estimate

estimate • analyze • plan • control

Applying Earned Value Management to Agile Software Development Programs

Bob Hunt

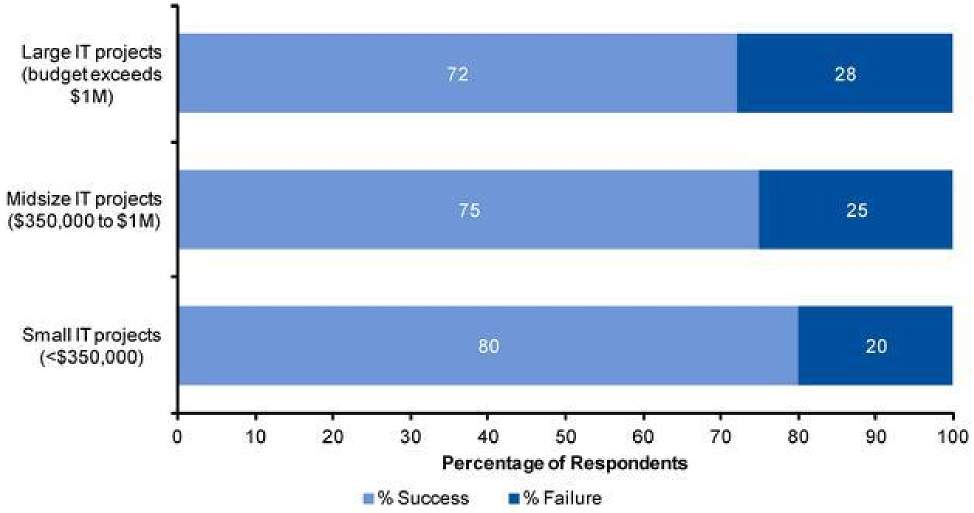
Michael Thompson

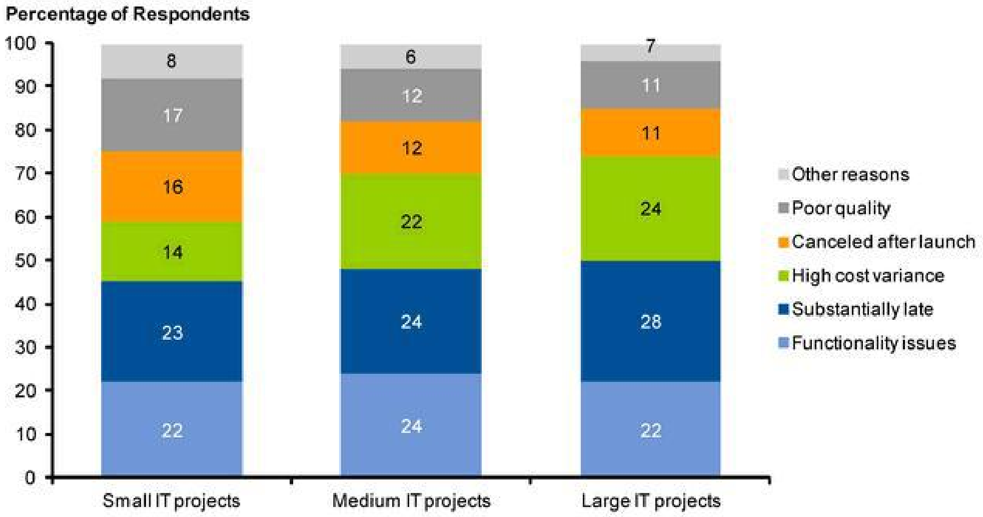
Galorath Federal Incorporated

GALORATH

Agenda

IT Project Success





Why they fail

Software Development

- While there are many approaches to Software Development, they can generally be placed into 2 categories:
  - Plan Driven – following a version of the Waterfall Development Process
  - Iterative Driven – following a “version” of the Agile Development Process
- Plan Drive programs have an assumption of some reliable/realistic size metric, for example:
  - Source Lines of Code (SLOC)
  - Function Points
  - Use Cases, User Stories, Web Pages

What is Agile Software Development?

- In the late 1990s, several methodologies received increasing public attention
- Each had a different combination of old, new, and transmuted old ideas, but they all emphasized:
  - Close collaboration between the programmer and business experts
  - Face-to-face communication (as more efficient than written documentation)
  - Frequent delivery of new deployable business value
  - Tight, self-organizing teams
  - And ways to craft the code and the team such that the inevitable requirements churn was not a crisis

How Formal Is Agile?

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

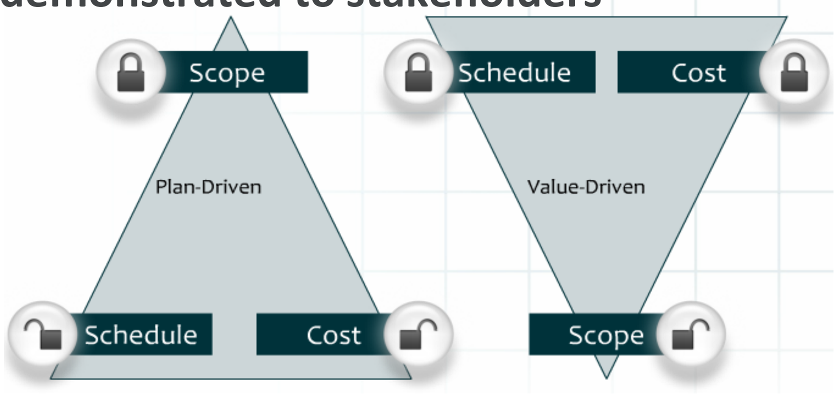
That is, while there is value in the items on the right, we value the items on the left more."

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

Agile 101 – How It Works

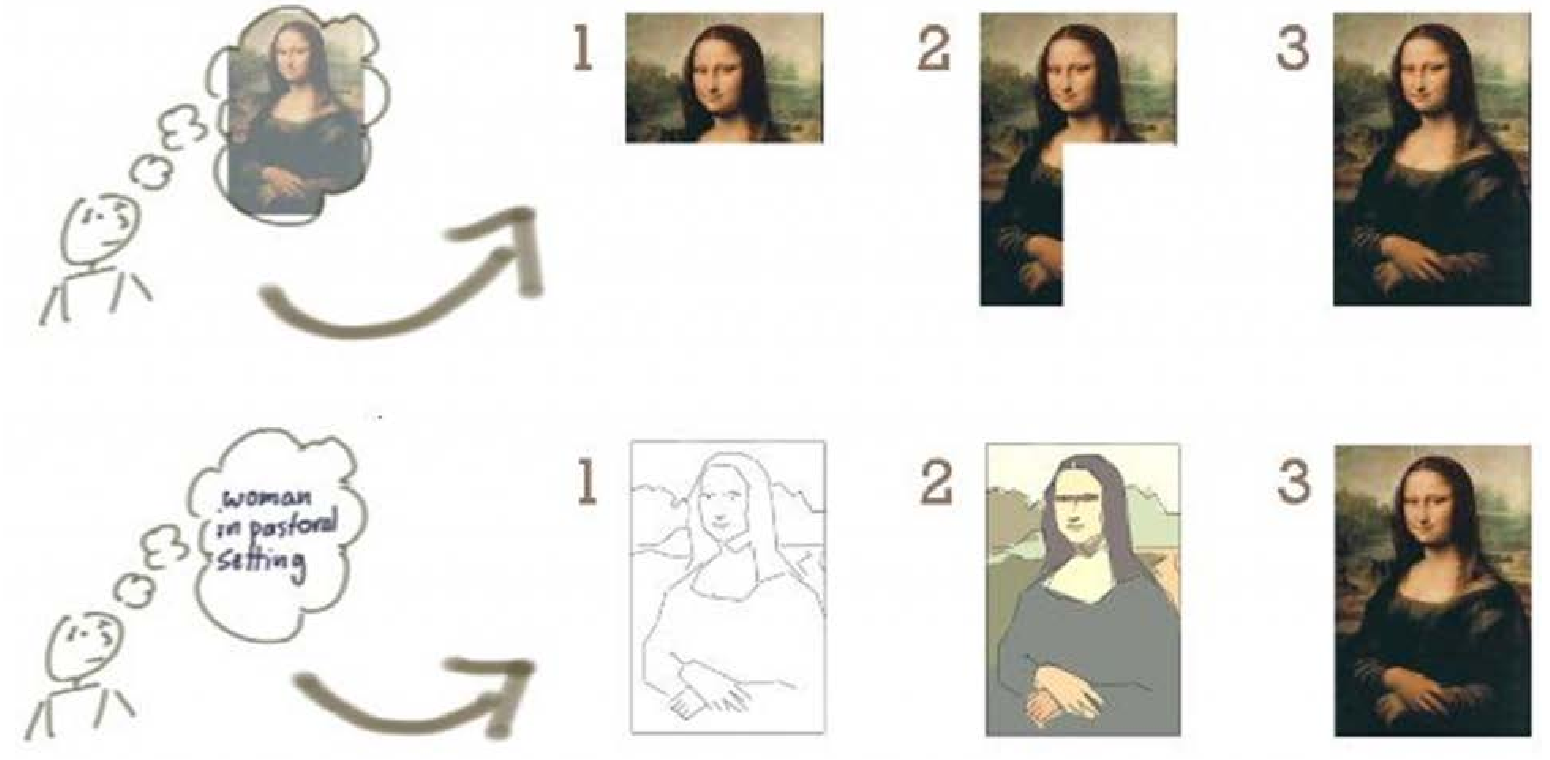
GALORATH

- **Agile is a set of software development methods in which solutions evolve through collaboration**
  - Integrated teams include PeopleSoft SMEs to configure the product, PeopleSoft Developers, Data Integration Developers, and Testers
- **Development is iterative**
  - The traditional software development phases are seen as continuous activities
  - Work is broken into smaller tasks
  - Multiple iterations may be required to release a product
  - Documentation is created as-built
- **For each iteration, a working product is demonstrated to stakeholders**
- **Emphasizes value-driven approach**
  - The usual project constraints still apply
  - Focusing on value allows most important functionality to be delivered first
- **Technology agnostic**



