estimate

estimate • analyze • plan • control

Beyond the Manifesto

Once you commit to an Agile Methodology, how do you measure your progress?

Gordon Kranz
Michael Thompson
ICEAA Professional Development and Training Workshop – Portland, Oregon
4 – 9 June 2017

GALORATH

Agenda

- Starting Point – The Manifesto
- Background and History
- Busting Agile Myths
- Agile Hierarchy -> WBS
- Planning Cautions
- Measuring Progress Agile/EVM
- Baseline Management
- Conclusion

DoD / NASA Agile History

http://intenseminimalism.com/2012/a-brief-history-of-agile-methods/

- 1930s — Walter Shewhart proposes a series of short "plan-do-study-act" (PDSA) cycles.
- 1950s — The X-15 hypersonic jet applied incremental and iterative development.
- 1958 — Project Mercury (NASA) software development, ran with half-day iterations. "All of us, as far as I can remember, thought waterfalling of a huge project was rather stupid, or at least ignorant of the realities."— Weinberg G. M. (Project Mercury)
- 1972 — The USS "Trident" Ohio submarine command and control system, developed by IBM FSD. More than 1 million lines of code. Four 6 month iterations.
- 1972 — Army Site Defense missile tracking software. $100 million project, developed by TRW in 5 iterations.
- 1970s — Light Airborne Multipurpose System (US Navy). 45 one-month iterations.
- "Every one of those deliveries was on time and under budget"— Mills H.

DoD / NASA Agile History

http://intenseminimalism.com/2012/a-brief-history-of-agile-methods/

"Software development should be done incrementally, in stages with continuous user participation and replanning and with design-to-cost programming within each stage."— Mills H. (1976)

- **1977-1980** — Space Shuttle (NASA) avionic software. 17 iterations over 31 months (8 weeks average).
- **1980s** — Artificial intelligence researchers used Lisp machines and evolutionary prototyping.
- **1987** — Command and Control Processing and Display System Replacement, developed by TRW in 6 time-boxed iterations.
- **1980s** — The DoD was experiencing a project failure rate of 75% in a sample of waterfall project of about $37 billion overall, where only 2% of them were used without extensive modification. At the end of 1987 the DoD changed its policies to allow iterative development.
- **1994** — The DoD was still victim of the waterfall mindset, developing too much using waterfall and so Paul Kaminsky issued a report stating: *"DoD must manage programs using iterative development"*

The Starting Point

GALORATH

Manifesto for Agile Software Development

"We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more."

Agile is NOT a Method – it's a mindset!
Individual Methods are Formal – sort-of

Principles behind the Agile Manifesto

(We Follow These Principles)

GALORATH

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile – EVM Myths / Questions

- Will Agile replace EVM?
- Agile has no standards!
- How does Agile progress roll up to EVM?

Traditional Development Programs

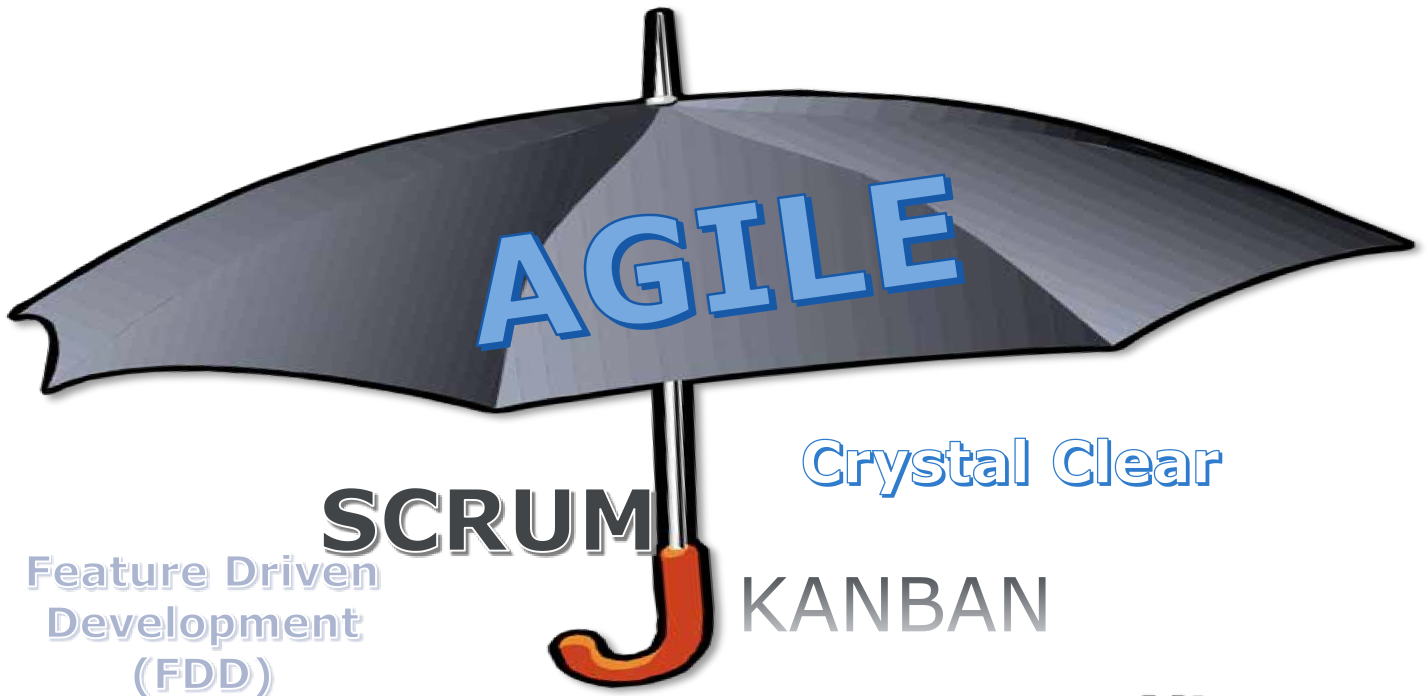
GALORATH

EVM Principle	Traditional Development Program
Decomposition of work into manageable pieces.	Mil-Std-881C WBS – Appropriate Appendix
Assignment of resources against that work.	OBS, RAM
Assigning value to work to be accomplished.	Earned Value Technique (Discrete, %complete, apportioned, LOE, QBD, etc.)
Time phasing of the work	WBS->CA->WP Hierarchy; Decomposition of WBS Dictionary
Tracking performance against technical objective criteria to claim value.	EVM Metrics: CPI, SPI, TCPI, Variance Analysis
Compare claimed value, actual costs, and planned value to support daily decision making.	Rolling Wave Planning, Formal Re-planning, Risk Management
Updating forecasts and technical plan as the team learns from history.	Monthly EACs derived from bottoms up ETC estimates

