Agile Estimation: Beyond the Myths

Webinar Sponsored by Computer Aid, Inc.

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Agenda

- Some Key Agile Principles
- What Do We Estimate? The importance of planning a release in agile organizations
- Considerations for estimation:
  - Size
  - Shape of the work
  - Milestones
  - Productivity and Effort
- Project Tracking and Control
- Questions

Some Key Agile Principles

"The fundamental things apply, as time goes by"

By Herman Hupfeld,
made famous in "Casablanca"

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Some values, principles, and practices

(Paraphrased from multiple sources, including the Agile Manifesto, books, talks, articles, and blogs)

- Focus is on value of delivered software
- Develop on cadence (time boxed sprints)
- Embrace change
- Emergent requirements and design
- Working software is the primary measure of progress
- Definition of ready/groom the backlog

Focus is on value of delivered software

- From the Agile Manifesto principles:

  Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

- Value is in the eyes of the consumer

Develop on cadence (time boxes)

- Project team (or teams, when scaling agile methods) choose an iteration or sprint length and stick to it.
- Time-boxing provides the opportunity for frequent feedback and agile direction setting.
- “Sprint length” is the new “month”!

Embrace change

- Or as the Agile Manifesto principles say:

  Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Caution: Welcome change is different from indecision, lack of careful consideration, and churn

Emergent requirements and design

- "Just in time" requirements and design
  - Details of stories are worked out at the time they are developed
- Probably the most important practice to get the benefits of agile methods
  - Enables the team to embrace change

Emergent requirements and design

- The biggest change from waterfall methods
- No “big upfront requirements phase”
- No “big upfront design phase”
- Requires customer/business owner involvement throughout the development life cycle

Working software is the primary measure of progress

- Progress = which stories have been developed, which remain to be developed
- Emphasis on “potentially shippable software” at each iteration
  - Must be able to review the working software to get feedback
- Methodology steps like “architecture approved” or “test plan complete” are not primary

QSM®

Definition of ready/groom the backlog

- The productivity and quality of the development work hinges on having well thought through, carefully understood requirements (albeit, “just in time”).
- The “definition of ready” and the work of getting the stories ready is as important as the “definition of done” and the work to get them developed
- Otherwise, “garbage in, garbage out”

Some values, principles, and practices

(Paraphrased from multiple sources, including the Agile Manifesto, books, talks, articles, and blogs)

- Focus is on value of delivered software
- Develop on cadence (time boxed sprints)
- Embrace change
- Emergent requirements and design
- Working software is the primary measure of progress
- Definition of ready/groom the backlog

What Do We Estimate? The importance of planning a release in agile organizations

"Everything's different, nothing's changed"

From "Company" by Stephen Sondheim

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

What to estimate? NOT the duration of a sprint!

Sometimes there's a misunderstanding because the same word means two different things:

Led in the race

Lead in the race

What to estimate? NOT the duration of a sprint!

- Two very different meanings of the same word, “estimation,” in an agile environment:
  - Sprint level: Decide which stories to commit to defining in detail and developing in the next sprint (which is a fixed length).
    - Often referred to as “agile estimation” in the literature
  - Project Release level: Estimate the time and cost of a project to develop software that meets chosen business goals
    - help decide what projects to do.
    - In some cases, estimating how much functionality can be developed to meet a fixed deadline.

Sprint Level: Develop on Cadence

- Sprint length is the same for all sprints.
- Sticking to this time-box enables quick feedback and just-in-time decision making
- The length of the sprints is CHOSEN, not estimated.

Short term questions apply at each sprint

- At each sprint, we have to think about:
  - What detailed stories should we develop next?
  - Based on feedback, do we have a better understanding of the content or priorities?
  - What did we learn in the last couple of weeks, and how can we apply it?

… ***Next Iteration:***
Choose a return flight
Provide passenger information
Check flight arrival
Buy extra miles for loyalty account
Change the return flight
Choose a seat on a flight
Cancel a reservation

People's intuition is good for sprint decisions

- At the sprint level, the team members can decide which stories will fit in the next sprint
- Developers are very good at knowing what can be done in short bursts

- **The nonlinearities of software development don't surface in a short sprint**

Different questions at different levels

Question	Sprint Level	Release Level
What stories should we develop next?	✓	
What are the details of those stories?	✓	
What needs to change based on feedback?	✓	
What did we learn this sprint?	✓	
How long will it take until we have enough value?		✓
Should we trade off schedule for resources?		✓
Can I expect to reach my goal at a reasonable cost?		✓

What makes software ready to release?

- New and enhanced function that provides sufficient value:
  - "Consumable": has enough functionality and the necessary functionality so that users can use it for the intended purpose
  - "Delivered": has gone through all the final deployment and other preparation needed to actually be used

QSM®

“Release” vs. “Potentially Shippable”

- Don’t we have potentially shippable software that provides new or enhanced functionality at the end of each sprint?
  - We chose the highest priority stories
  - We made sure all the test cases ran
  - We got to the “definition of done” for each of the stories
- “Potentially shippable” allows for good review and feedback.
  - We are heading towards the final goal
  - It doesn’t have to reach the final goal

“Release” vs. “Potentially Shippable”

- Purpose at each sprint is to get feedback to do course corrections and learn
  - Stories were broken down into “developer sized bites” that fit into the sprint. Not all of a higher-level function must be completed
  - Not all the functionality needed to consume and use the software is ready at each sprint. “Highest priority that fits” is not enough for production use
- Only over multiple sprints will the functionality be enough to serve a business purpose for the users
  - You can’t arbitrarily decide on a time box for that!

QSM®

“Release” vs. “Potentially Shippable”

- If you use the Scaled Agile Framework:
- The Scaled Agile Framework (SAFe) v3 no longer uses the term “Potentially Shippable Increment” to avoid this confusion.
- SAFe v3 also introduced a formal notion of “Release” to distinguish the time-boxed development cadence from when the software is ready to be delivered, consumed, and used.
- Releases do not have to occur on the time boxed development boundaries

Other activities must coordinate with release

- Other activities coordinate with a release of the software
  - Marketing and sales if it's a software product
  - Manufacturing if it's included in hardware
  - Process changes for internally used software
- The people and groups responsible need to be able to plan all the project activities

QSM®

People need help for longer term estimates

- People try to extrapolate from what they know they can do in a short term to what can be done in a longer term, like a 6 or 8 month release, but that doesn't work.
- Intuition does not take into account the nonlinearities of software development (software size and duration are not proportional, duration and team size are not inversely proportional, velocity is not constant, etc.)
- People are "wishful thinkers" and don't learn from history

QSM®

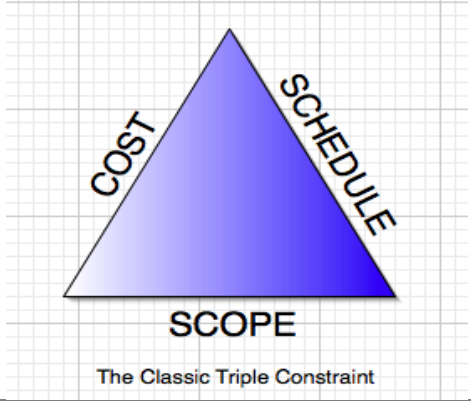
Pe

- P n
  c m,
  li
- I of
  s
  r sely
  p
- People are “wishful thinkers” and don’t learn from history

To account for the nonlinearities of software development , use proven statistical techniques together with your organization’s historical data or industry-wide historical data to offer credible answers to the medium and long term questions.