Agile 101 – Development Approach Metaphor

- The waterfall approach is akin to painting by numbers, as it calls for a fully formed idea at the start, which is built incrementally, piece by piece without flexibility



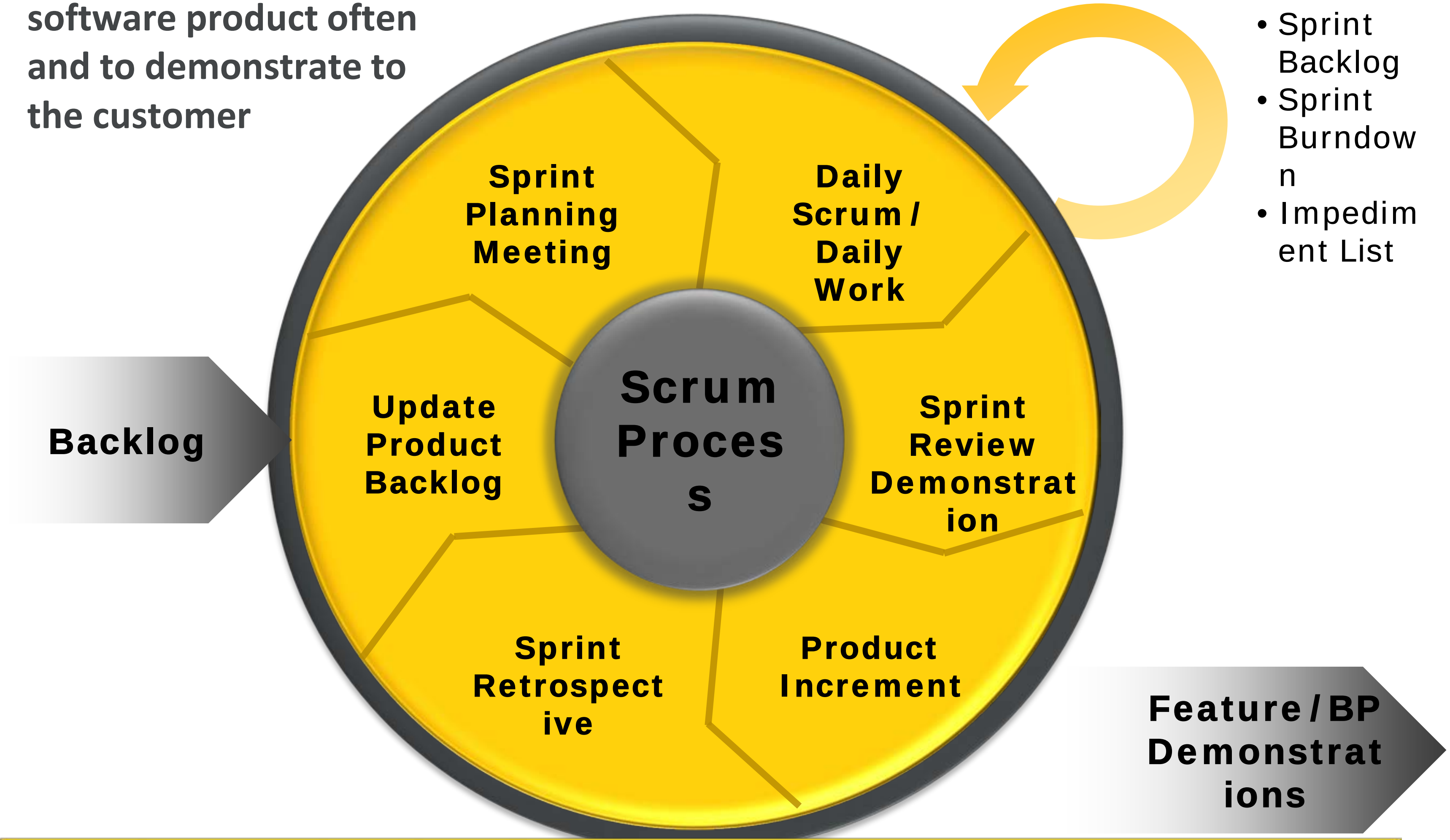
- With Agile, we start with a concept: for IPPS-A, the COTS product is the starting point
- Then, we iteratively build a rough version and validate it, slowly improving the definition and quality

Agile 101 – Key Agile Terminology

Term	Definition
Scrum	A framework for team collaboration on complex software projects. 1-10 people (have seen up to **20**)
Sprint	A short multiple-week period where a team completely builds working, tested software. All phases of the SDLC are executed iteratively during a sprint – Analysis, Design, Code, Test. 1-6 weeks (have seen up to **13** weeks) *(13 conveniently give 4 sprints per year*)
Feature	A set of specifications that can be shown in a user demonstration and oriented on system capabilities.
Epic	A description of how work gets done using the new software (To-Be business process).
Spike	A special type of story used for research and prototyping activities, which can be functional or technical.
Backlog	A single definitive repository for all upcoming work. It consists primarily of future features intended to address user needs and deliver business benefits, as well as architectural features required to build the product.

Agile Implementation

- To create working software product often and to demonstrate to the customer



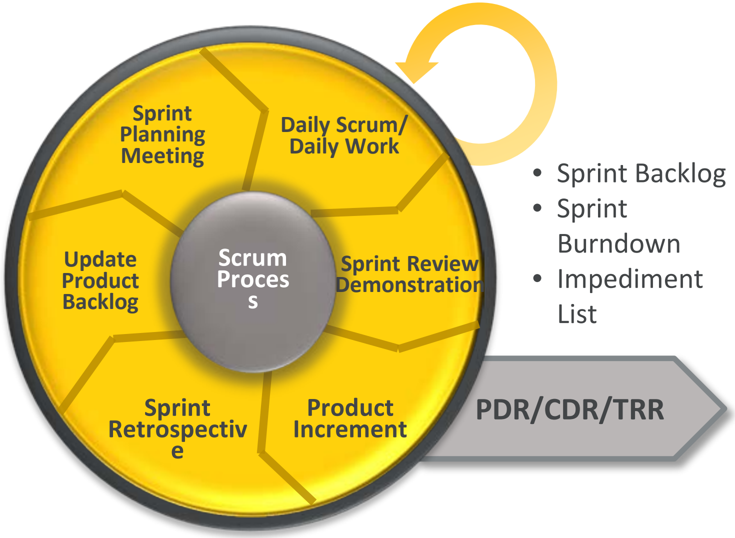
- Sprint Backlog
- Sprint Burndown
- Impediment List

Ensures holistic activities and iterative processes

Agile 101 – Sprint Breakdown

GALORATH

- **Sprint Planning Meeting**
  - Product owner and scrum team meet at the start of the sprint to review the sprint goal, including a review of the requirements/features to be accomplished and corresponding tasks.
- **Daily Scum/Daily Work**
  - Scrum team meets daily to review yesterday’s accomplishments, plan for today, and any blockers.
  - Scrum team performs design, build, and test tasks.
- **Sprint Review Demonstration**
  - Scrum team demonstrates sprint accomplishments (requirements and/or features implemented during that sprint) to the product owner.
- **Product Increment**
  - The sum of all the product backlog items completed across all the scrum teams through the last sprint.
- **Sprint Retrospective**
  - The scrum team meets to discuss what to keep doing, stop doing, and start doing. Focus on what is actionable for the next sprint.
- **Update Product Backlog**
  - The product owner continuously updates (adds to, reprioritizes, etc.) the product backlog.