SW Development Programs (Agile)

EVM Principle	SW Development Programs (Agile)
Decomposition of work into manageable pieces.	EPIC and Feature Based WBS for SW (Product Backlog)
Assignment of resources against that work.	SW Development Teams (Sprint Teams)
Assigning value to work to be accomplished.	Business Value assigned at Feature level and above; story point values used to plan and execute the detailed work
Time phasing of the work	Roadmap->Release Planning->Sprint Planning. Priority based execution to deliver incremental capability.
Tracking performance against technical objective criteria to claim value.	Agile metrics: Velocity, burndown and burn up charts, etc. EVM Metrics: CPI, SPI, TCPI, Variance Analysis, done at feature level of above.
Compare claimed value, actual costs, and planned value to support daily decision making.	Sprint Retrospective, Story point claims, EVM % complete taken at feature level of above.
Updating forecasts and technical plan as the team learns from history.	Agile is in a constant state of planning and executing, allows for creating a forecast as often as daily.

Agile is not a single method



SCRUM

Crystal Clear

Feature Driven Development (FDD)

KANBAN

Agile Unified Process (AUP)

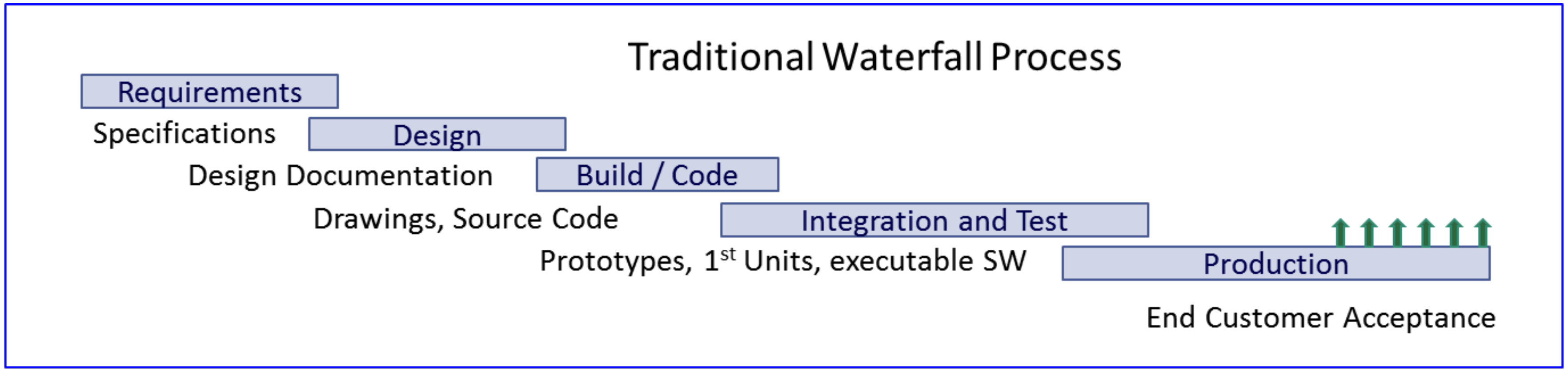
LEAN

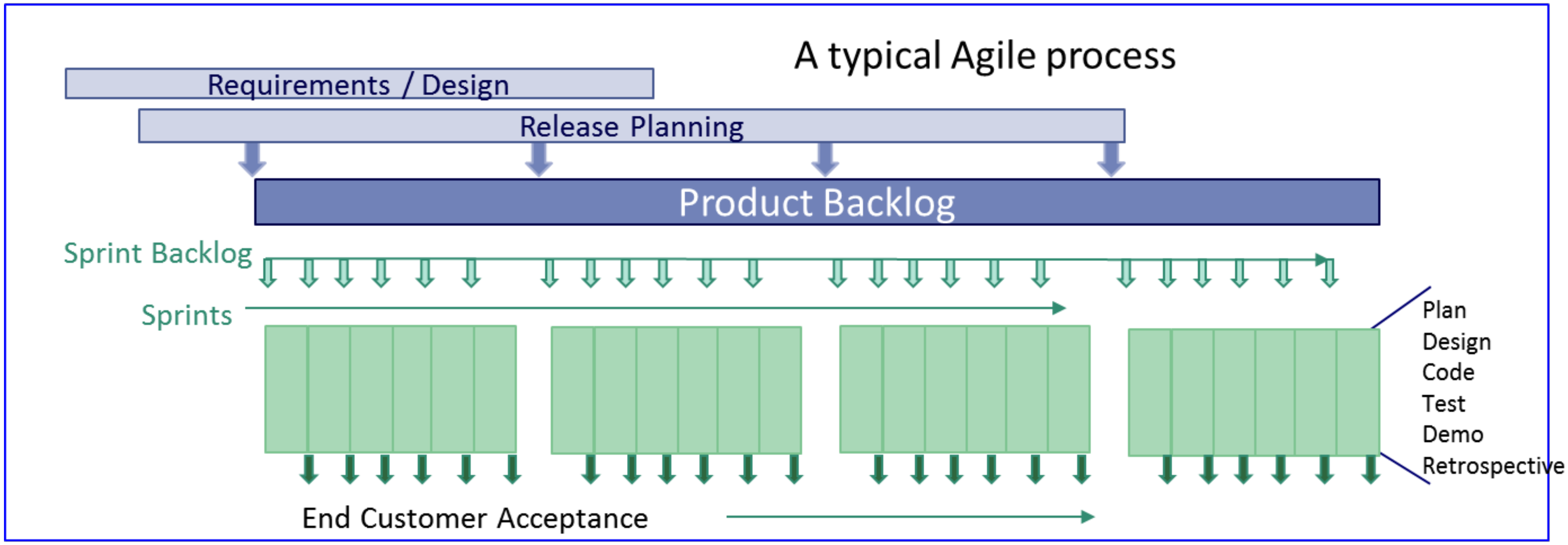
eXtreme Programming (XP)

Dynamic Systems Development Method (DSDM)