Myth: “In agile, we decide on schedule not scope”

- No more so in agile projects than any other!
  - Managers have always asked for schedule estimates then treated them as hard and fast commitments.
- Agile and lean thinking emphasize short durations (“small batch size”)
  - Cost of delay
  - Time to value



QSM®

Myth: "In agile, we decide on schedule not sc

- N
- A
  (

Whether schedule or scope dominates the choices and tradeoffs depends on business goals

- *Sometimes there is an absolute deadline*
- *Sometimes there is a mandate on scope (e.g. regulated content)*
- *Most often, there are tradeoffs to consider*
- *As the project progresses and you get feedback at each sprint, you may adjust scope, schedule, cost or all three!*

The Classic Triple Constraint

Lean Thinking: Minimal Viable Release

- Minimal
  - Cost of delay can offset savings in the cost of development
  - Quick feedback
- Viable
  - If it doesn't contain the features they need, users won't use it: must be feature rich enough to consume
  - "Value is in the eye of the beholder"

What makes a release viable?

- Depends on the business goals and the needs of the consumers
- The “most important function” may not be the “only important function”
- Users don’t want to change unless there’s enough new and improved function
- Must be “user sized” not “developer sized bites”

Ladies and gentlemen, the new release of our navigation software worked flawlessly. We will be able to land in a few months when the next release comes out.

In summary

- Release must provide consumable value
- Unlike sprints, projects to develop a releases are not always time-boxed.
- Team members should make sprint level decisions, but estimating a release requires tools and methods that take into account:
  - Overall goals: Balance “minimal” with “viable”
  - Top down estimate of the release as a whole rather than extrapolation from small chunks
  - Organizational or industry-wide historical data
  - Statistical methods that understand the nonlinearities of software development

Size

"Is it bigger than a breadbox?"

Steve Allen, on "What's My Line"

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Size is the key to estimation

- QSM educates our customers that software size is the key input to a credible estimates.
- Companies using waterfall methods often tried to “guesstimate” the duration of the various waterfall phases and add them up
- Agile methodologists have recognized that size is the key!

QSM®

User Stories are the input to size

- Most agile teams use “user stories” to specify scope.
- “Story” was invented to bypass all the formality that was creeping into requirements management methodologies
  - If you can’t summarize a piece of functionality on a 3 x 5 card, you don’t really understand what you want.

Epics, Stories, and Everything In Between

- Look at the following backlog items:
  - “Select an available seat for your flight”
  - “Look up the current departure gate for your flight”
  - “Plan a trip”
- Is “Plan a trip” a story?
  - Sort of, but it’s a really big, really vague story
  - It MUST be broken down to be understood, not just to fit into a sprint
  - The term “Epic” was introduced to describe this

QSM®

Epics, Stories, and Everything In Between

- Epics and stories, like any other way to express scope, are hierarchical! Each level breaks down the stories in the level above it.
- It's a multi-level hierarchy.
  - How many levels depends on the particular function
  - Some stories break down to more levels than others until you get to "developer sized bites"
  - User sized bites may be at any level

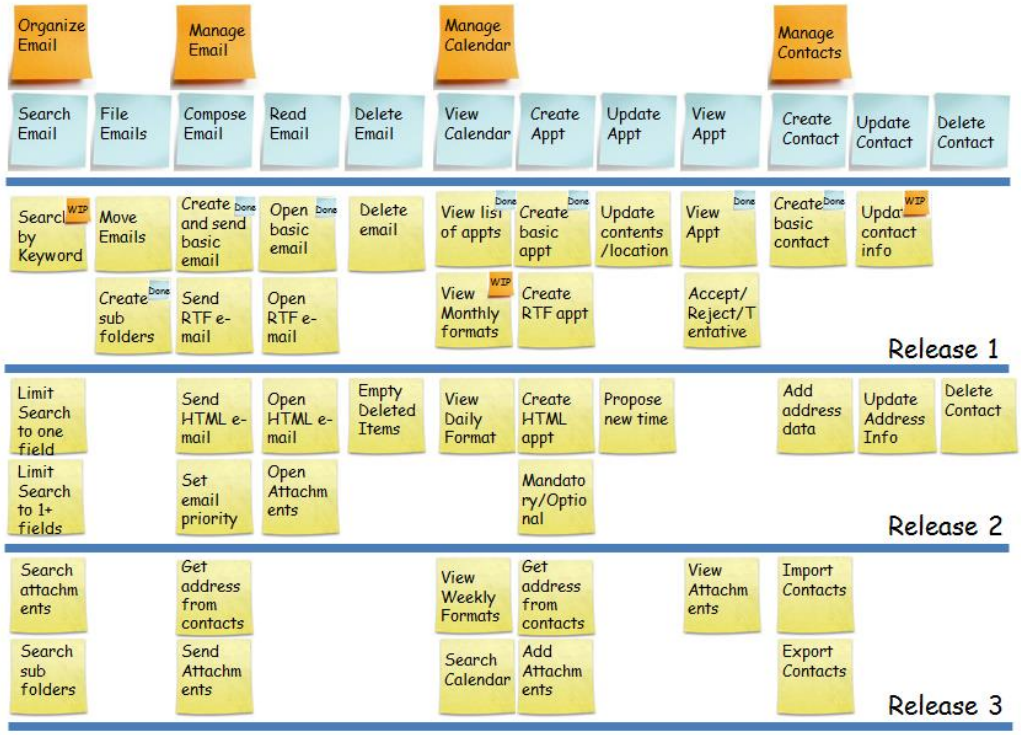
Epics, Stories, and Everything In Between

- B
- T

The developer sized bites selected for a single sprint are closer to the same size than the original stories. But the "user sized bite" of the original story may vary a lot, and development may span sprints.

Ways of showing the hierarchy

- User Story Maps (Jeff Patton)



Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

Ways of Showing the Hierarchy

- Minimal Marketable Features
  - Mark Denne and Jane Cleland-Huang
  - “Software by Numbers”
  - Prioritize and schedule larger scale features and their dependencies for overall value
- Traditional requirements hierarchy and trace relationships

Ways of Showing the Hierarchy

- M
- T
  r

The hierarchy emerges as the project progresses. At any point where you need to measure size for an estimate, some levels will be known but others will be broken out later.