- Sprint Backlog
- Sprint Burndown
- Impediment List

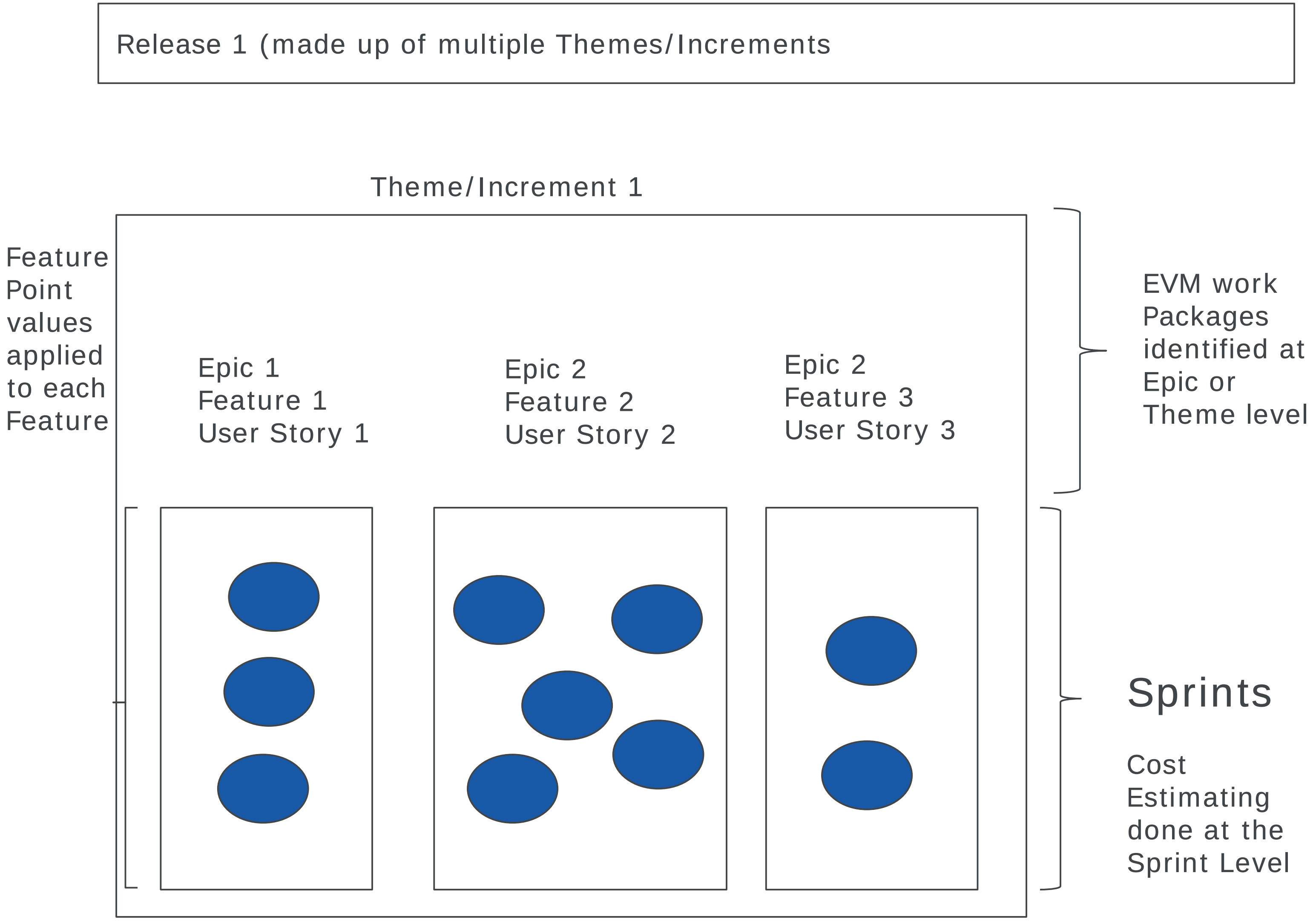
Agile 101 – Recap

- **Agile is a disciplined methodology.**
- **Agile is not…**
  - …unlimited or uncontrolled scope.
  - …unplanned.
  - …undocumented.
  - …unverified.
  - …mini waterfall.
  - …trial and error.
  - …a synonym for flexible.
  - …a synonym for fast.

Agile 101 - Summary

- **Waterfall faces challenges in a large-scale ERP implementation.**
  - Significant effort to build and maintain momentum
  - High risk of a resulting product that does not meet needs
- **Looking for an alternative:**
  - Best candidate is Scaled Agile Framework (SAFe)
  - Lower risk implementation
  - Produces better results, meets IPPS-A Vision
  - Allows us to confirm that we are building the right thing
  - Gets early buy-in
  - Reduces the risk of rework when it is too late and more costly to change
  - Improved understanding of progress and cost
- **Better software development process, better user experience:**
  - Break work into smaller, more manageable segments
  - Measure progress based on working software product
  - Drive functionality working on through to completion
  - Emphasize showing working versions of product early and often to validate
  - Use product to review whether it meets requirements – shift to As-Built documentation
  - Use government time and resources better by reviewing software, not paper
  - Use cumulative assembly testing for earlier validation of product
  - Have automated regression test bed available on go-live

Agile Building Blocks*



Release 1 (made up of multiple Themes/Increments

Theme/Increment 1

Feature Point values applied to each Feature

Epic 1
Feature 1
User Story 1

Epic 2
Feature 2
User Story 2

Epic 2
Feature 3
User Story 3

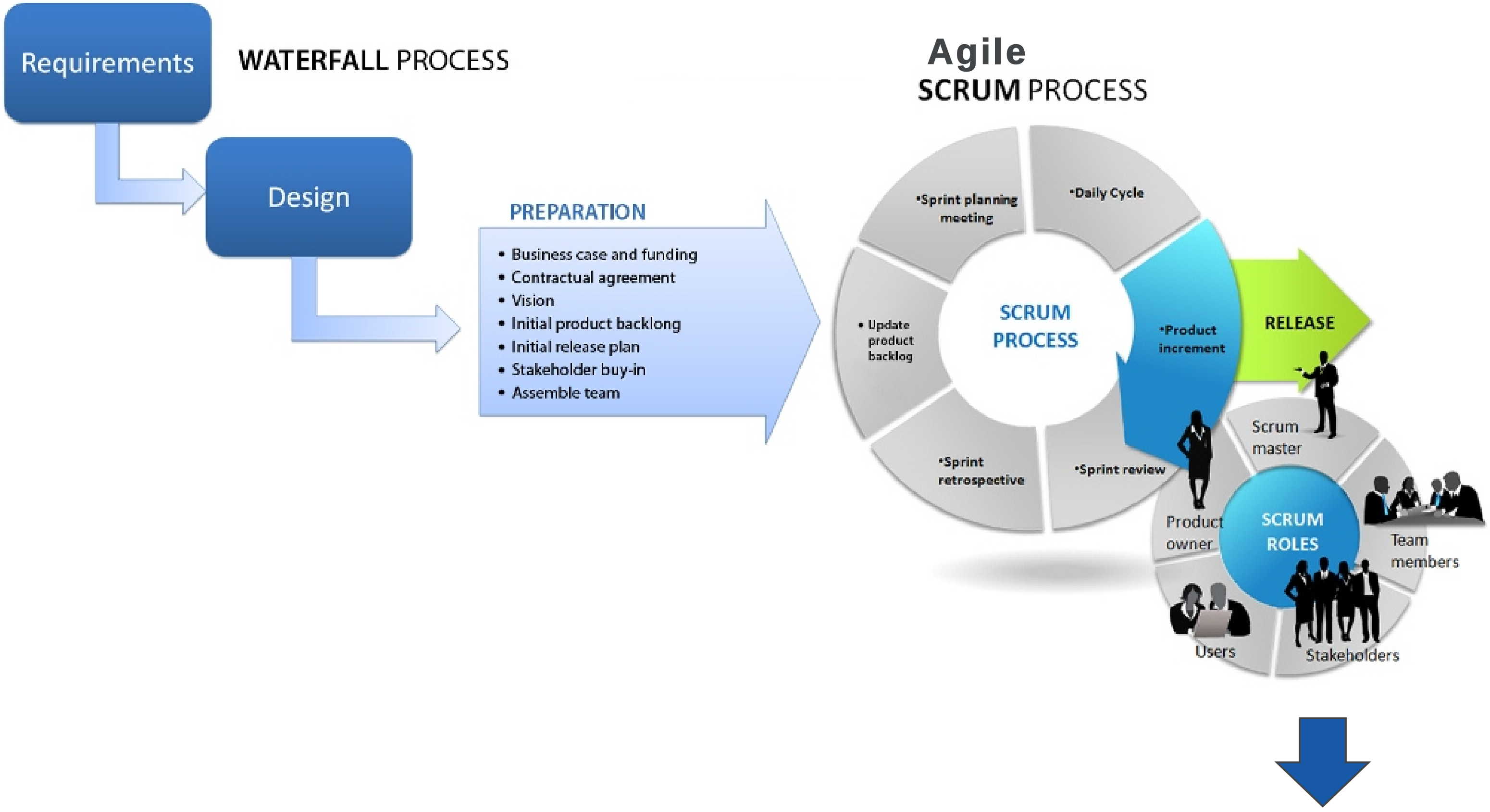
EVM work Packages identified at Epic or Theme level

Sprints

Cost Estimating done at the Sprint Level

* These "building blocks" are program specific and may be called by different names

Hybrid Agile Development/Acquisition



Requirements

WATERFALL PROCESS

Design

PREPARATION

- Business case and funding
- Contractual agreement
- Vision
- Initial product backlong
- Initial release plan
- Stakeholder buy-in
- Assemble team

Agile
SCRUM PROCESS

SCRUM PROCESS

- Sprint planning meeting
- Daily Cycle
- Product increment
- Sprint review
- Sprint retrospective
- Update product backlog