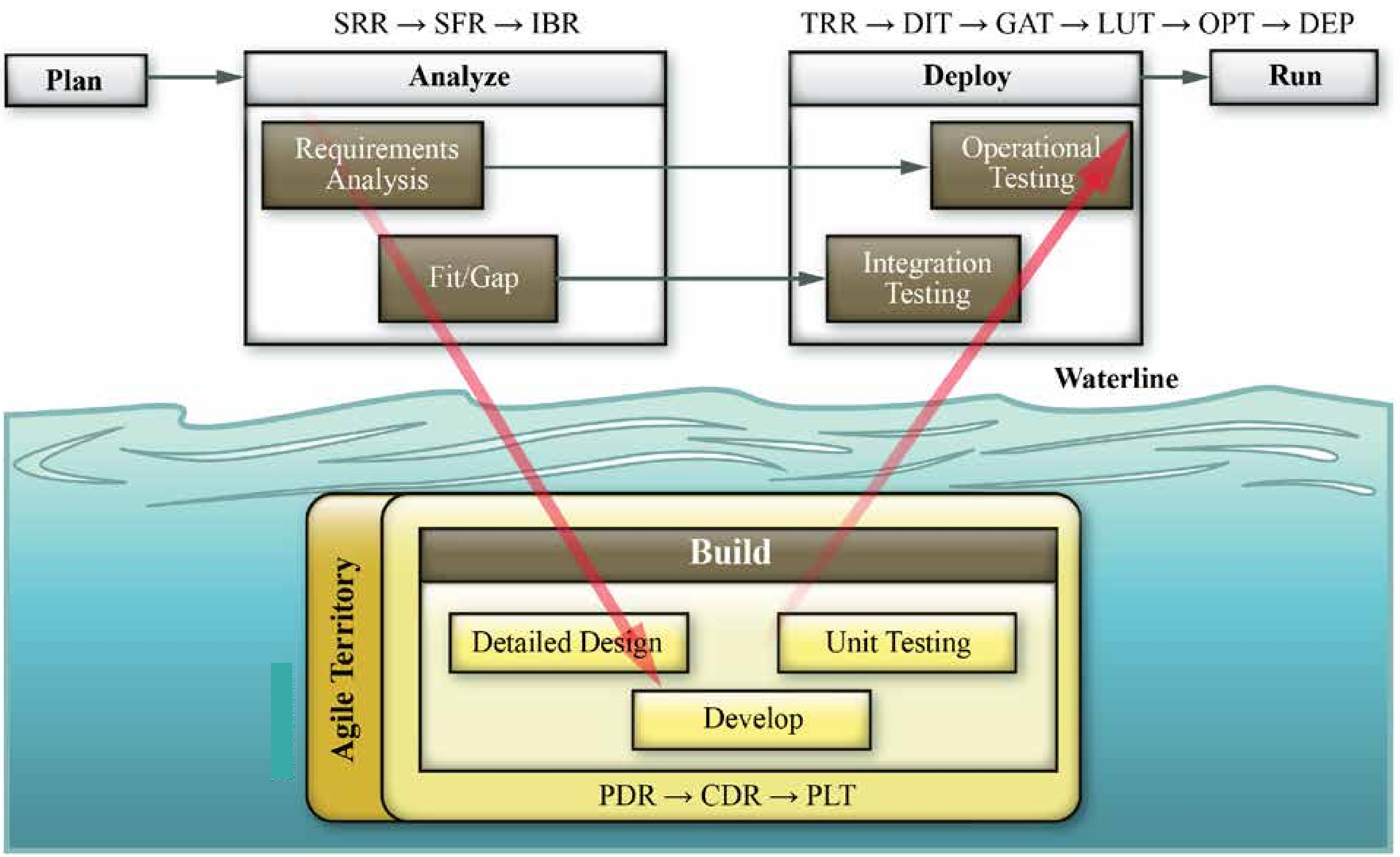
Waterfall Vice Agile

GALORATH

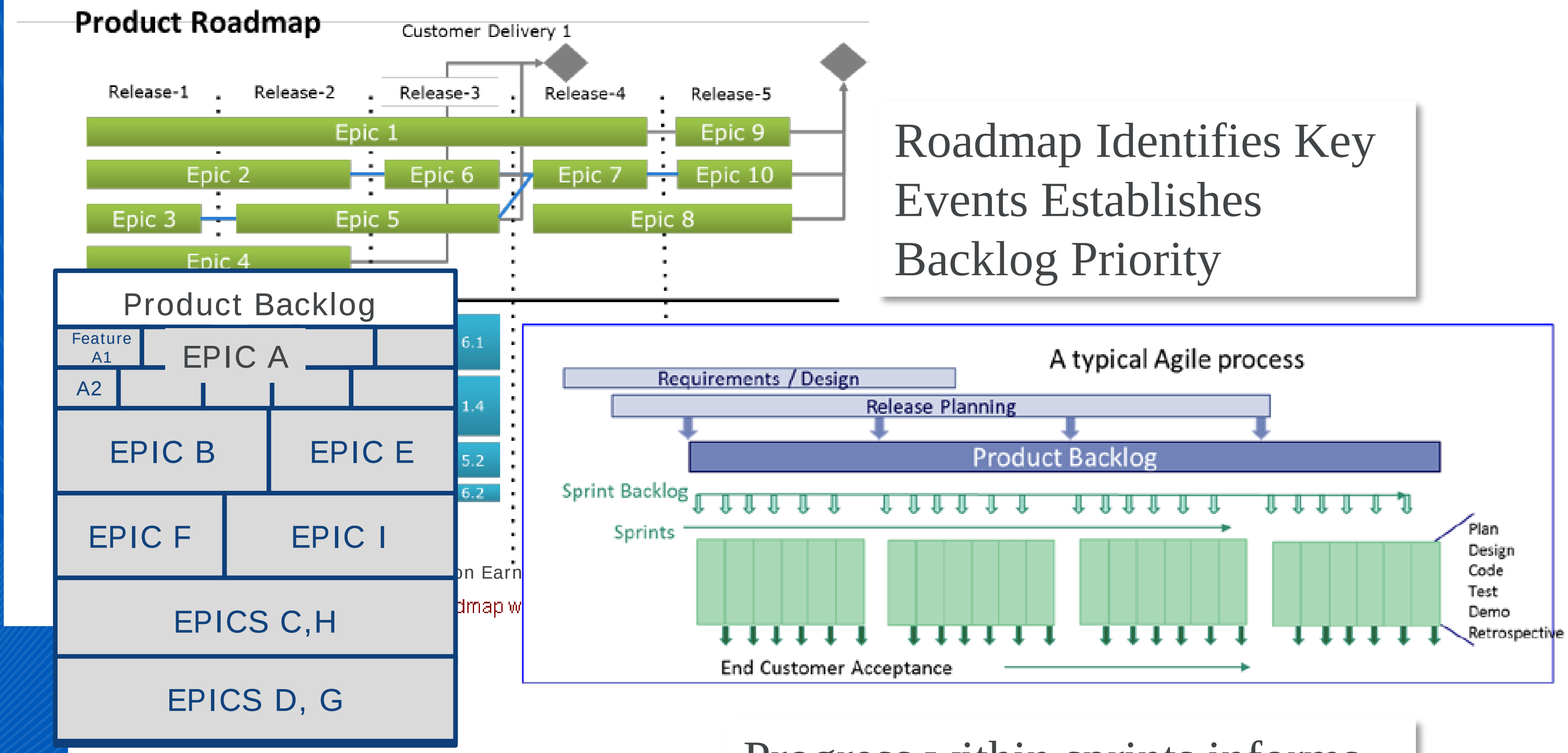




How to Apply Agile in a Non-Agile World



Time Phasing the Work In Practice



Roadmap Identifies Key Events Establishes Backlog Priority

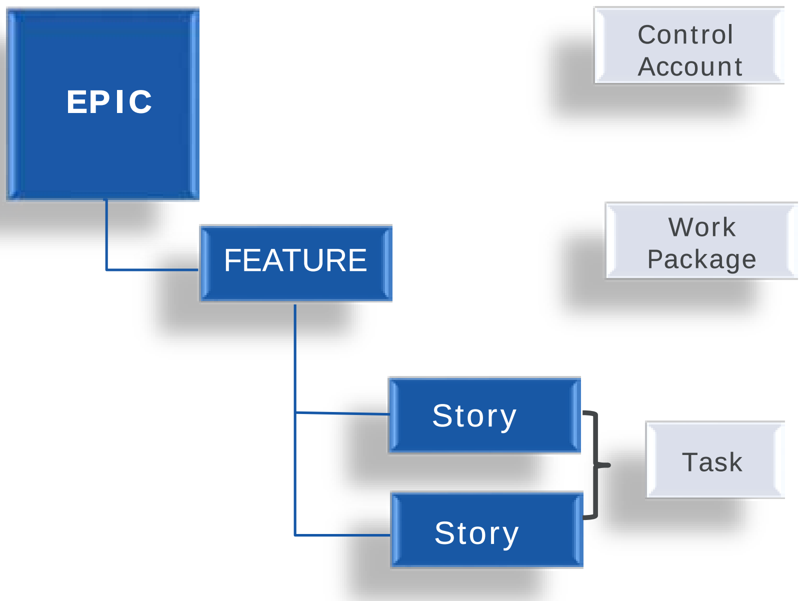
Progress within sprints informs backlog

GALORATH

Agile Hierarchy
Why the WBS is important

- Epic – Control Account
  - Group of Functional Features
- Feature – Work Package
  - A Specific Function within the Program
  - Costs and resource planning should be performed at the Work Package Level (Best Practices)