You may only know epics. Some may already be broken down more concretely. The stories will likely be of widely varying sizes, which will be broken out further as the sprints progress.

QSM®

Size and duration: a complicated relationship

- Size is a key input to an estimate
  - We will discuss in a few minutes ways of measuring size appropriate for agile projects
- Seems obvious: The bigger the size, the more there is to get done, so the longer it will take
- But the relationship between size and duration is NOT a simple one.

Duration is not the only input

- Consider a “bake off”:
- Two teams are asked to build exactly the same system
- Team 1 has 10 of your best programmers
- Team 2 has 5 junior programmers
- Will they take the same amount of time?

Duration is not the only input

- C
- T
- T
- T
- V

We will discuss the issues of effort and productivity later in this webinar.

For now, though, let's look closer at the relationship between duration and size.

QSM®

Size and duration– a nonlinear relationship

- So another “bakeoff”:
- A single team is asked to build two different systems
- The first is twice the size of the second
- Since the team is the same, will the first take twice as long?

QSM®

Size and duration– a nonlinear relationship

- So another “bakeoff”:
- A single team is asked to build two different systems
- The first is twice the size of the second
- Since the team is the same, will the first take twice as long?

NO!

QSM®

Size and duration– a nonlinear relationship

- The larger system will take longer, but less than twice as long!
- There is an “economy of size”, and the amount produced by the team varies over the duration of the project.
- In terms often used by agile teams:

Velocity is not constant

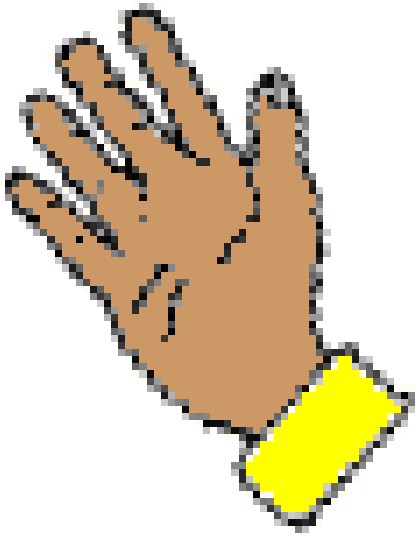
An Old Kid's Joke

- The first kid says to the second kid, "What's this?"

- The second kid says, "I don't know"
- The first kid then replies….

An Old Kid's Joke

- "I don't know either, but here come five of them.



An Old Kid's Joke

- "I don't know either, but here come five of them.

Moral of the story: Sometimes we can know how many we have, even if we don't know what one of them is!

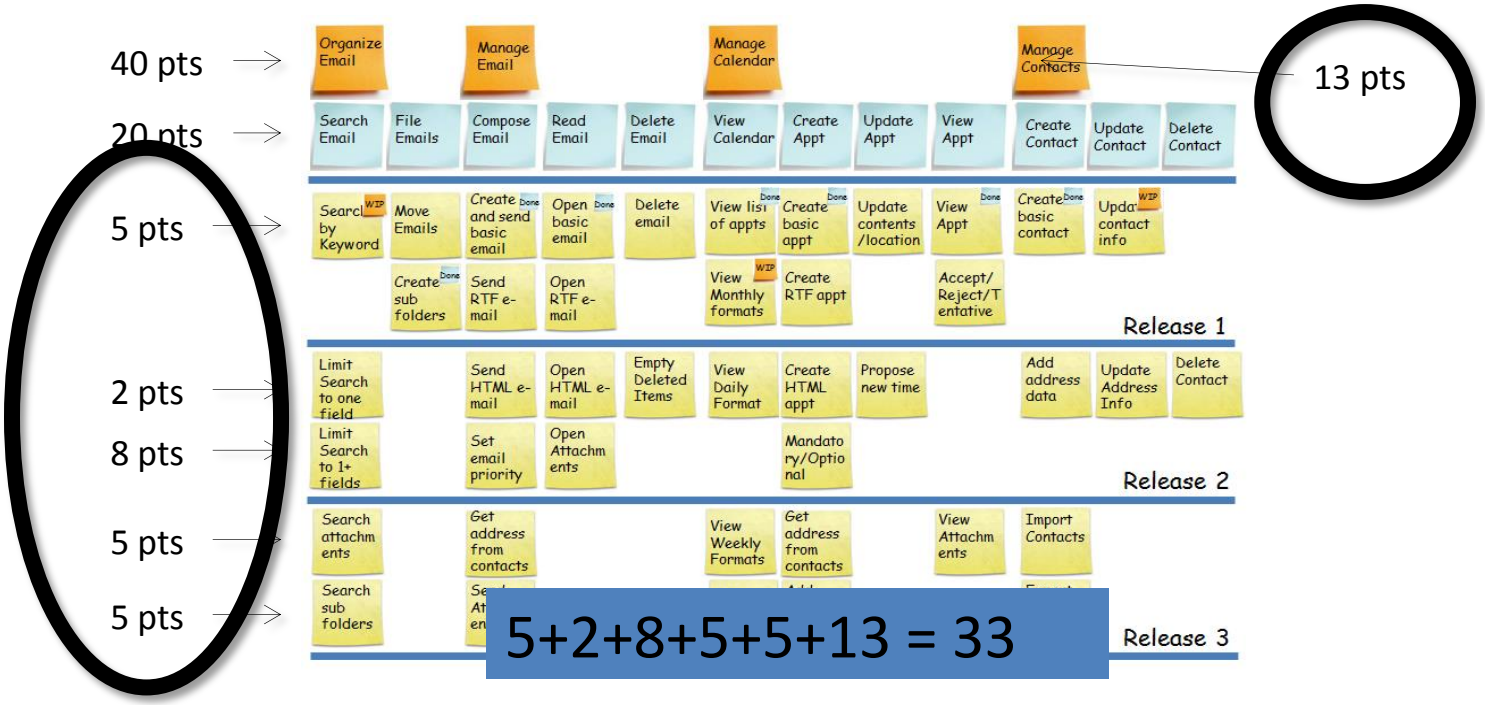
QSM®

Function Points and Story Points

- Both are used to measure the size of software scope
- We can count how many we have without knowing what "one" is
- They differ in two very significant ways:
  - Function point counting is standardized by IFPUG
  - They require different descriptions of the scope

Using Story Points

- CAREFUL: Don't double count stories that are broken out a multiple levels. But you don't have to use the same level in the hierarchy for each story



QSM®

Normalizing Size Measures

- In order to use the size measure for an estimate, we have to be able to compare the size of the system we're estimating to the size of systems already developed.
- Since "a story point" doesn't have a definition, how do we know that the rating from one project compares to the rating of another?
  - This is the same issue that researchers in many fields face, called "inter-rater reliability". It's debated in the agile literature as "normalizing story points"

Four Levels of Normalization

- A team working on a project.
  - At the beginning of each sprint, they rate the stories currently at the top of the backlog in story points. They are consistent from sprint to sprint.
- A team, across projects
  - Sometimes a team will stay together from one project to another. They can consistently rate size of stories across those projects.