RELEASE

SCRUM ROLES

Scrum master

Team members

Stakeholders

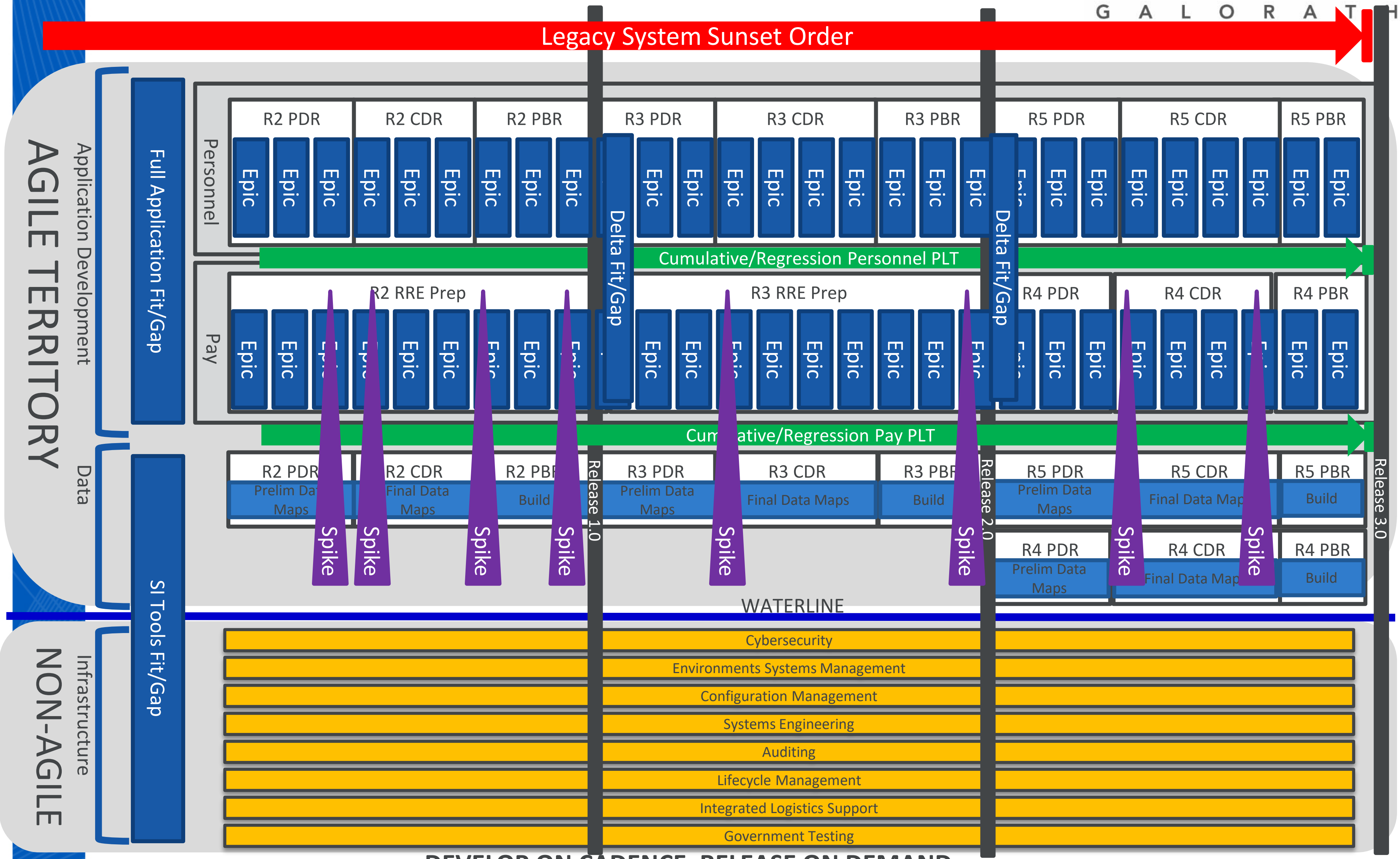
Users

Product owner

Testing and Sustainment ?

Agile – System View

GALORATH



Legacy System Sunset Order

AGILE TERRITORY

Application Development

Full Application Fit/Gap

Personnel

R2 PDR | R2 CDR | R2 PBR | R3 PDR | R3 CDR | R3 PBR | R5 PDR | R5 CDR | R5 PBR

Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic

Delta Fit/Gap

Cumulative/Regression Personnel PLT

Pay

R2 RRE Prep | R3 RRE Prep | R4 PDR | R4 CDR | R4 PBR

Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic Epic

Cumulative/Regression Pay PLT

Data

R2 PDR Prelim Data Maps | R2 CDR Final Data Maps | R2 PBR Build | R3 PDR Prelim Data Maps | R3 CDR Final Data Maps | R3 PBR Build | R5 PDR Prelim Data Maps | R5 CDR Final Data Maps | R5 PBR Build

R4 PDR Prelim Data Maps | R4 CDR Final Data Maps | R4 PBR Build

Spike Spike Spike Spike Spike Spike Spike Spike

Release 1.0

Release 2.0

Release 3.0

WATERLINE

NON-AGILE

Infrastructure

SI Tools Fit/Gap

Cybersecurity

Environments Systems Management

Configuration Management

Systems Engineering

Auditing

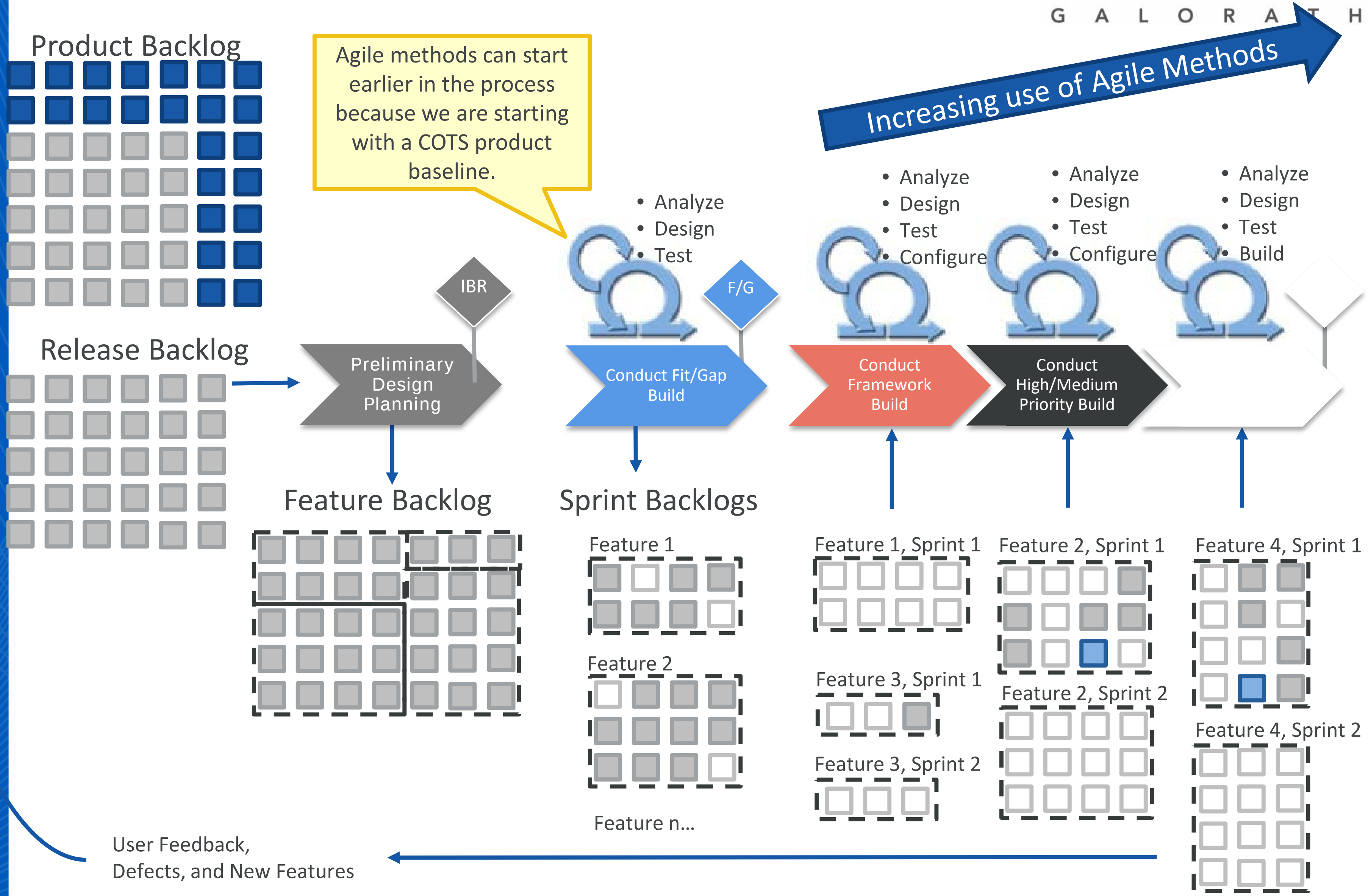
Lifecycle Management

Integrated Logistics Support

Government Testing

DEVELOP ON CADENCE, RELEASE ON DEMAND

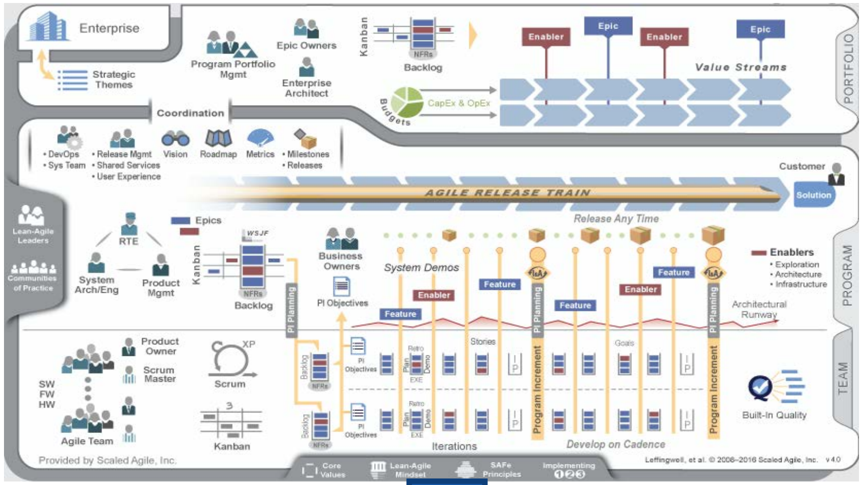
Agile Release and Sprint Backlog

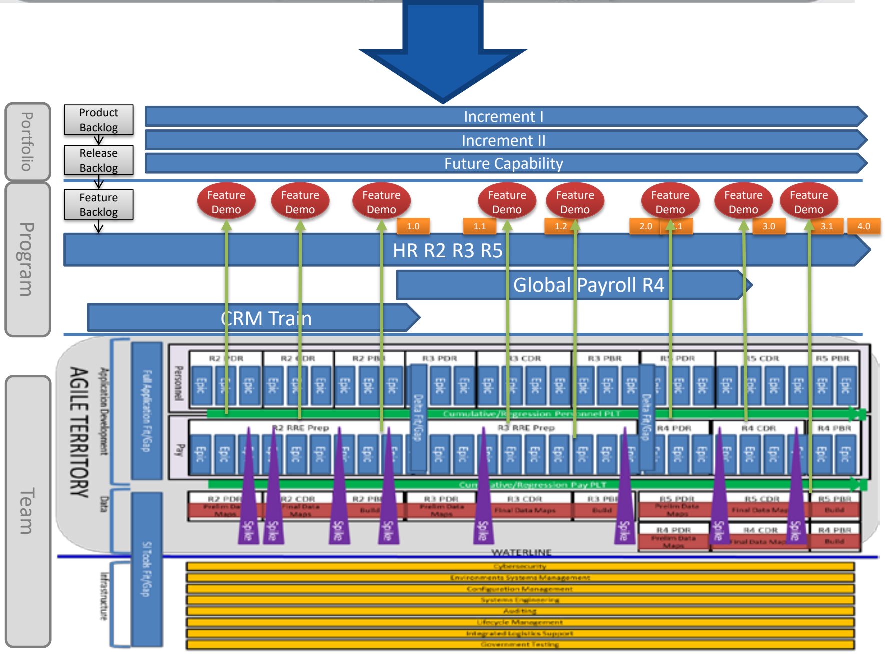


Agile methods can start earlier in the process because we are starting with a COTS product baseline.