- Story – An individual part of completing the work package, which rolls into a sprint
  - Made up of Measurable Story Points

*Note: In Agile it is important to note that a Sprint is referred to as a “time box”

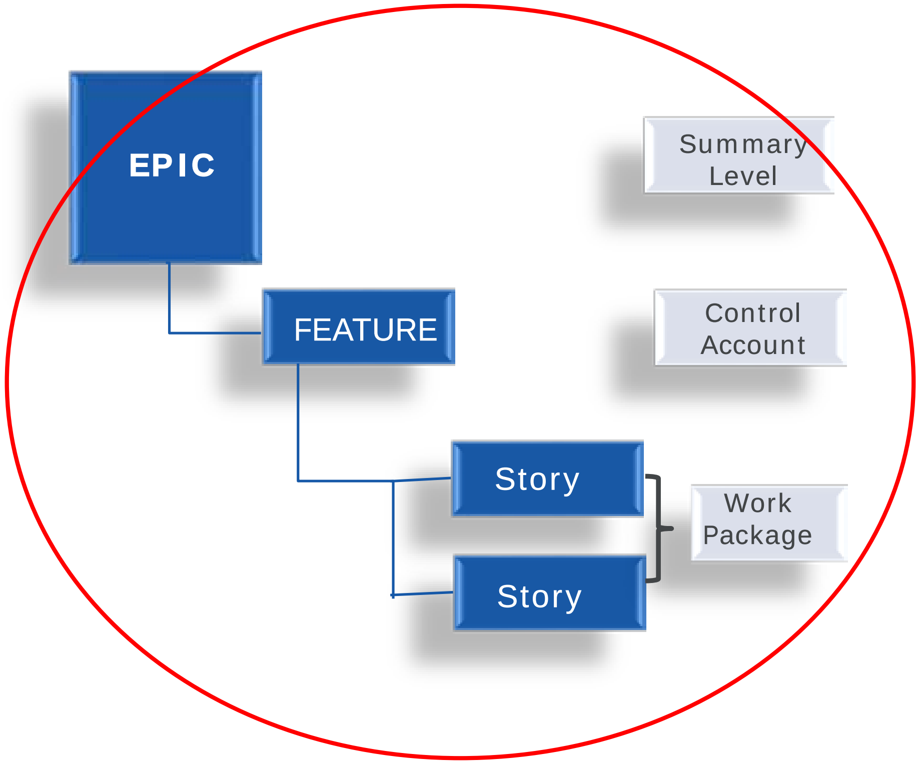


GALORATH

Agile Planning
Pitfalls in Planning

- Feature located too high on the WBS, as the Control Account
  - Causes the Story to be Baselined
  - Requires a BCR when a Sprint is late or not finished
- New Requirements are **<u>not</u>** added to a current Story, they will become a new Story in Backlog