Four Levels of Normalization

- Within a company
  - All teams within the company rate stories consistently. Then as you capture more and more history within your company, you can use this consistent rating to get the size of new projects.
- Across the industry
  - This would allow us to benchmark your project against similarly sized projects across the industry.

Four Levels of Normalization

- V
- A

Unlike story points, Function Points are normalized at the industry level. The International Function Point Users Group (IFPUG) standardizes and certifies people in function point counting.

as

rly

os

someday there will be, but there isn't right now.

QSM®

Three Sizing Techniques

- No sizing technique is perfect. Each has its merits and problems.
- Three techniques for agile releases:
  - Measure using function points
  - Measure using story points
  - Count stories

Measure using function points

- Industry standard
  - Can compare to other projects in your company using your own history
  - Can compare to thousands of projects industry wide
- Have to look at the scope as a whole
  - You do not count the “function points for each story” and add them up
- Captures effect of shared code better than other methods
- Emergent design makes it difficult to count function points completed at each iteration

Measure using function points

- I
- H m
- C
- E
  completed at each iteration

You may be interested in an article, "Counting Function Points for Agile: Iterative Software Development" by Carol Dekkers in the 2014 QSM Software Almanac. Go to www.qsm.com to request a copy.

Measure using Story Points

- Fits with what the team will be doing sprint by sprint and agile planning and tracking tools
- Measures the stories on the backlog directly, no additional translation of scope required
- Story points completed and story points remaining (“burndown chart”) can be captured each iteration
- Major issue: Requires the “company” level of normalization to be useful for estimation
  - Must invest in obtaining inter-rater reliability
  - Only requires project team level for the sprint decisions

QSM®

Count Stories

- Easiest to do
- Does NOT take into account different sized stories
  - You have to be very lucky for the sizes of the stories to average out, especially when they haven't been broken down to similar "developer sized bites"
- Can be counted at the end of each iteration
  - But hard to compare to the number of developer sized bites already developed to the larger stories remaining on the backlog.

Shape of the Work

"Everything's different, nothing's changed, only maybe slightly rearranged"

From "Company" by Stephen Sondheim

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

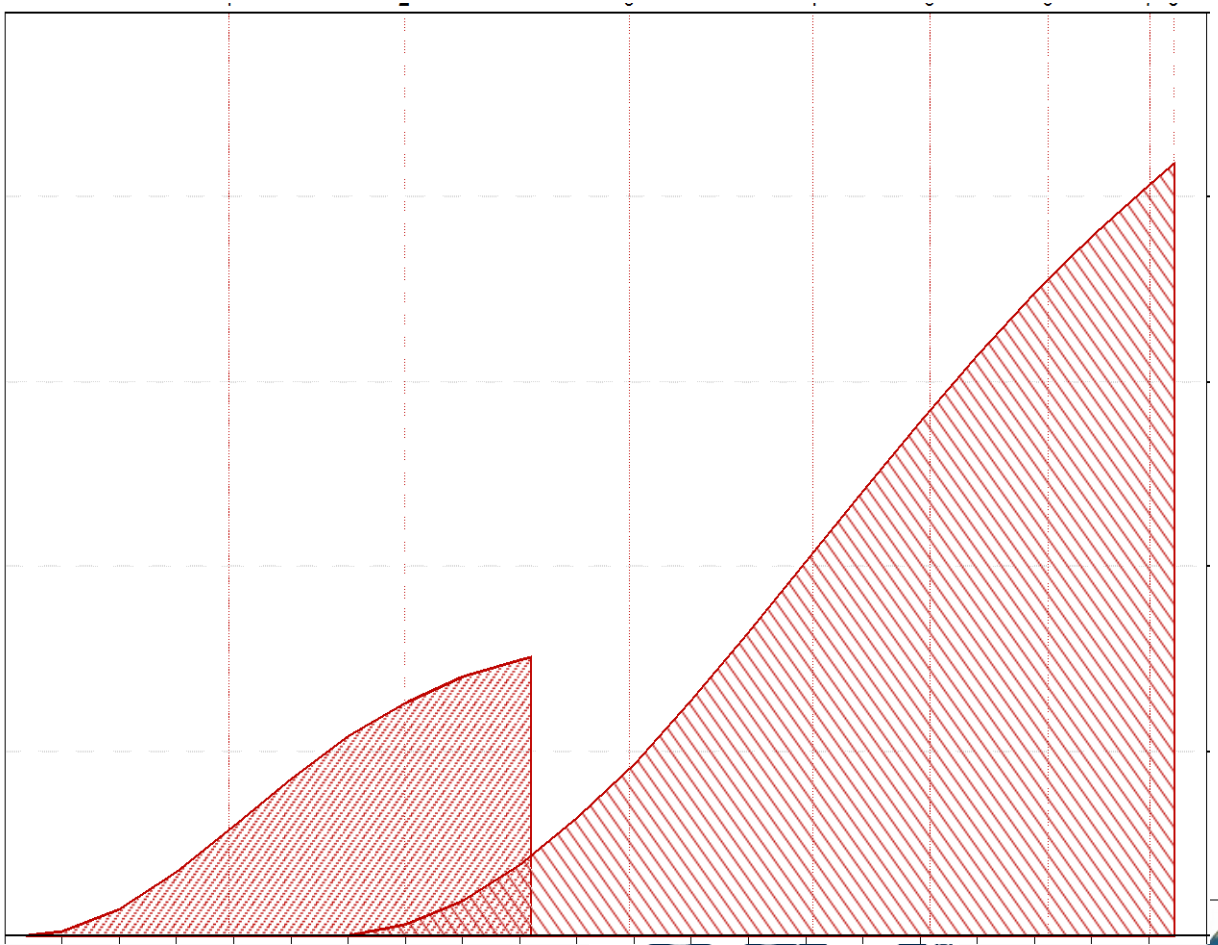
When is a phase not a phase?

