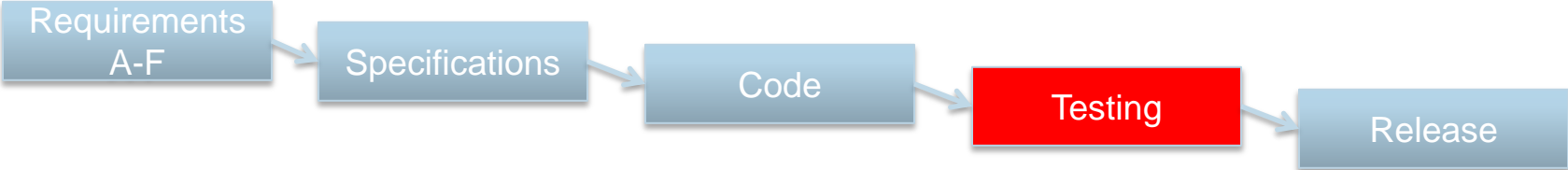
Constant measure of actual costs incurred

Measure of quality tested

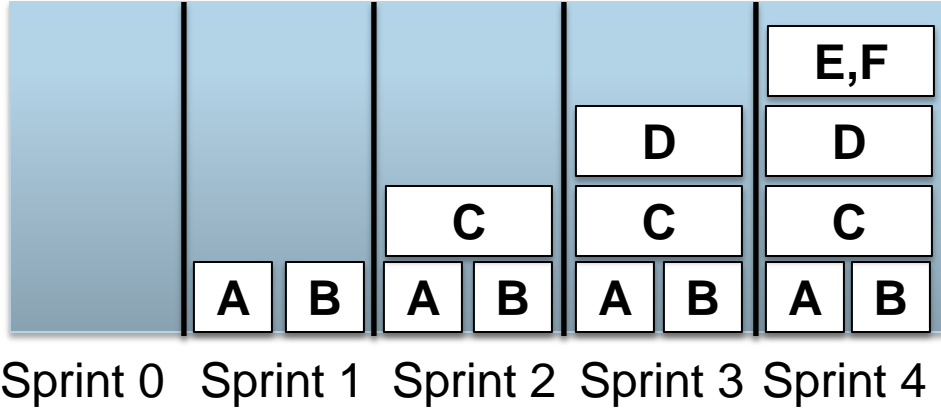


Traditional vs. Agile Testing

Traditional