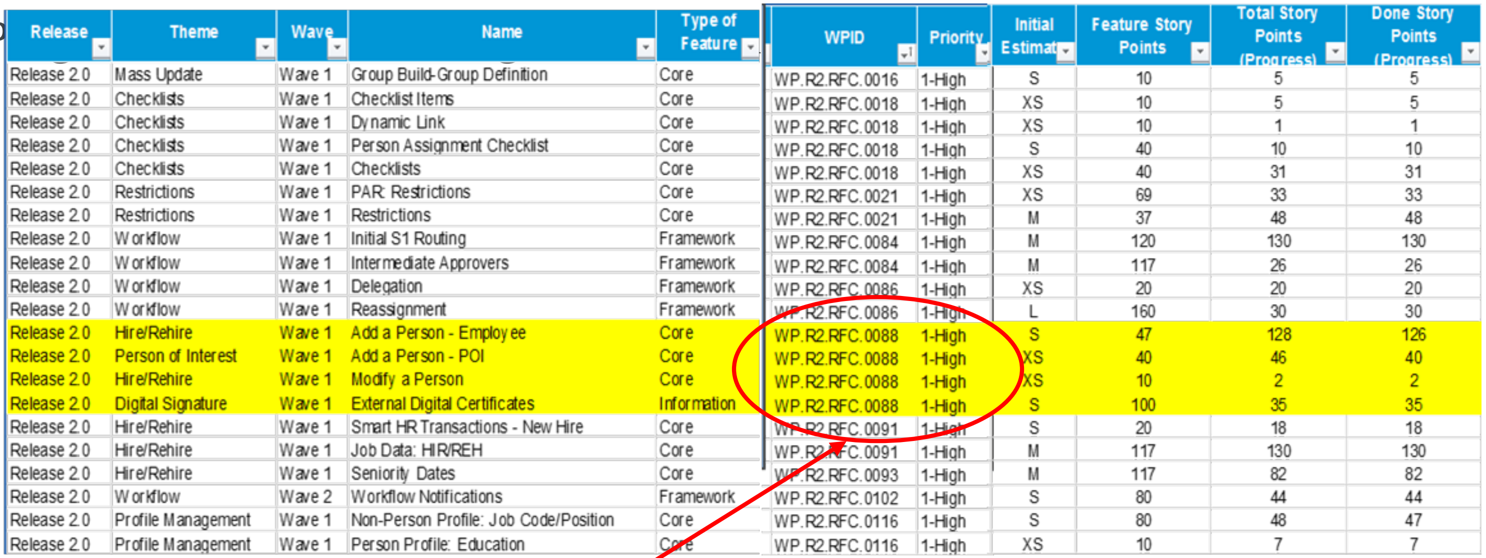
Where things get too complex



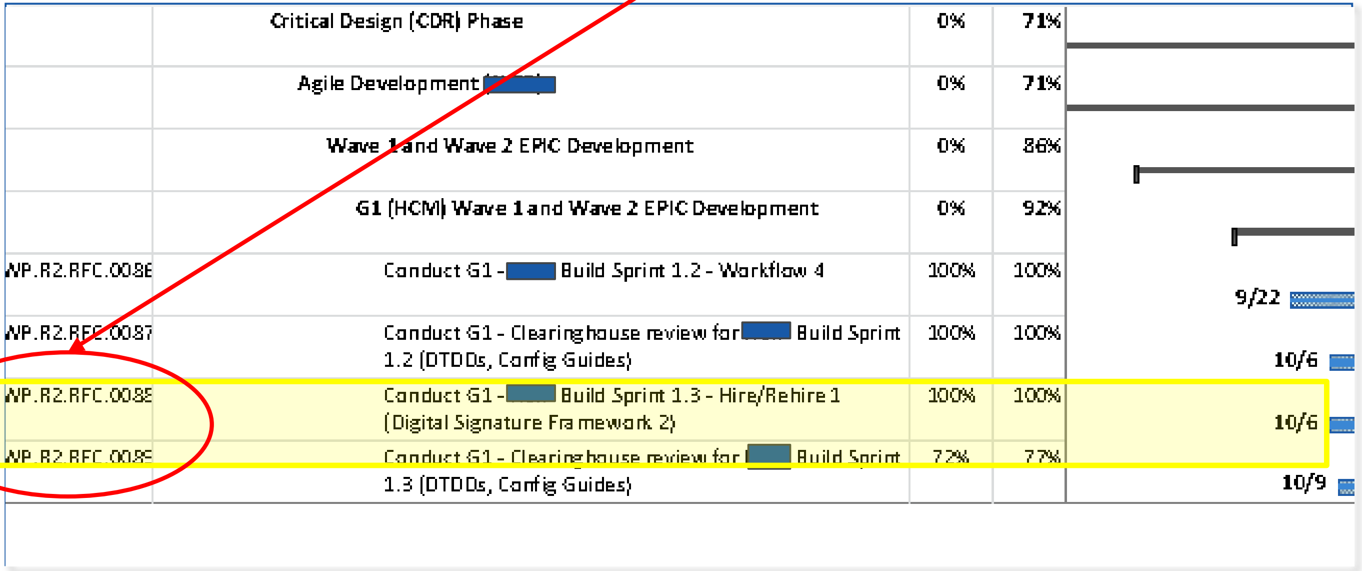
Be careful how you develop your WBS!!!