- Biggest difference in scheduling agile and waterfall projects:
  - In waterfall projects, different types of work (e.g. requirements definition, coding, testing) happen largely at different times
  - In agile projects, these different types of work happen much more concurrently

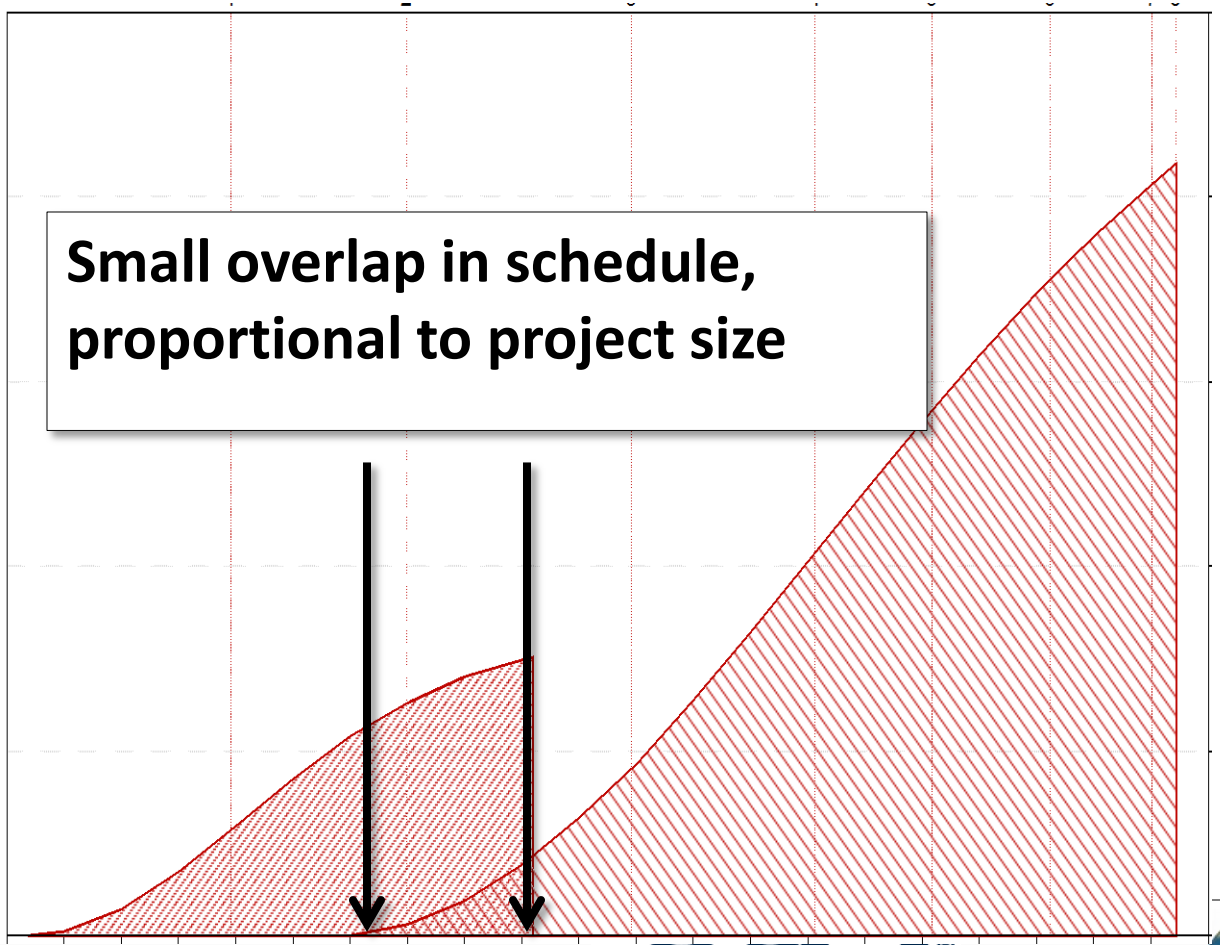
Two key types of work

- Story Writing (well, maybe “Story discussion”)
  - “Groom the backlog”: Choose and prioritize the stories that will be developed, refine them into a hierarchy from high level business value to “developer sized bites”
  - Hold the conversations needed to flesh out the critical details
  - Detail constraints and basic architecture that’s an input to the development work
  - Cooperative effort of product manager, development team, and business representatives
- Development/Test
  - Coding, testing, and other work to get to the definition of done on the defined stories

Typical Waterfall Shape



“Big Upfront Requirements Phase”