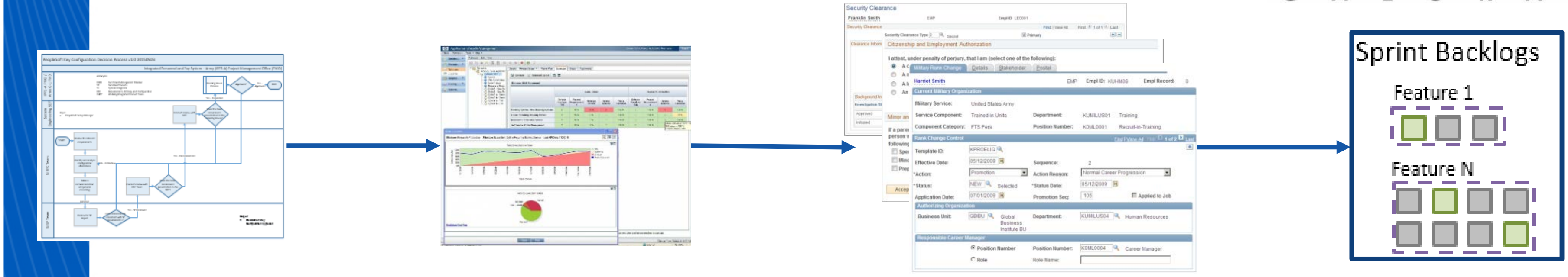
How to Scale

- *Agile methods are generally used for projects of smaller scope.*
- *Scaled Agile Framework (SAFe) builds from Agile methods to provide a construct for large-scale implementations.*
- *A playbook tailored to IPPS-A will inform the development of the IMS and form the basis to guide teams.*
  - Introduces features to bridge the gap between epics (business processes) and sprints.
  - Introduces spikes to explore various approaches to address key foundational decisions.





How to Define Work



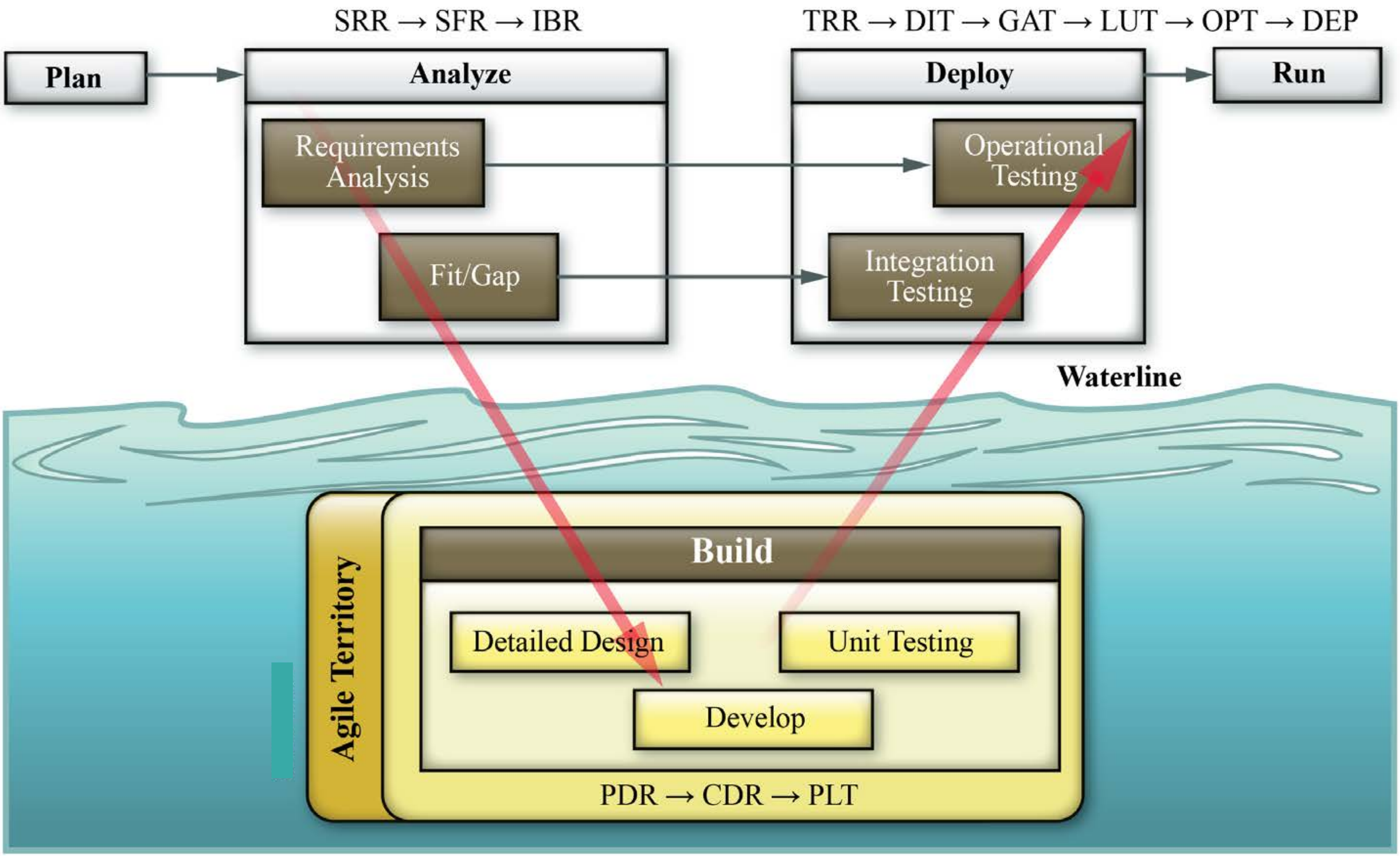
1 - Requirements are defined based on the Business Process.

2 - Requirements are grouped into features aligned with the delivered software product.

3 - Features are decomposed into tasks to be executed in sprints.

- **Features will be classified as one of the following types:**
  - Framework: Foundational capabilities that need to be completed early as they will be leveraged throughout the solution.
  - Core: Capabilities support transactional processing. These include setup tables that maintain valid values and capabilities to manage employee records.
  - Self-Service: Any feature that has a self-service component is identified.
  - Information: Reports, queries, dashboards.
  - Data: Conversions and interfaces.

How to Apply Agile in a Non-Agile World

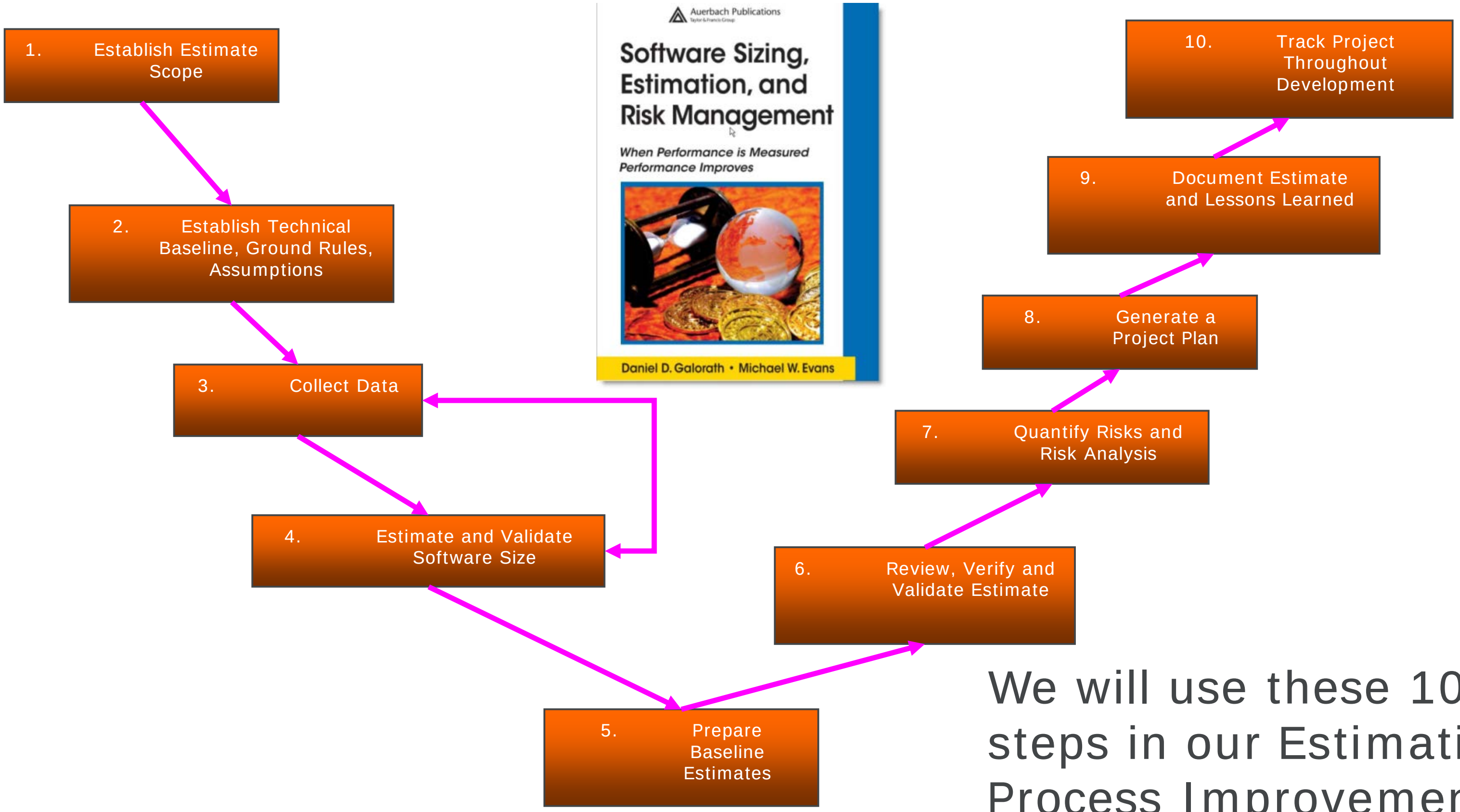


Galorath: Driving The State of the Art (10 Step Estimation Process)