Progress in Agile Rolled to Feature Level

P

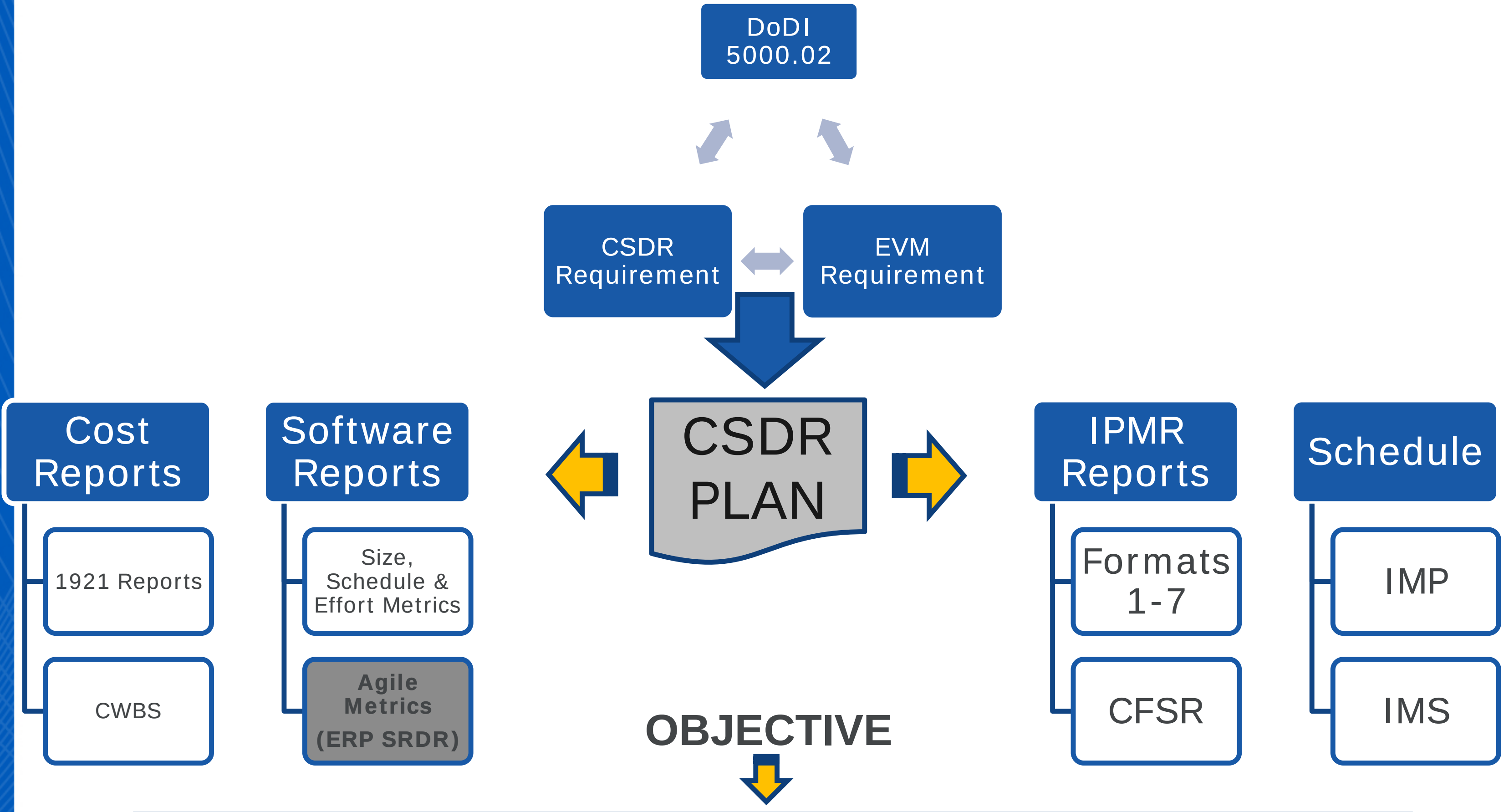


Release	Theme	Wave	Name	Type of Feature	WPID	Priority	Initial Estimate	Feature Story Points	Total Story Points (Progress)	Done Story Points (Progress)
Release 2.0	Mass Update	Wave 1	Group Build-Group Definition	Core	WP.R2.RFC.0016	1-High	S	10	5	5
Release 2.0	Checklists	Wave 1	Checklist Items	Core	WP.R2.RFC.0018	1-High	XS	10	5	5
Release 2.0	Checklists	Wave 1	Dynamic Link	Core	WP.R2.RFC.0018	1-High	XS	10	1	1
Release 2.0	Checklists	Wave 1	Person Assignment Checklist	Core	WP.R2.RFC.0018	1-High	S	40	10	10
Release 2.0	Checklists	Wave 1	Checklists	Core	WP.R2.RFC.0018	1-High	XS	40	31	31
Release 2.0	Restrictions	Wave 1	PAR Restrictions	Core	WP.R2.RFC.0021	1-High	XS	69	33	33
Release 2.0	Restrictions	Wave 1	Restrictions	Core	WP.R2.RFC.0021	1-High	M	37	48	48
Release 2.0	Workflow	Wave 1	Initial S1 Routing	Framework	WP.R2.RFC.0084	1-High	M	120	130	130
Release 2.0	Workflow	Wave 1	Intermediate Approvers	Framework	WP.R2.RFC.0084	1-High	M	117	26	26
Release 2.0	Workflow	Wave 1	Delegation	Framework	WP.R2.RFC.0086	1-High	XS	20	20	20
Release 2.0	Workflow	Wave 1	Reassignment	Framework	WP.R2.RFC.0086	1-High	L	160	30	30
Release 2.0	Hire/Rehire	Wave 1	Add a Person - Employee	Core	WP.R2.RFC.0088	1-High	S	47	128	126
Release 2.0	Person of Interest	Wave 1	Add a Person - POI	Core	WP.R2.RFC.0088	1-High	XS	40	46	40
Release 2.0	Hire/Rehire	Wave 1	Modify a Person	Core	WP.R2.RFC.0088	1-High	XS	10	2	2
Release 2.0	Digital Signature	Wave 1	External Digital Certificates	Information	WP.R2.RFC.0088	1-High	S	100	35	35
Release 2.0	Hire/Rehire	Wave 1	Smart HR Transactions - New Hire	Core	WP.R2.RFC.0091	1-High	S	20	18	18
Release 2.0	Hire/Rehire	Wave 1	Job Data: HIR/REH	Core	WP.R2.RFC.0091	1-High	M	117	130	130
Release 2.0	Hire/Rehire	Wave 1	Seniority Dates	Core	WP.R2.RFC.0093	1-High	M	117	82	82
Release 2.0	Workflow	Wave 2	Workflow Notifications	Framework	WP.R2.RFC.0102	1-High	S	80	44	44
Release 2.0	Profile Management	Wave 1	Non-Person Profile: Job Code/Position	Core	WP.R2.RFC.0116	1-High	S	80	48	47
Release 2.0	Profile Management	Wave 1	Person Profile: Education	Core	WP.R2.RFC.0116	1-High	XS	10	7	7



	Critical Design (CDR) Phase	0%	71%	
	Agile Development	0%	71%	
	Wave 1 and Wave 2 EPIC Development	0%	86%	
	G1 (HCM) Wave 1 and Wave 2 EPIC Development	0%	92%	
WP.R2.RFC.0086	Conduct G1 - Build Sprint 1.2 - Workflow 4	100%	100%	9/22
WP.R2.RFC.0087	Conduct G1 - Clearinghouse review for Build Sprint 1.2 (DTDDs, Config Guides)	100%	100%	10/6
WP.R2.RFC.0088	Conduct G1 - Build Sprint 1.3 - Hire/Rehire 1 (Digital Signature Framework 2)	100%	100%	10/6
WP.R2.RFC.0089	Conduct G1 - Clearinghouse review for Build Sprint 1.3 (DTDDs, Config Guides)	72%	72%	10/9