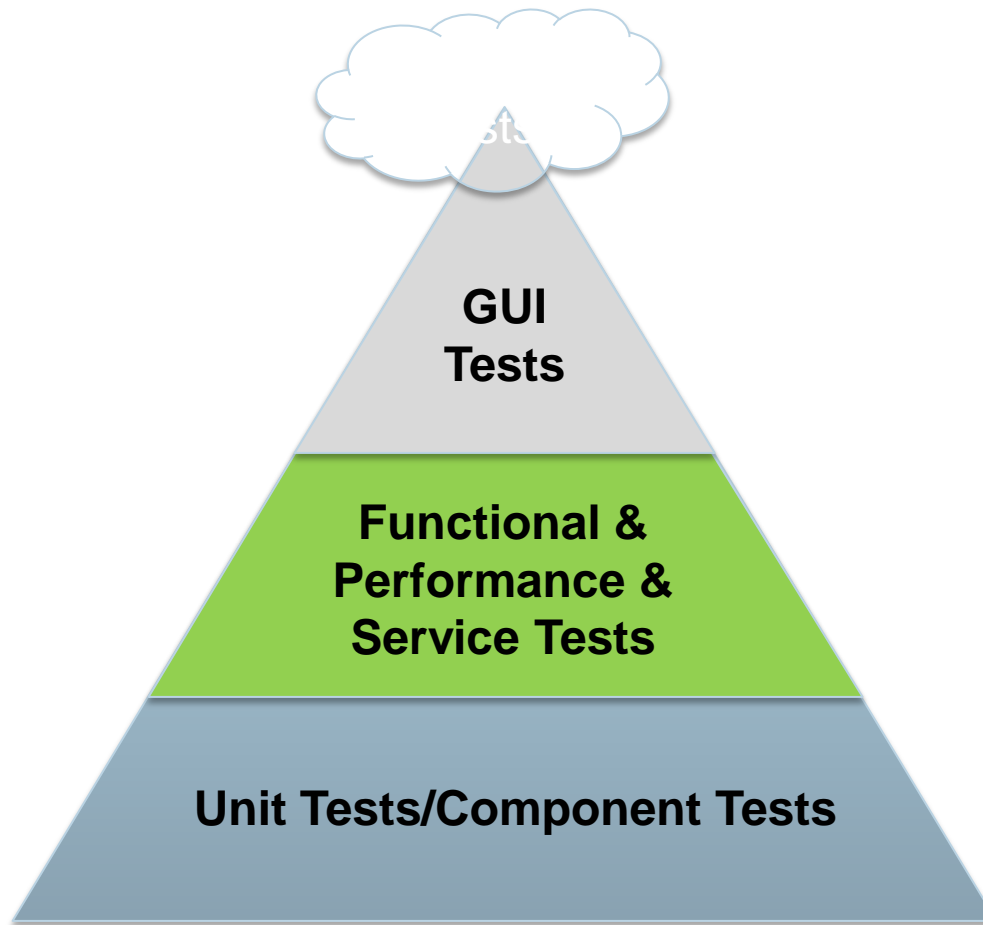


Agile



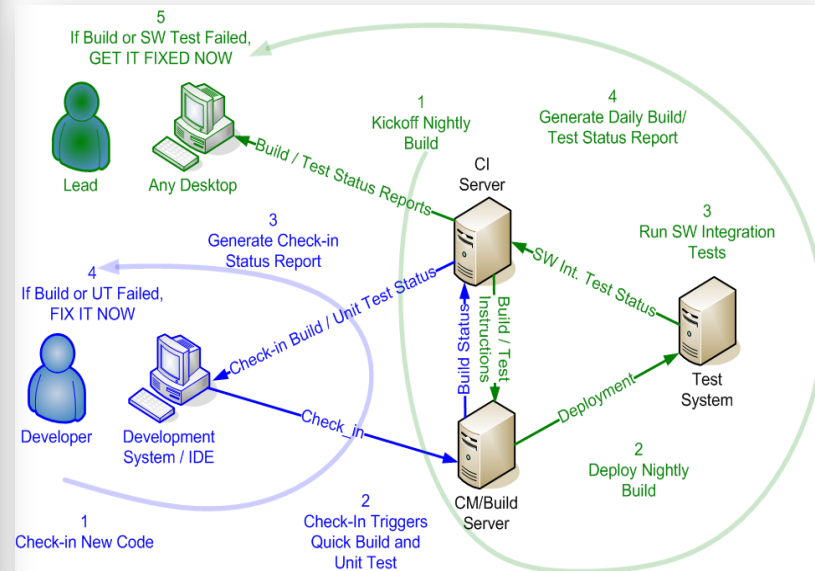
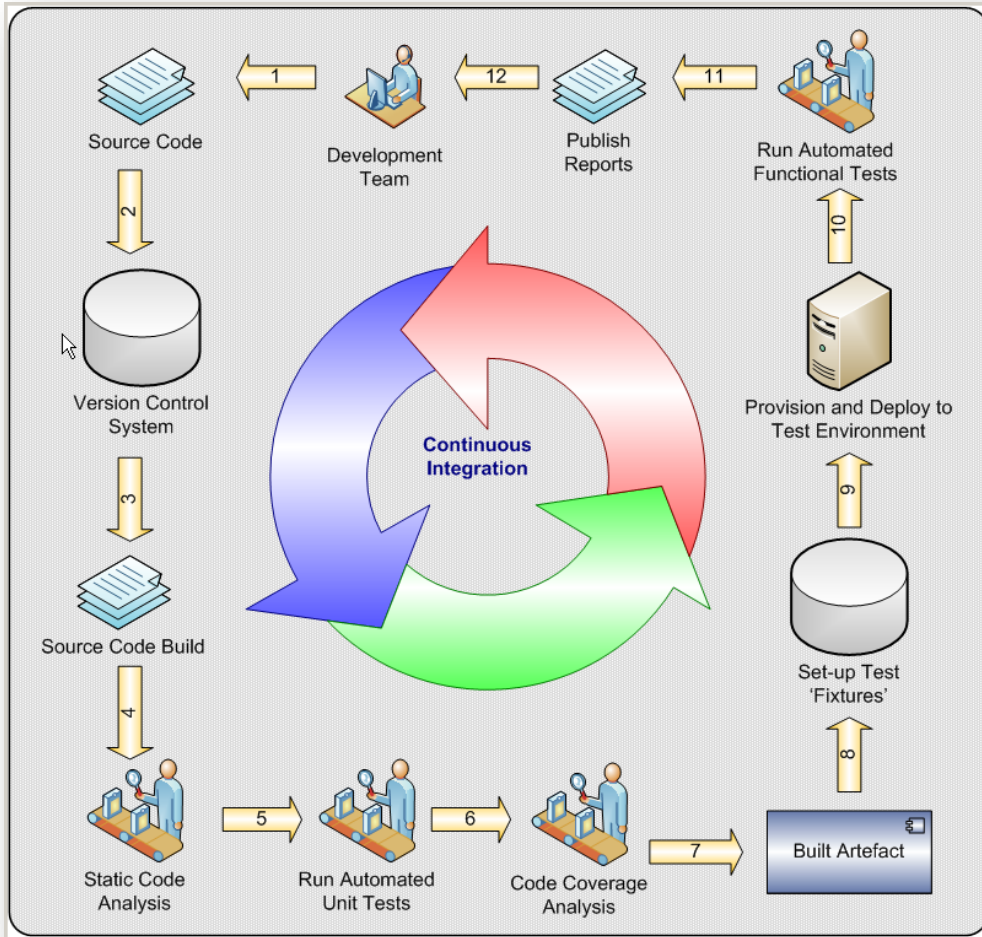
Agile produces working, tested, and deployable software sooner