G A L O R A T H

When performance is measured performance improves

Estimation processes are independent of tools



1. Establish Estimate Scope
2. Establish Technical Baseline, Ground Rules, Assumptions
3. Collect Data
4. Estimate and Validate Software Size
5. Prepare Baseline Estimates
6. Review, Verify and Validate Estimate
7. Quantify Risks and Risk Analysis
8. Generate a Project Plan
9. Document Estimate and Lessons Learned
10. Track Project Throughout Development

Auerbach Publications

Software Sizing, Estimation, and Risk Management

When Performance is Measured Performance Improves

Daniel D. Galorath • Michael W. Evans

We will use these 10 steps in our Estimation Process Improvement Program (EPIP)

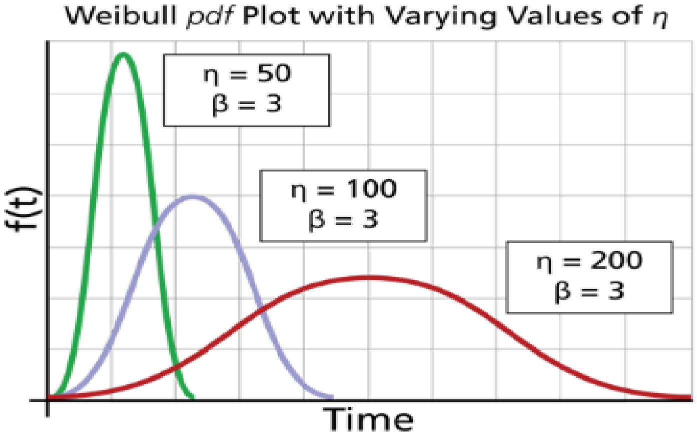
Fundamental assumptions of most Software Estimating Models

- There is a fixed relationship between size and effort, e.g.

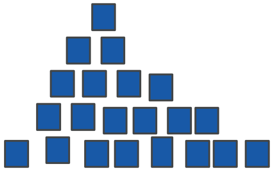
(Effort**n)*Time = Size/Technology

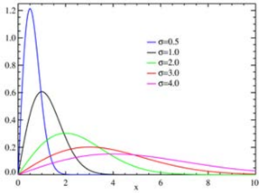
- Results are then modified by current trends and analyses

- Total effort can be distributed by a mathematical model; e.g. Weibull, Rayleigh



- **Methodology 1: Since many Agile programs are fixed price, it is often just a matter of labor rates times quantity**
- **Methodology 2: Simple Build-up approach** based on averages can be defined as:
  - Sprint Team Size (SS) x Sprint length (Sp time) x Number of Sprints (# Sprints)
- **Methodology 3: Structured approach** based on established “velocity” – most often used internally by the developer since detailed/sensitive data are available to them
- **Methodology 4: Automated Models approach** based on a size metric – which may be difficult to quantify





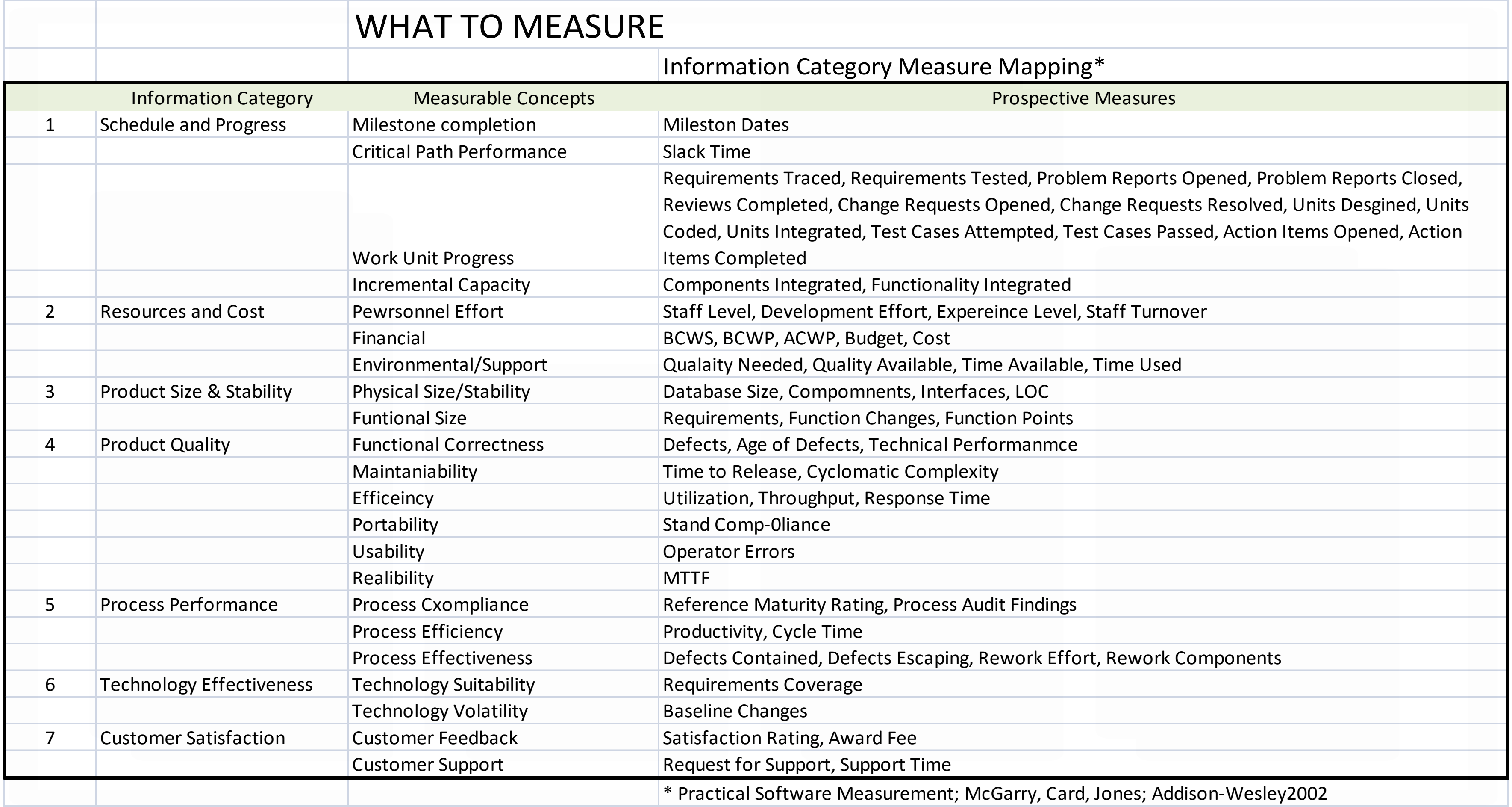
- **Methodology 5: Factor/Complexity approach** based on data generated in early iterations

Scale	Functional Description	Effort Multipliers
- - -	Significantly less functionality to be delivered	0.5
- -	Moderately less functionality to be delivered	0.7
-	Slightly less functionality to be delivered	0.9
=	Functionality equivalent to Increment X	1.0
+	Slightly more functionality to be delivered	1.3
+ +	Moderately more functionality to be delivered	1.7
+ + +	Significantly more functionality to be delivered	2.0

Scale	Complexity Description	Effort Multipliers
- -	Significantly less complex	0.7
-	Slightly less complex	0.9
=	Complexity equivalent to Increment X	1.0
+	Slightly more complex	1.3
+ +	Significantly more complex	1.7

- These initial set of factors came from the environmental factor from traditional software cost models
- Because each Increment is a mini project, use a Rayleigh or simple Beta Curve (such as a 60/50 Beta curve) to phase costs