Req. and Design

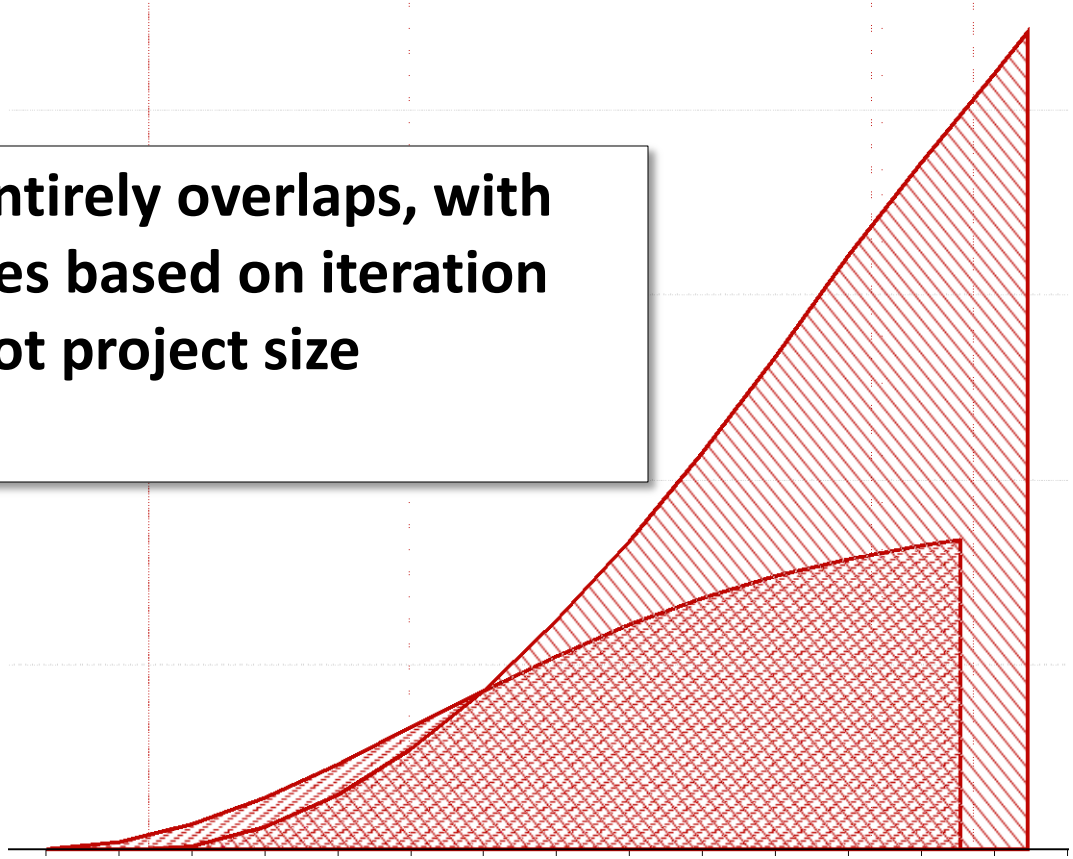
Build and Test

QSM®

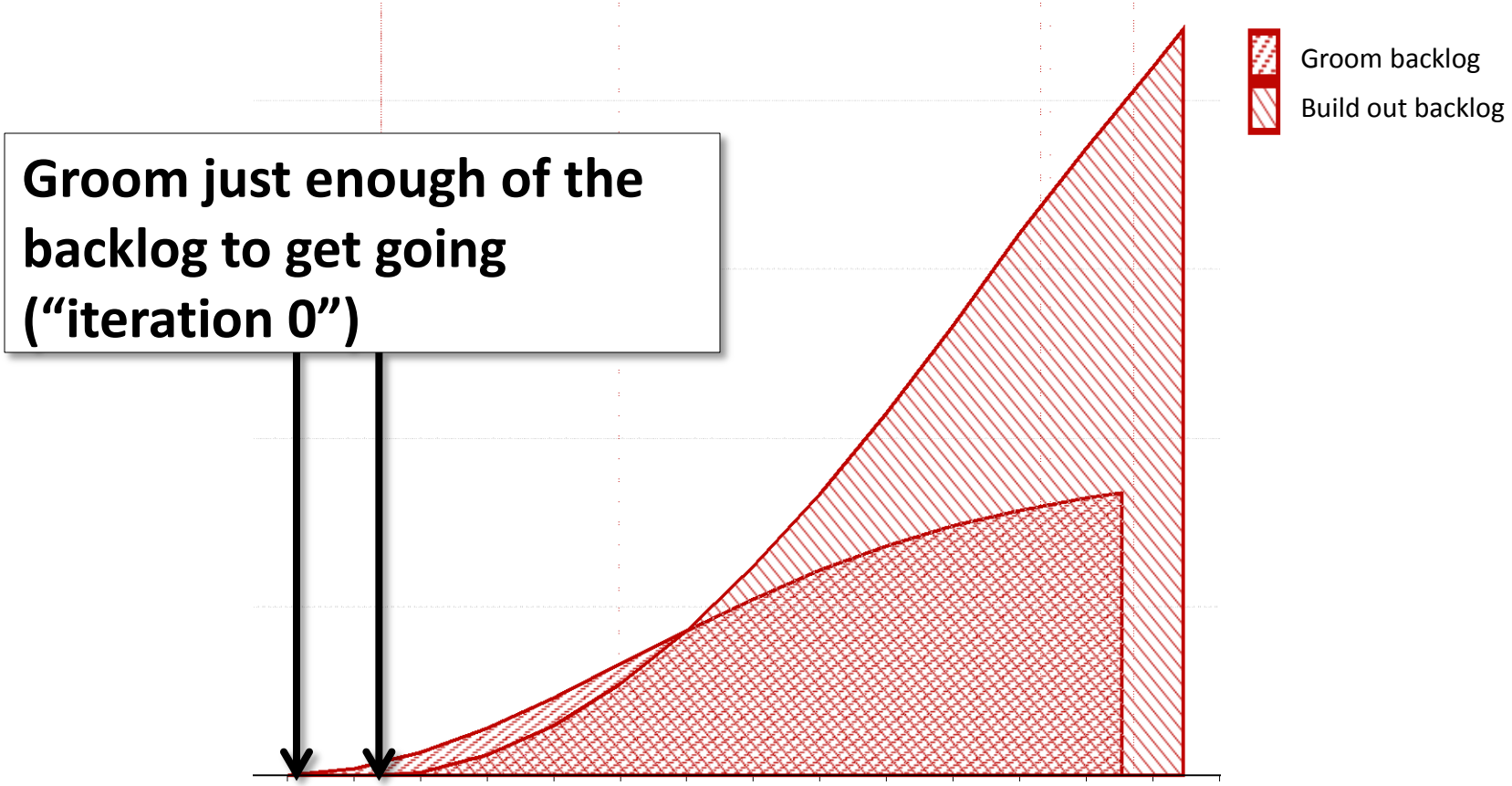
CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Typical Agile Shape

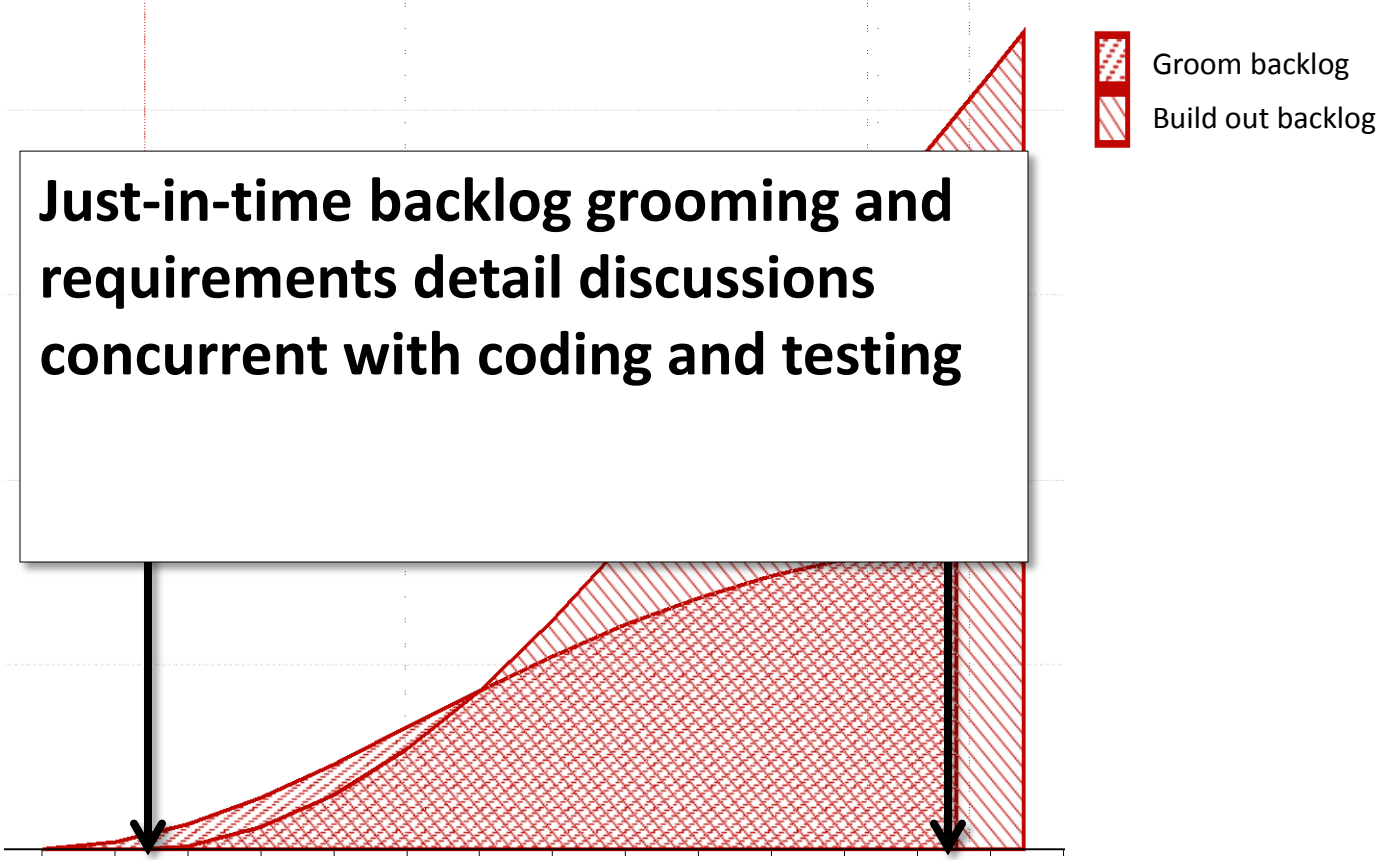
Almost entirely overlaps, with differences based on iteration length, not project size