Test Automation Pyramid



Goal is to achieve 100% automation at the Unit Level – where the best ROI occurs

Continuous integration and test



Continuous Integration and Testing environment using COTS, Open Source CI tool

The screenshot shows the Jenkins web interface. On the left, there are navigation links like 'New Job', 'Manage Jenkins', 'People', 'Build History', etc. The main area displays a table of build jobs. A blue box at the bottom of the table contains the text 'Constant measure of quality tested'.

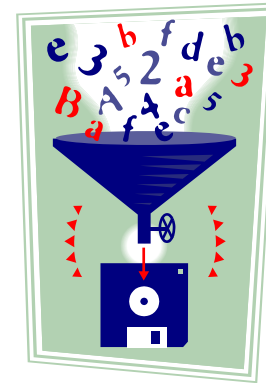
All	Branch-2.0.2.a	Clean	Code-Freeze	Components	Master Installer	Master Test	PS	WPS	Web Client	z-OLD-Branch-2.0.1.x	z-OLD-Branch-2.0.1.y	z-OLD-Branch-2.0.1.z	+
S	W	Job ↓	Last Success	Last Failure	Last Duration								
🌧️	🌧️	branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z	4 mo 0 days (#80)	4 mo 0 days (#79)	1 hr 56 min								
🌧️	🌧️	branch-2.0.1-code-freeze-build-and-test-deploy-2.0.1.z-JUPITER	4 mo 4 days (#71)	4 mo 5 days (#70)	3 hr 14 min								
🌧️	☀️	branch-2.0.1-components-dist-WinXP32-2.0.1.x	8 mo 20 days (#170)	8 mo 22 days (#167)	2 hr 0 min								
🌧️	☀️	branch-2.0.1-components-dist-WinXP32-2.0.1.y	7 mo 21 days (#40)	8 mo 1 day (#36)	2 hr 3 min								
🌧️	☀️	branch-2.0.1-products-dist-Win7-2.0.1.x	8 mo 19 days (#202)	N/A	31 min								
🌧️	☀️	branch-2.0.1-products-dist-Win7-2.0.1.y	7 mo 20 days (#55)	7 mo 28 days (#51)	31 min								
🌧️	☀️	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.x	8 mo 25 days (#58)	N/A	6 min 26 sec								
🌧️	☀️	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.y	7 mo 27 days (#19)	8 mo 3 days (#11)	6 min 40 sec								
🌧️	☀️	branch-2.0.1-products-InstallWizard-WinXP-2.0.1.z	4 mo 1 day (#30)	N/A	7 min 29 sec								
🌧️	☀️	branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.x	8 mo 19 days (#189)	8 mo 24 days (#184)	1 hr 25 min								
🌧️	☀️	branch-2.0.1-products-MasterInstaller-install-intTest-webTest-asService-WinXP32-2.0.1.y	7 mo 20 days (#39)	8 mo 0 days (#34)	2 hr 2 min								
🟢	🌧️	code-freeze-build-and-test-2.0.2.a	18 hr (#263)	6 days 22 hr (#259)	2 hr 42 min								
🟢	☀️	components-dist-2.0.2.a	5 days 19 hr (#161)	23 days (#145)	2 hr 10 min								
🟢	🌧️	install-release-2.0.1.20110303-Demo-VM-WinXP32	1 yr 1 mo (#35)	1 yr 1 mo (#34)	1 hr 13 min								
🟢	🌧️	install-release-Demo-Export	1 mo 11 days (#35)	1 mo 11 days (#34)	1 hr 19 min								
🟢	☀️	products-dist-2.0.2.a	18 hr (#181)	8 days 19 hr (#174)	35 min								
🟢	☀️	products-InstallWizard-2.0.2.a	11 hr (#198)	5 days 2 hr (#191)	1 min 46 sec								
🟢	☀️	products-MasterInstaller-install-asService-webTest-BRANCH-2.0.2.a	17 hr (#217)	N/A	57 min								
🟢	☀️	trunk-build-code-freeze-build-and-test	6 hr 48 min (#340)	11 hr (#339)	3 hr 39 min								
🟢	☀️			1 day 10 hr (#71)	1 hr 18 min								

Agile with Continuous Integration and Test

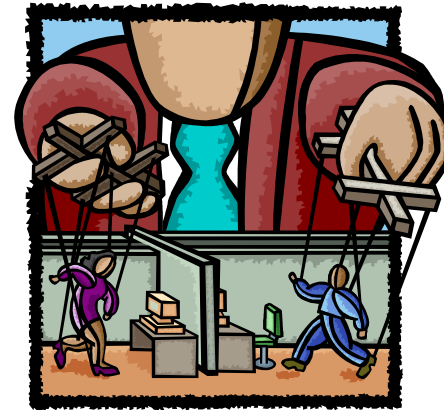
- When an capability development phase is planned, time is held “fixed” within the iterations (i.e., multiple four week Agile Scrums)
 - It becomes *obvious* at the daily meetings (both by CI&T automated results and discussions with the programmers) where the difficulties are occurring
 - Peer / Team relationships foster a culture of feedback and improvement
 - Schedule adherence or deviations is near real-time
- Actual cost are accounted for daily and reviewed weekly
 - Actual costs rise and fall proportionally to the degree of difficulty difference from the “Planned” tasks
- Quality of the software under development is monitored nightly by CI&T
 - Metrics collected show trends and areas of concern
 - Decisions can be made with insight about high risk areas