What to measure



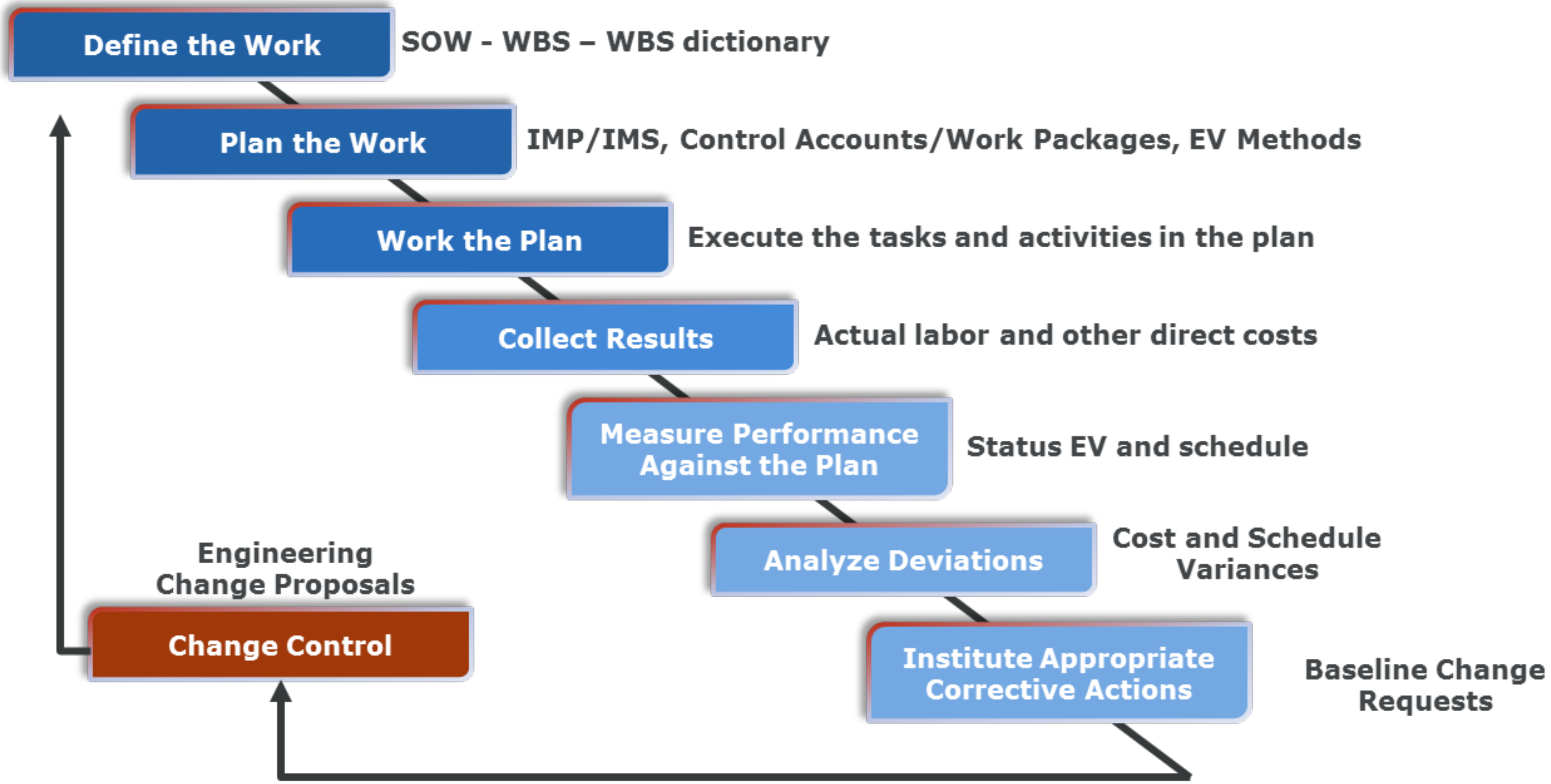
WHAT TO MEASURE

Information Category Measure Mapping*

	Information Category	Measurable Concepts	Prospective Measures
1	Schedule and Progress	Milestone completion	Mileston Dates
		Critical Path Performance	Slack Time
		Work Unit Progress	Requirements Traced, Requirements Tested, Problem Reports Opened, Problem Reports Closed, Reviews Completed, Change Requests Opened, Change Requests Resolved, Units Desgined, Units Coded, Units Integrated, Test Cases Attempted, Test Cases Passed, Action Items Opened, Action Items Completed
		Incremental Capacity	Components Integrated, Functionality Integrated
2	Resources and Cost	Pewrsonnel Effort	Staff Level, Development Effort, Expereince Level, Staff Turnover
		Financial	BCWS, BCWP, ACWP, Budget, Cost
		Environmental/Support	Qualaity Needed, Quality Available, Time Available, Time Used
3	Product Size & Stability	Physical Size/Stability	Database Size, Compomnents, Interfaces, LOC
		Funtional Size	Requirements, Function Changes, Function Points
4	Product Quality	Functional Correctness	Defects, Age of Defects, Technical Performanmce
		Maintaniability	Time to Release, Cyclomatic Complexity
		Efficeincy	Utilization, Throughput, Response Time
		Portability	Stand Comp-0liance
		Usability	Operator Errors
		Realibility	MTTF
5	Process Performance	Process Cxompliance	Reference Maturity Rating, Process Audit Findings
		Process Efficiency	Productivity, Cycle Time
		Process Effectiveness	Defects Contained, Defects Escaping, Rework Effort, Rework Components
6	Technology Effectiveness	Technology Suitability	Requirements Coverage
		Technology Volatility	Baseline Changes
7	Customer Satisfaction	Customer Feedback	Satisfaction Rating, Award Fee
		Customer Support	Request for Support, Support Time

* Practical Software Measurement; McGarry, Card, Jones; Addison-Wesley2002

Earned Value Management Process



Define the Work — SOW - WBS – WBS dictionary

Plan the Work — IMP/IMS, Control Accounts/Work Packages, EV Methods

Work the Plan — Execute the tasks and activities in the plan

Collect Results — Actual labor and other direct costs

Measure Performance Against the Plan — Status EV and schedule

Analyze Deviations — Cost and Schedule Variances

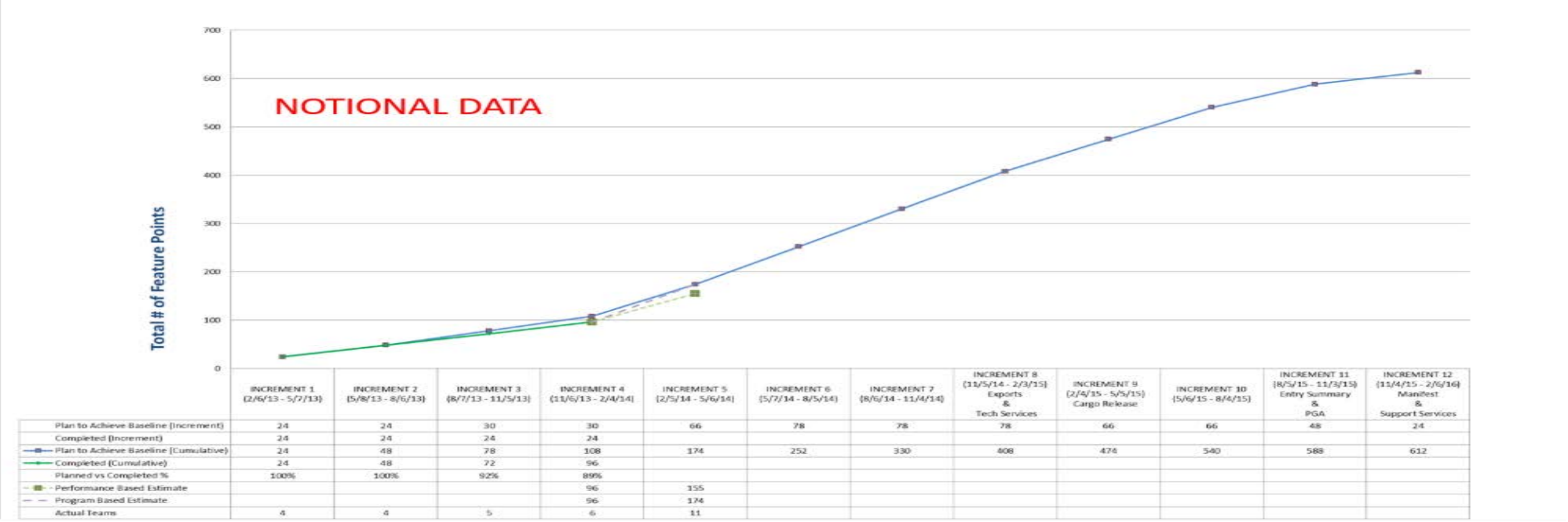
Institute Appropriate Corrective Actions — Baseline Change Requests

Change Control — Engineering Change Proposals

Feature Delivery

GALORATH

Plan for Delivery of Features Versus Actual Delivery (This chart will provide a good "Top Level" assessment)



	INCREMENT 1 (2/6/13 - 5/7/13)	INCREMENT 2 (5/8/13 - 8/6/13)	INCREMENT 3 (8/7/13 - 11/5/13)	INCREMENT 4 (11/6/13 - 2/4/14)	INCREMENT 5 (2/5/14 - 5/6/14)	INCREMENT 6 (5/7/14 - 8/5/14)	INCREMENT 7 (8/6/14 - 11/4/14)	INCREMENT 8 (11/5/14 - 2/3/15) Exports & Tech Services	INCREMENT 9 (2/4/15 - 5/5/15) Cargo Release	INCREMENT 10 (5/6/15 - 8/4/15)	INCREMENT 11 (8/5/15 - 11/3/15) Entry Summary & PGA	INCREMENT 12 (11/4/15 - 2/6/16) Manifest & Support Services
Plan to Achieve Baseline (Increment)	24	24	30	30	66	78	78	78	66	66	48	24
Completed (Increment)	24	24	24	24								
—■— Plan to Achieve Baseline (Cumulative)	24	48	78	108	174	252	330	408	474	540	588	612
—●— Completed (Cumulative)	24	48	72	96								
Planned vs Completed %	100%	100%	92%	89%								
- ■ - Performance Based Estimate				96	155							
— — Program Based Estimate				96	174							
Actual Teams	4	4	5	6	11							