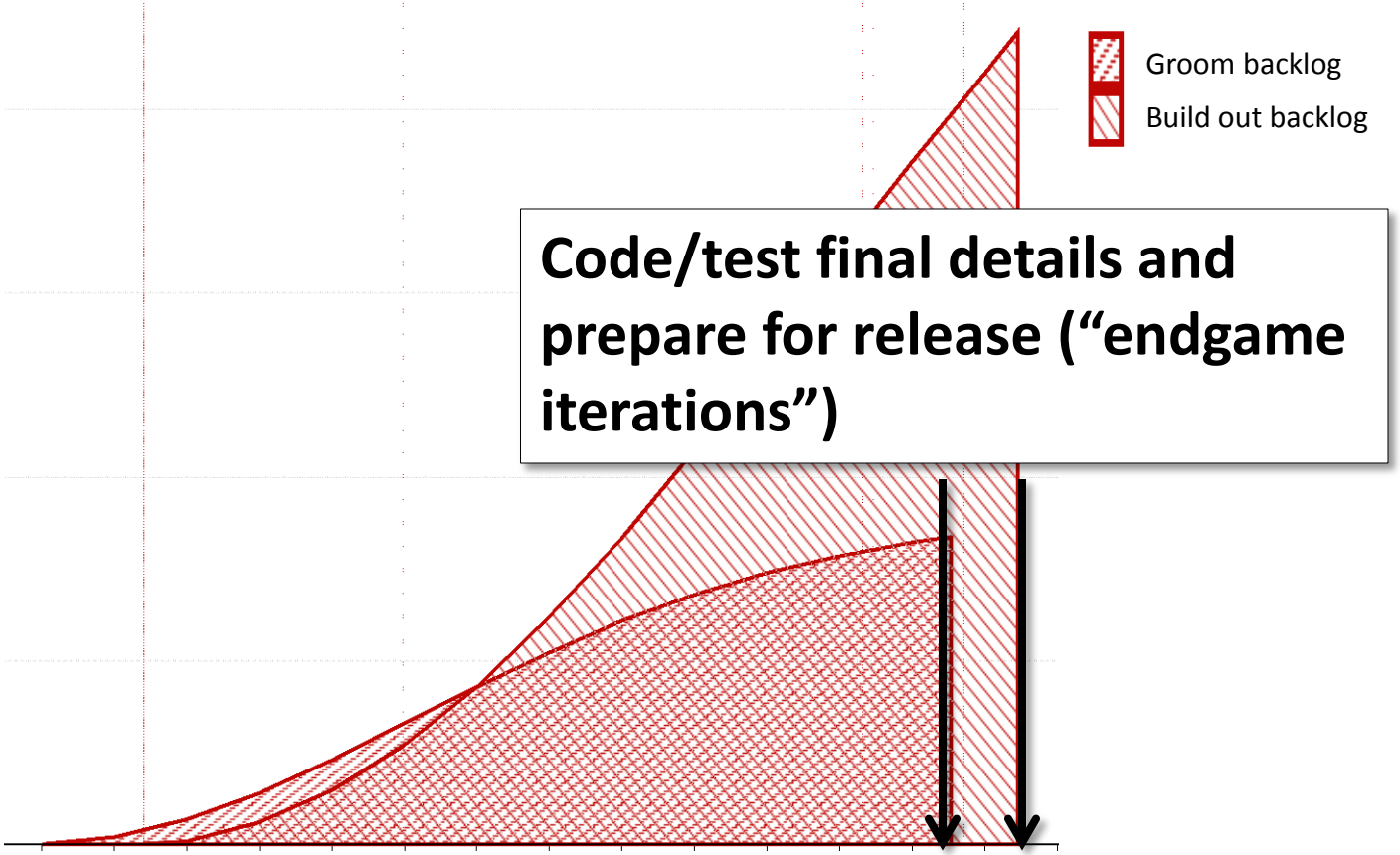
Emergent Requirements



Emergent Requirements



Emergent Requirements



Milestones

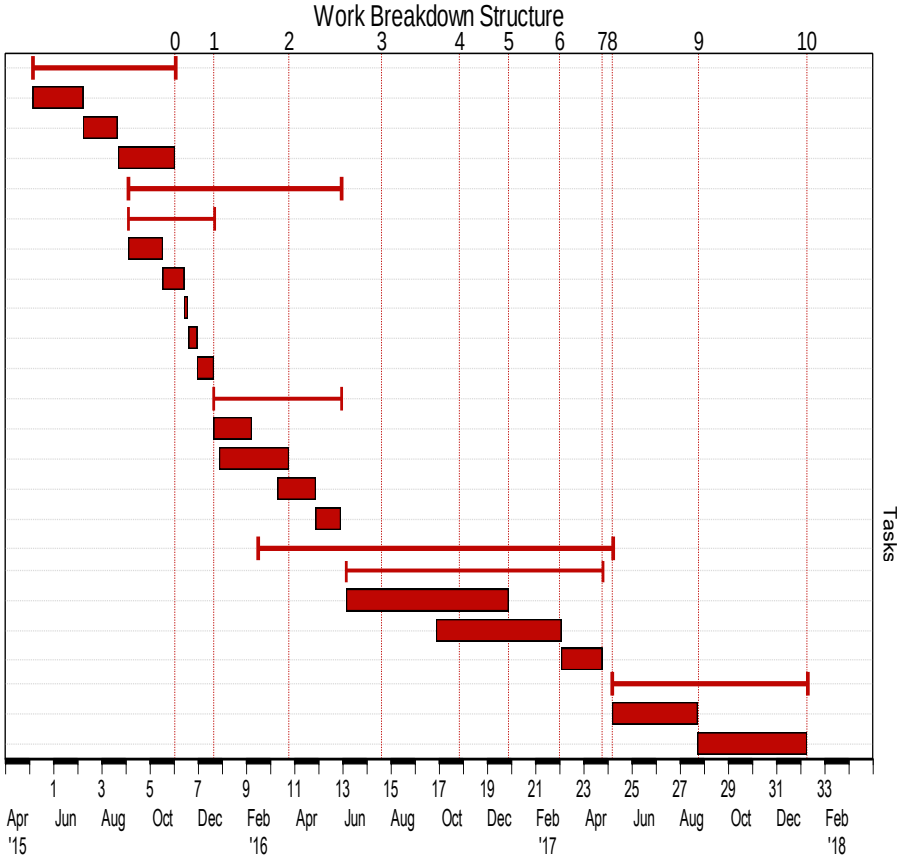
"Life isn't a matter of milestones, but of moments"