Earned Value Management conundrum with Agile

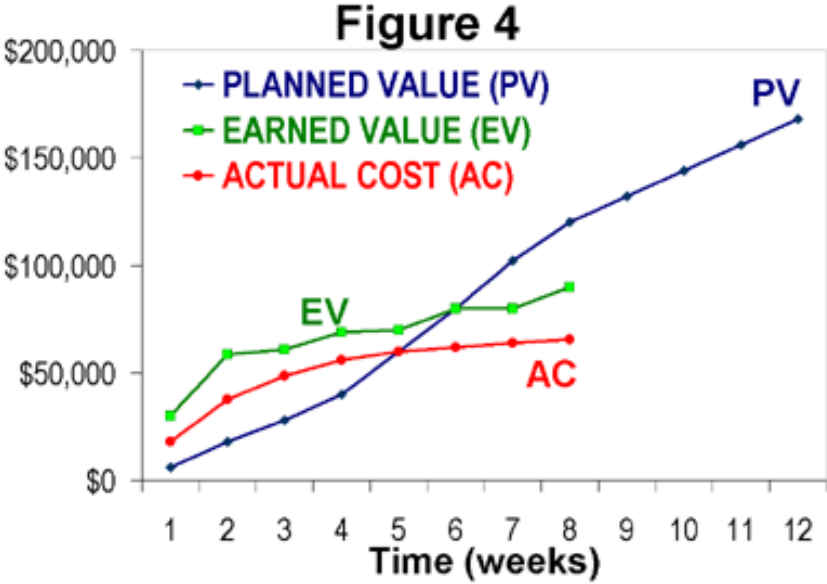
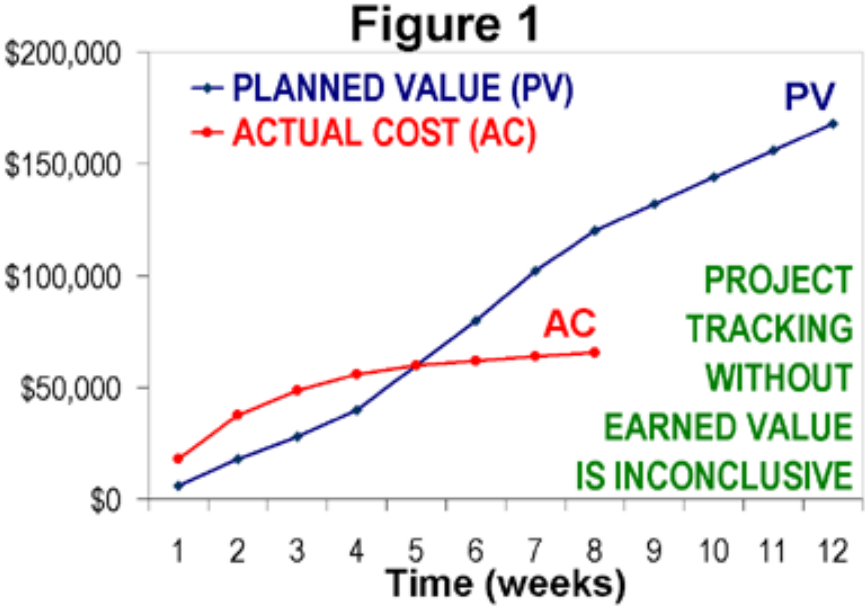
- Because EVM requires quantification of a project plan, it is often perceived to be inapplicable for Agile software development projects