Traditional EVM Reporting



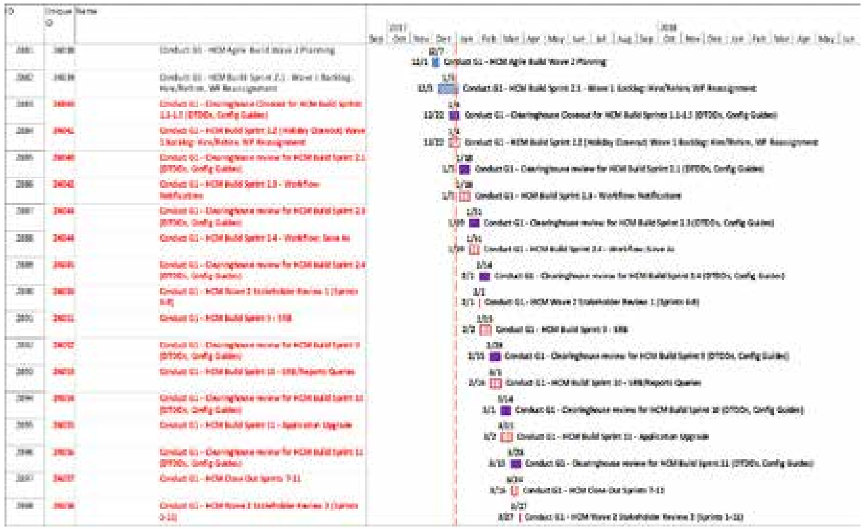
CSDR -Plan
Common Work Breakdown Structure (CWBS & PWBS)

CSDR & EVM Planning, Execution and Reporting on Track

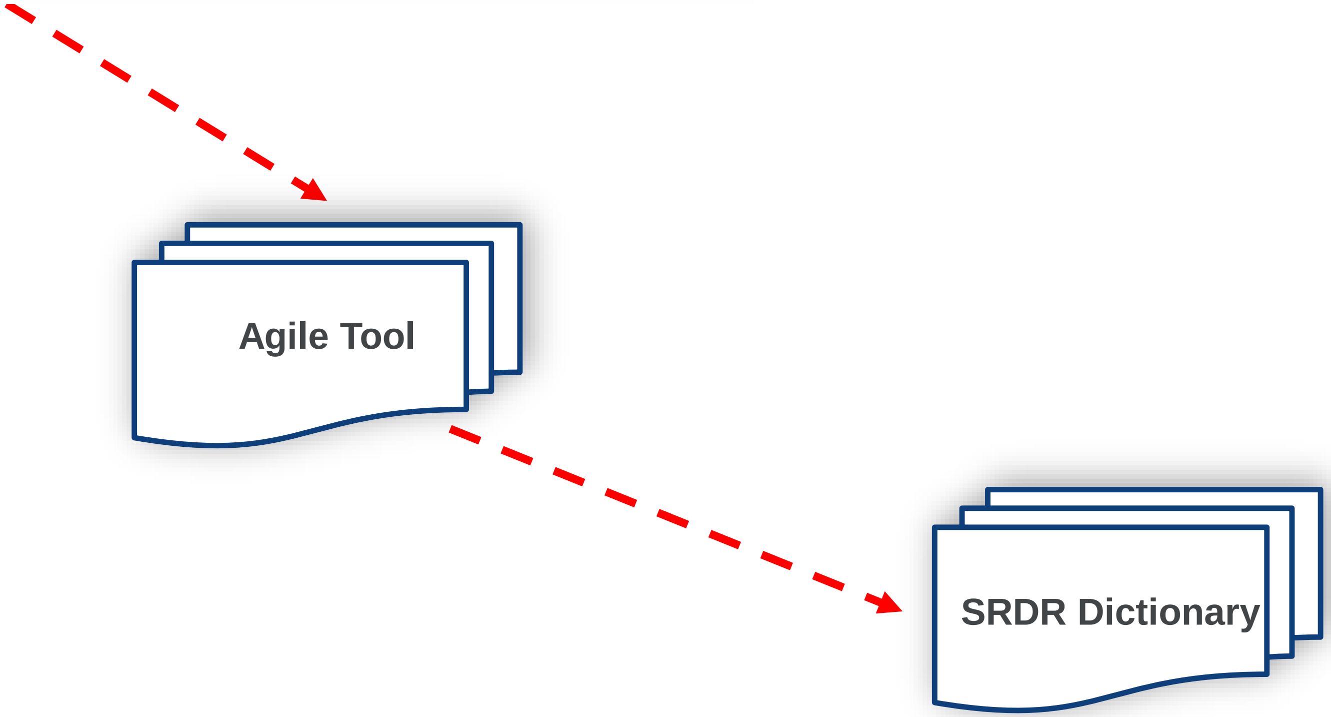
GALORATH

Integration of IMS, Agile Tool & SRDR Dictionary

IMS



ID	Unique ID	Name
[illegible]	[illegible]	Conduct G1 - HCM Agile Build Wave 2 Planning
[illegible]	[illegible]	Conduct G1 - HCM Build Sprint 2.1 - Wave 1 Backlog: Hire/Rehire, WF Reassignment
[illegible]	[illegible]	Conduct G1 – Clearinghouse Closeout for HCM Build Sprints 1.1-1.5 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 1.2 (Holiday Closeout) Wave 1 Backlog: Hire/Rehire, WF Reassignment
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 2.1 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 1.3 - Workflow: Notifications
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 2.3 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 2.4 - Workflow: Save As
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 2.4 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Wave 2 Stakeholder Review 1 (Sprints 6-8)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 9 - SRB
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 9 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 10 - SRB/Reports Queries
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 10 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Build Sprint 11 - Application Upgrade
[illegible]	[illegible]	Conduct G1 – Clearinghouse review for HCM Build Sprint 11 (DTDOs, Config Guides)
[illegible]	[illegible]	Conduct G1 – HCM Close Out Sprints 7-11
[illegible]	[illegible]	Conduct G1 – HCM Wave 2 Stakeholder Review 3 (Sprints 1-11)



The integration of IMS, Agile PM Tool, and SRDR Dictionary are all linked via WBS numbers.