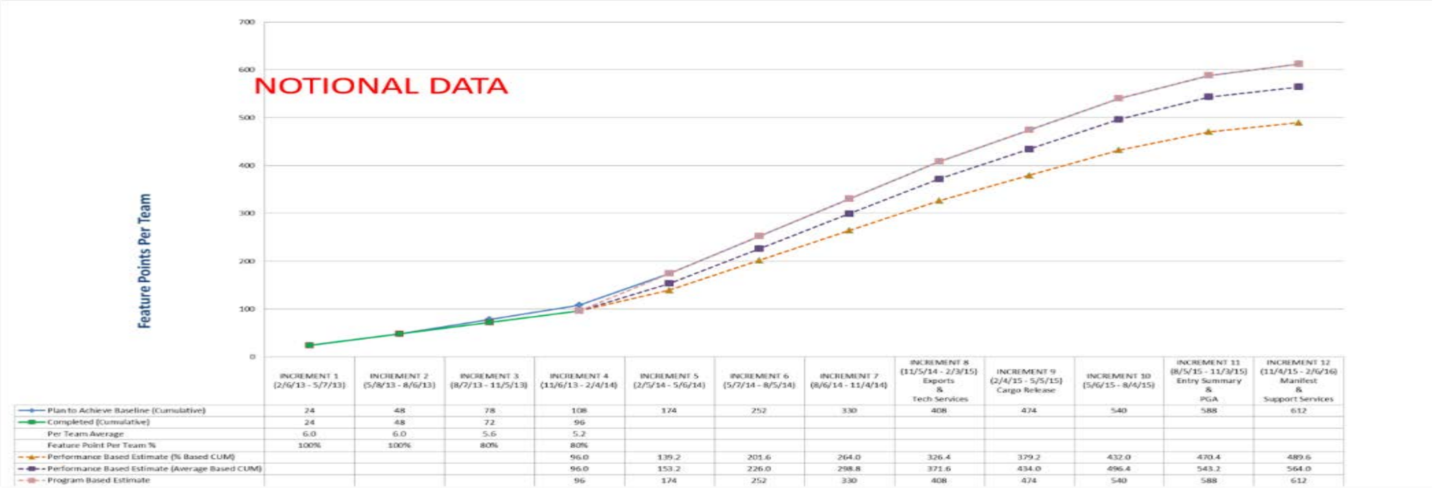
Summary:

- **Use a chart like this and the following two charts for estimating feature velocity/work performance.**

Feature Point Delivery

GALORATH

Feature Point Plan & Performance
(This Chart provided an assessment of “Technical” accomplishment)



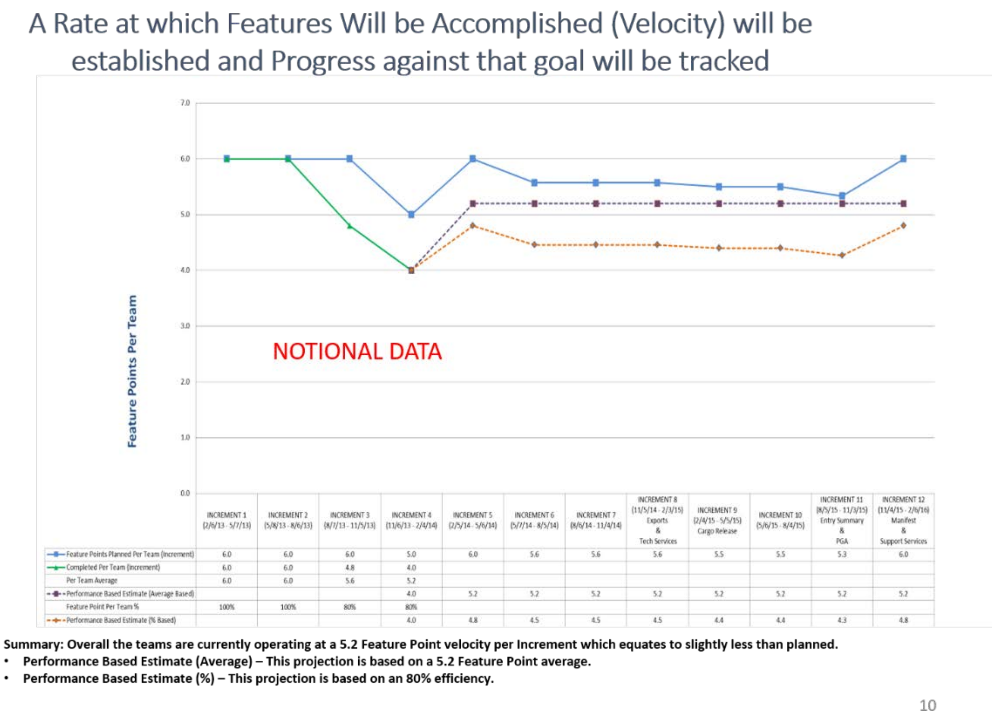
	INCREMENT 1 (2/6/13 - 5/7/13)	INCREMENT 2 (5/8/13 - 8/6/13)	INCREMENT 3 (8/7/13 - 11/5/13)	INCREMENT 4 (11/6/13 - 2/4/14)	INCREMENT 5 (2/5/14 - 5/6/14)	INCREMENT 6 (5/7/14 - 8/5/14)	INCREMENT 7 (8/6/14 - 11/4/14)	INCREMENT 8 (11/5/14 - 2/3/15) Exports & Tech Services	INCREMENT 9 (2/4/15 - 5/5/15) Cargo Release	INCREMENT 10 (5/6/15 - 8/4/15)	INCREMENT 11 (8/5/15 - 11/3/15) Entry Summary & PGA	INCREMENT 12 (11/4/15 - 2/6/16) Manifest & Support Services
◆ Plan to Achieve Baseline (Cumulative)	24	48	78	108	174	252	330	408	474	540	588	612
■ Completed (Cumulative)	24	48	72	96								
Per Team Average	6.0	6.0	5.6	5.2								
Feature Point Per Team %	100%	100%	80%	80%								
▲ Performance Based Estimate (% Based CUM)				96.0	139.2	201.6	264.0	326.4	379.2	432.0	470.4	489.6
■ Performance Based Estimate (Average Based CUM)				96.0	153.2	226.0	298.8	371.6	434.0	496.4	543.2	564.0
■ Program Based Estimate				96	174	252	330	408	474	540	588	612

Summary: Overall the teams are currently operating at a 5.2 Feature Point velocity per Increment which equates to slightly less than planned.

- **Performance Based Estimate (Average) – This projection is based on a 5.2 Feature Point average.**
- **Performance Based Estimate (%) – This projection is based on an 80% efficiency.**
- **Program Based Estimate – This projection is based on team expectations.**

9

Feature Velocity



A Rate at which Features Will be Accomplished (Velocity) will be established and Progress against that goal will be tracked

	INCREMENT 1 (2/6/13 - 5/7/13)	INCREMENT 2 (5/8/13 - 8/6/13)	INCREMENT 3 (8/7/13 - 11/5/13)	INCREMENT 4 (11/6/13 - 2/4/14)	INCREMENT 5 (2/5/14 - 5/6/14)	INCREMENT 6 (5/7/14 - 8/5/14)	INCREMENT 7 (8/6/14 - 11/4/14)	INCREMENT 8 (11/5/14 - 2/3/15) Exports & Tech Services	INCREMENT 9 (2/4/15 - 5/5/15) Cargo Release	INCREMENT 10 (5/6/15 - 8/4/15)	INCREMENT 11 (8/5/15 - 11/3/15) Entry Summary & PGA	INCREMENT 12 (11/4/15 - 2/6/16) Manifest & Support Services
Feature Points Planned Per Team (Increment)	6.0	6.0	6.0	5.0	6.0	5.6	5.6	5.6	5.5	5.5	5.3	6.0
Completed Per Team (Increment)	6.0	6.0	4.8	4.0								
Per Team Average	6.0	6.0	5.6	5.2								
Performance Based Estimate (Average Based)				4.0	5.2	5.2	5.2	5.2	5.2	5.2	5.2	5.2
Feature Point Per Team %	100%	100%	80%	80%								
Performance Based Estimate (% Based)				4.0	4.8	4.5	4.5	4.5	4.4	4.4	4.3	4.8

Summary: Overall the teams are currently operating at a 5.2 Feature Point velocity per Increment which equates to slightly less than planned.

- **Performance Based Estimate (Average) – This projection is based on a 5.2 Feature Point average.**
- **Performance Based Estimate (%) – This projection is based on an 80% efficiency.**

10

Summary

- Fixed Price and/or LOE contracts in the early phases should be written so that key “value-added” metrics are collected and reported during each increment
- Estimators may have to employ a variety of software estimating methodologies within a single estimate to model the blended development approaches being utilized in today’s development environments
  - An agile estimating process can be applied to each iteration/sprint
  - Future Increments can be estimated based on most recent/successful IID performance
- Cost estimators will have to scrutinize these programs like a schedule analyst might to determine the most likely IOC capabilities and associated date
  - The number of increments are an important cost driver as well as an influential factor in uncertainty/risk modeling

Summary

- All of the estimation methods are susceptible to error, and require accurate historical data to be useful within the context of the organization
- When developers and estimators use the same “proxy” for effort, there is more confidence in the estimate

Recommended Reading

- “The Death of Agile” blog
- “Agile Hippies and The Death of the Iteration” blog
- Story Point Inflation

5

Endnotes

- *1, 2, 4, 10, 11:* Larman, C. (2010). *Agile and Iterative Development: A Manager's Guide.*
- *3:* Kilgore, J. (2012). Senior Associate, Kalman & Company, Inc.
- *5, 6, 7, 8:* Agile Alliance. (2012). *Agile Alliance*. Retrieved 2012, from http://www.agilealliance.org
- *9:* Coaching, T. L. (n.d.). Rally Software Scaling Software Agility.
- *12:* Bittner, K., & Spence, I. (2006). *Managing Iterative Software Development Projects.* Addison-Wesley Professional.

Additional References

- Cohn, M. (2009). *Succeeding with Agile Software Development using Scrum.*
- Dooley, J. (2011). *Software Development and Professional Practice.*
- Gack, G. (2010). *Managing the Black Hole.*
- George, J., & Rodger, J. (2010). *Smart Data (Enterprise Performance Optimization Strategy).*
- Royce, W., Bittner, K., & Perrow, M. (2009). *The Economics of Iterative Software Development: Steering Towards Better Business Results.* Addision Wesley Professional.
- Smith, G., & Sidky, A. (2009). *Becoming Agile in an Imperfect World.*

Contact Information

- Bob Hunt
  - Email: BHunt@Galorath.com
  - Phone: 703.201.0651