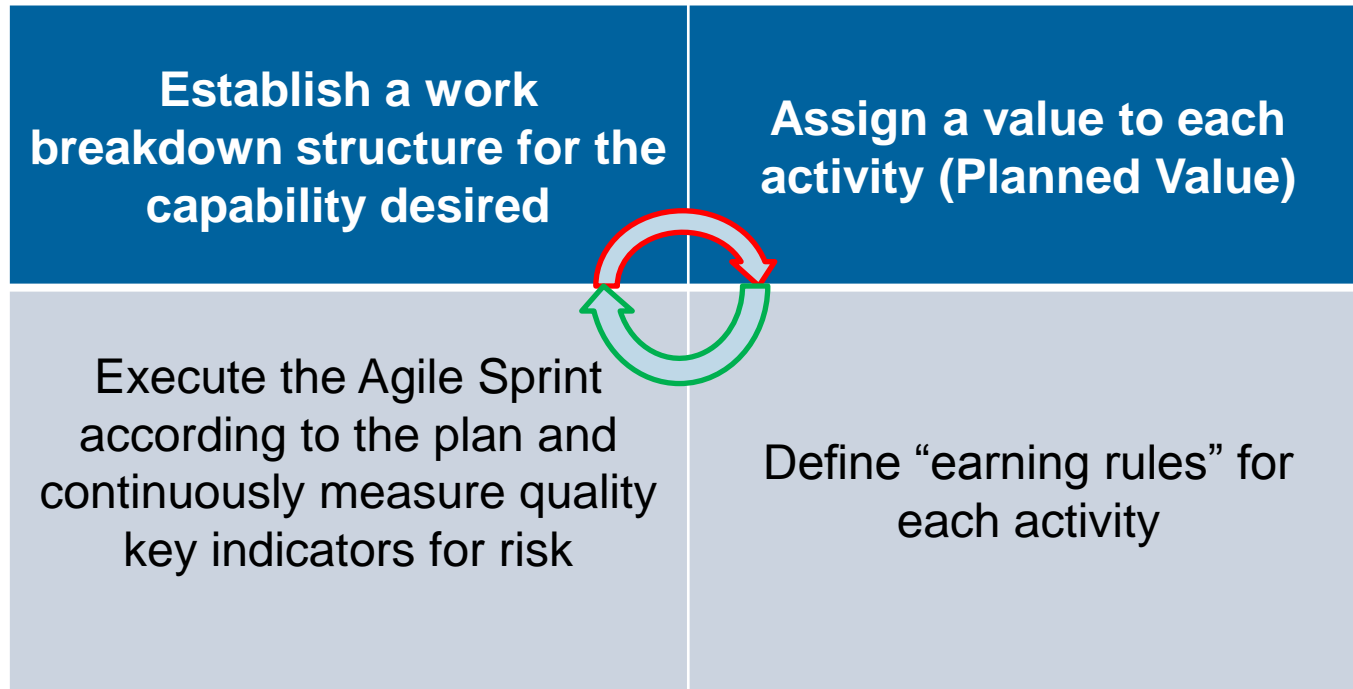
- However, another school of thought holds that all work can be planned, even if in weekly time boxes or other short increments



With EV the status is inconclusive



Agile EV



True understanding of cost and schedule performance *relies first on measuring quality objectively*