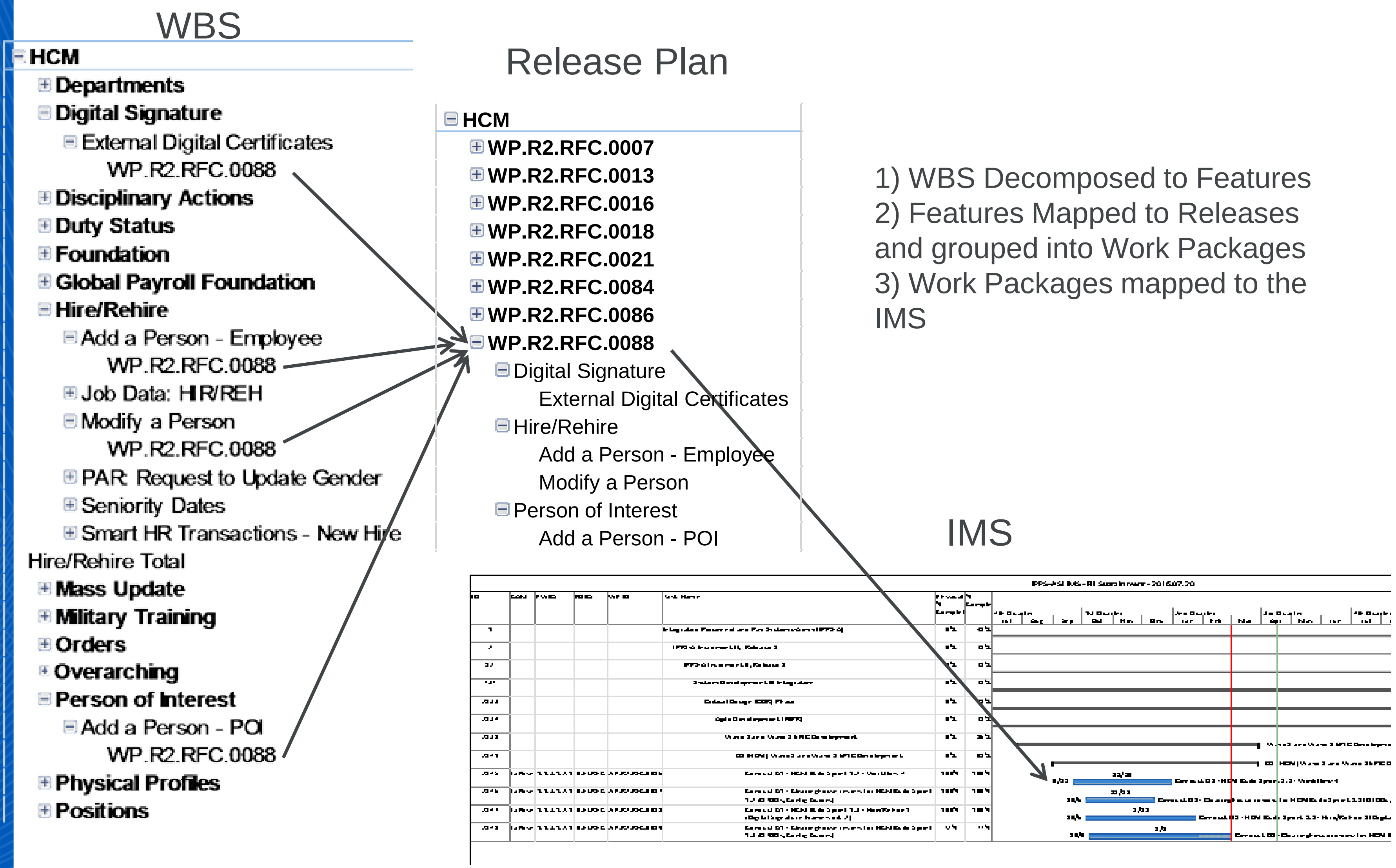
GALORATH

Agile Scheduling
How everything falls in line

- Features are assigned to a Scrum Team
- The stories in each Feature are prioritized and planned, as to when they will be worked
- During the planning process of story points begins with the Scrum Teams divide the assignments for design, development, test, and backlog/clearing house review takes place and assessing the Story Point count for each Story
- Each day the active Sprints are assessed and recorded as to the number of story points which were accomplished and entered into the Agile Management Tool
- An overall assessment of whether or not the Story can be completed on time

Agile to EVM Traceability

1) WBS Decomposed to Features
2) Features Mapped to Releases and grouped into Work Packages
3) Work Packages mapped to the IMS



WBS

- HCM
  - Departments
  - Digital Signature
    - External Digital Certificates
      - WP.R2.RFC.0088
  - Disciplinary Actions
  - Duty Status
  - Foundation
  - Global Payroll Foundation
  - Hire/Rehire
    - Add a Person - Employee
      - WP.R2.RFC.0088
    - Job Data: HIR/REH
    - Modify a Person
      - WP.R2.RFC.0088
    - PAR: Request to Update Gender
    - Seniority Dates
    - Smart HR Transactions - New Hire
  - Hire/Rehire Total
  - Mass Update
  - Military Training
  - Orders
  - Overarching
  - Person of Interest
    - Add a Person - POI
      - WP.R2.RFC.0088
  - Physical Profiles
  - Positions

Release Plan

- HCM
  - WP.R2.RFC.0007
  - WP.R2.RFC.0013
  - WP.R2.RFC.0016
  - WP.R2.RFC.0018
  - WP.R2.RFC.0021
  - WP.R2.RFC.0084
  - WP.R2.RFC.0086
  - WP.R2.RFC.0088
    - Digital Signature
      - External Digital Certificates
    - Hire/Rehire
      - Add a Person - Employee
      - Modify a Person
    - Person of Interest
      - Add a Person - POI

IMS

GALORATH

Agile Baseline

How to keep from going insane with Baseline Changes

- Baseline Changes will happen, they happen to most programs, but they should be limited to as few as possible
- There is constant movement in an Agile program. The changes are divided into "How do you account for those changes?" First these questions need to be asked:
  - Will the change impact the Baseline?
  - Can the change take place within the existing timeframe of the Feature?
  - If the change impacts the Baseline, a BCR needs to be submitted
  - It all goes back to careful and thoughtful PLANNING and development of the program WBS.

Conclusion

- Agile and EVM – Beyond the Manifesto
  - Being Agile is not just for SW folks
  - Agile can have real benefit to complex DoD Programs
- Agile and EVM complement one another
  - Agile methods are focused on delivering increments of working product often
  - Agile methods are strong at the tactical day to day management of work
  - EVM measure the value of delivered product in terms of cost and schedule
  - EVM processes focus on the strategic direction of a program based on tactical status