Risk becomes visible

$$PVR = PVi_1 + PVi_2 + \dots + PVi_n$$

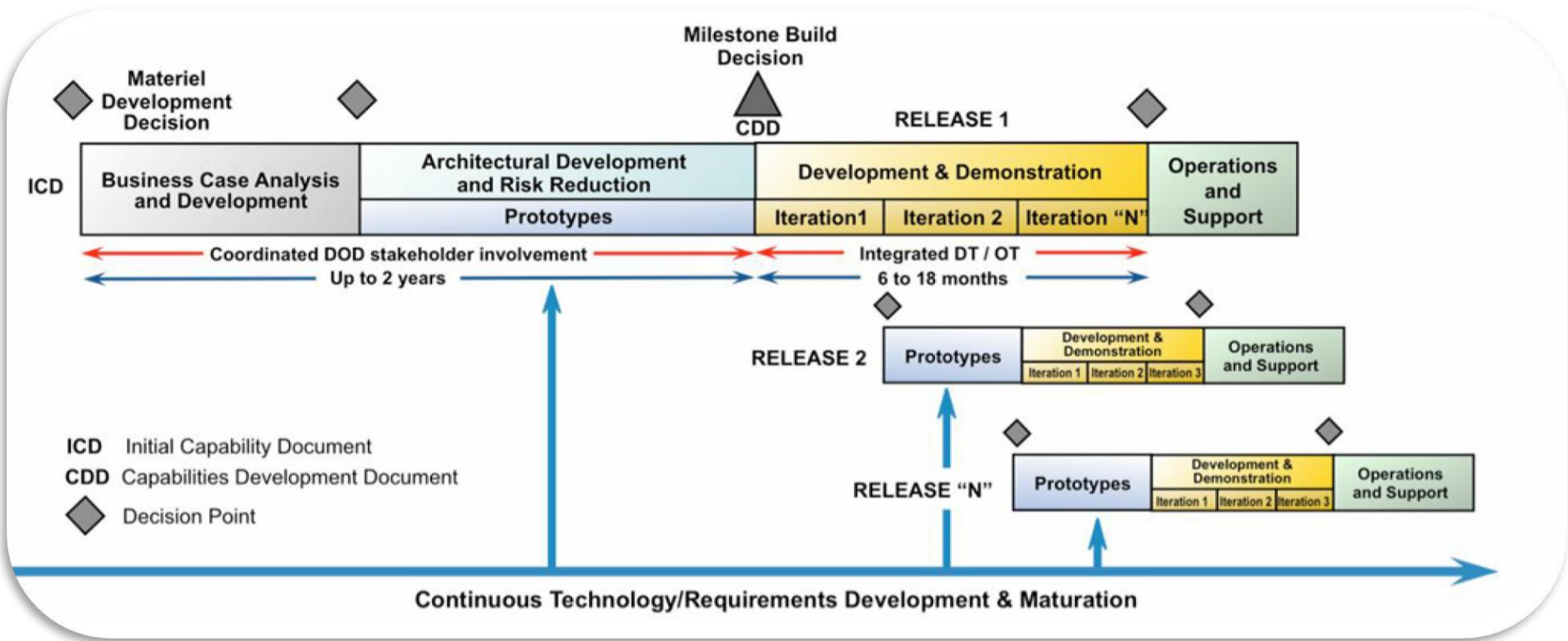
Planned Value of the release (PVR) is maintained constant (constrained cost and time) by allowing PVi to be modified based on efficiencies, enlightenment, or risks realized/avoided (both pos / neg)

Since agile EV is the measure of schedule and cost adherence to the PVi, software intensive risks becomes visible based upon the *degree of difficulty difference between the PVi and EVa*

$$EVa = \sum_{start}^{current} PVi$$

Change is coming

FY2010 NDAA Section 804



The Next Sprint has already started

- ☑ constant measure of schedule achieved
 - ☑ constant measure of actual cost incurred
 - ☑ constant measure of quality tested
-
- ☑ Agile Earned Value



Risk is actionable with Agile processes

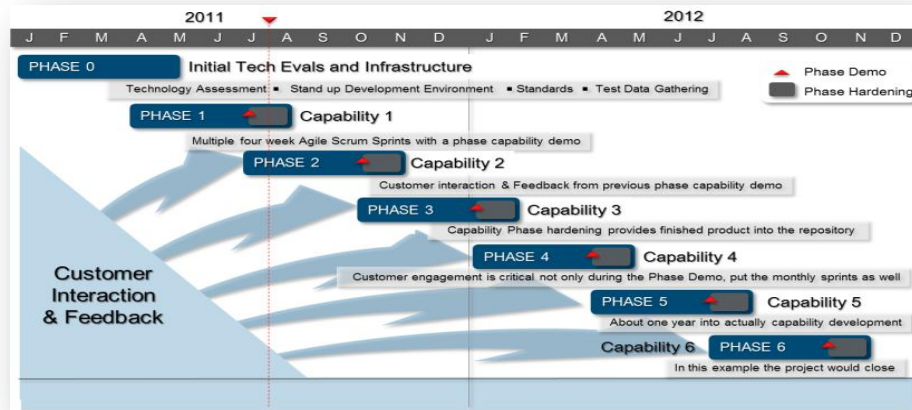
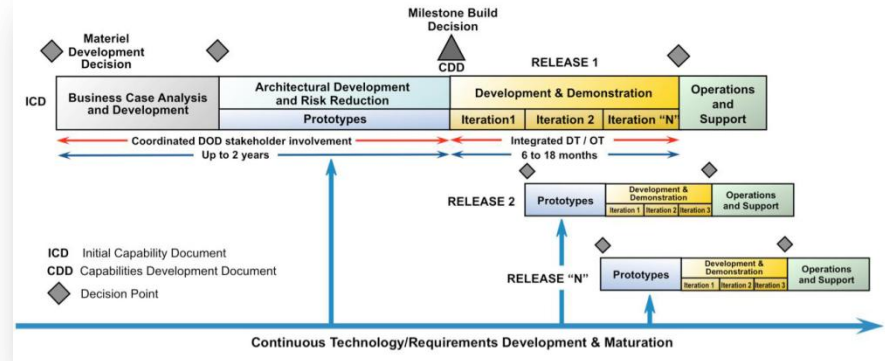
Innovation comes with agile EVMS

A project plan is developed that identifies work to be accomplished (work breakdown structure) both at the release level and with specific details at the iteration levels

A valuation of planned work at the release level, Planned Value “release” (PVr)

A pre-defined set of “earning rules” (metrics) to quantify the accomplishment of work called Earned Value (EV)

Prior to the start of the next capability iteration phase, the PVr remains constant (can always tell where you have been), but the Planned Value iteration (PVi) is adjustable to ensure that EVa is measuring the right items



Innovation comes with agile EVMS

Agile Scrums within an iteration phase provides actual cost information (c) and schedule (s) earnings as they are tightly coupled to the quality test event

	Week 1	Mon	Tues	Wed	Thrs	Fri
Team				daily SCRUM	daily SCRUM	
Leads & CE		Story Prioritization		SCRUM of SCRUMS	SCRUM of SCRUMS	
Leads		Move Stories into JIRA	Move Tasks into JIRA			Off Friday
Team		Bugs & Improvements		Design, Deliver, Accept - Update JIRA Status daily		
All				Backlog prep for next Sprint		
PM		Determine Velocity / Feedback Loop to Planning		Weekly Team Meeting	Status GreenHopper Tasks & Stories	
	Week 2	Mon	Tues	Wed	Thrs	Fri
Team		daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM
Leads & CE		SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS
Team		Bugs & Improvements		Design, Deliver, Accept - Update JIRA Status daily		
All				Backlog prep for next Sprint		
PM				Weekly Team Meeting	Status GreenHopper Tasks & Stories	
	Week 3	Mon	Tues	Wed	Thrs	Fri
Team		daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	
Leads & CE		SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
Team		Bugs & Improvements		Design, Deliver, Accept - Update JIRA Status daily		Off Friday
All				Backlog prep for next Sprint		
PM				Weekly Team Meeting	Status GreenHopper Tasks & Stories	
	Week 4	Mon	Tues	Wed	Thrs	
Team		daily SCRUM	daily SCRUM	daily SCRUM	daily SCRUM	
Leads & CE		SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	SCRUM of SCRUMS	
Leads & CE			Propose Next-Sprint Stories, Review Backlog			
Team		Bugs & Improvements		Design, Deliver, Accept - Update JIRA Status daily		JIRA Close-out
All				Backlog prep for next Sprint		Sprint Release / Design Review / Demo
PM				Weekly Team Meeting	Status GreenHopper Tasks & Stories	Retrospective
						Run Final Reports / Update Velocity

Continuous integration and test using automated test tools provides “continuous” monitoring of the build progress and flags areas of concern prior to the Scrum demo event (Q)

The screenshot shows the Jenkins web interface. On the left, there are navigation options like 'New Job', 'Manage Jenkins', 'Build History', and 'Build Queue'. The main area displays a list of jobs with columns for 'Job', 'Last Success', 'Last Failure', and 'Last Duration'. The jobs listed include various build and test tasks for different branches and components, such as 'branch-2.0.2-a', 'branch-2.0.1 code-freeze', and 'code-freeze build-and-test-2.0.2-a'. Each job entry includes a status icon (green for success, red for failure) and a duration.