Rose Kennedy

QSM®

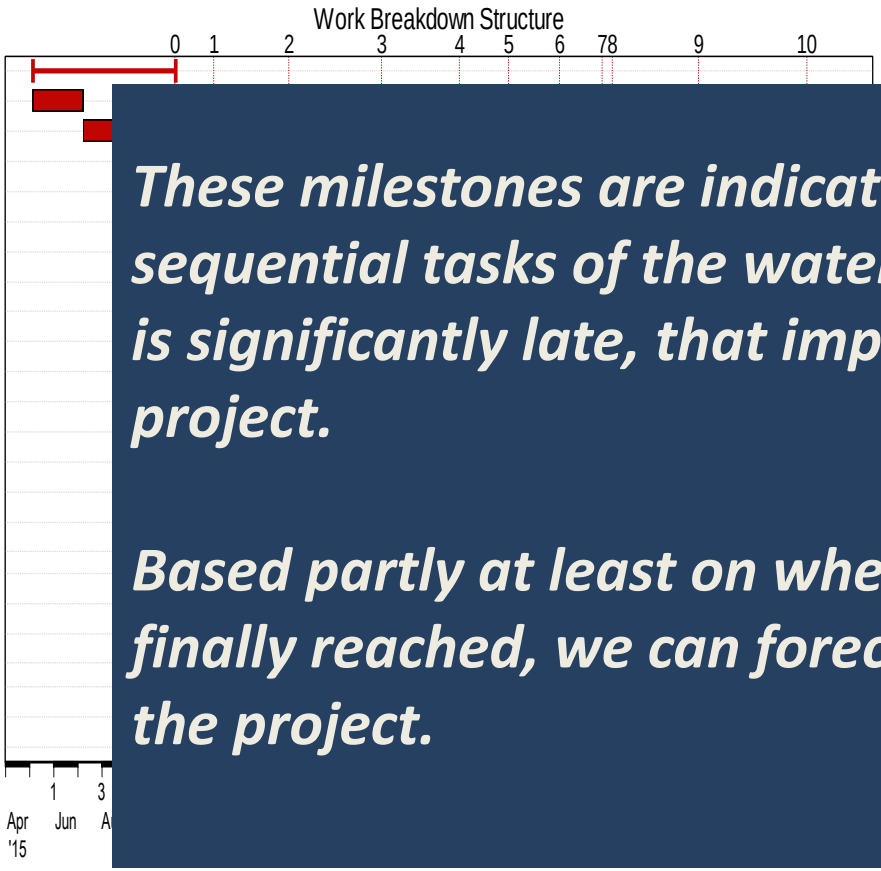
CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Waterfall is managed through milestones



Milestone ID	Milestone
0	Concept Sufficiency Review
1	Software Requirements Review
2	High Level Design Review
3	Low Level Design Review
4	Code & Unit Test Complete
5	Integration Complete
6	System Test Complete
7	User Acceptance Test Complete
8	First Customer Release
9	99% Defect Free
10	99.9% Defect Free

Waterfall is managed through milestones



These milestones are indicators of progress on the sequential tasks of the waterfall project. If a milestone is significantly late, that impedes the progress of the project.

Based partly at least on when those milestones are finally reached, we can forecast a new delivery date for the project.

What makes a milestone useful?

Task Name	Start	Finish	Actual Start	Actual Finish	% Complete
On Time and Budget	**Thu 1/1/15**	**Thu 12/31/15**	**Thu 1/1/15**	**NA**	**67%**
Winter	**Thu 1/1/15**	**Tue 3/31/15**	**Thu 1/1/15**	**Tue 3/31/15**	**100%**
January	Thu 1/1/15	Sat 1/31/15	Thu 1/1/15	Sat 1/31/15	100%
February	Sun 2/1/15	Sat 2/28/15	Sun 2/1/15	Sat 2/28/15	100%
March	Sun 3/1/15	Tue 3/31/15	Sun 3/1/15	Tue 3/31/15	100%
Spring	**Wed 4/1/15**	**Tue 6/30/15**	**Wed 4/1/15**	**Tue 6/30/15**	**100%**
April	Wed 4/1/15	Thu 4/30/15	Wed 4/1/15	Thu 4/30/15	100%
May	Fri 5/1/15	Sun 5/31/15	Fri 5/1/15	Sun 5/31/15	100%
June	Mon 6/1/15	Tue 6/30/15	Mon 6/1/15	Tue 6/30/15	100%
Summer	**Wed 7/1/15**	**Wed 9/30/15**	**Wed 7/1/15**	**NA**	**67%**
July	Wed 7/1/15	Fri 7/31/15	Wed 7/1/15	Fri 7/31/15	100%
August	Sat 8/1/15	Mon 8/31/15	Sat 8/1/15	Mon 8/31/15	100%
September	Tue 9/1/15	Wed 9/30/15	Tue 9/1/15	NA	0%
Fall	**Thu 10/1/15**	**Thu 12/31/15**	**NA**	**NA**	**0%**
October	Thu 10/1/15	Sat 10/31/15	NA	NA	0%
November	Sun 11/1/15	Mon 11/30/15	NA	NA	0%
December	Tue 12/1/15	Thu 12/31/15	NA	NA	0%

What makes a milestone useful?

- It must be able to be missed! (Early or late)
- It indicates a scheduling issue down the line:
  - Lagging indicator: When it's late, other tasks that needed to have started are already late
  - Leading indicator: Missing it doesn't prevent other tasks from starting, but there will be trouble further down the road

QSM®

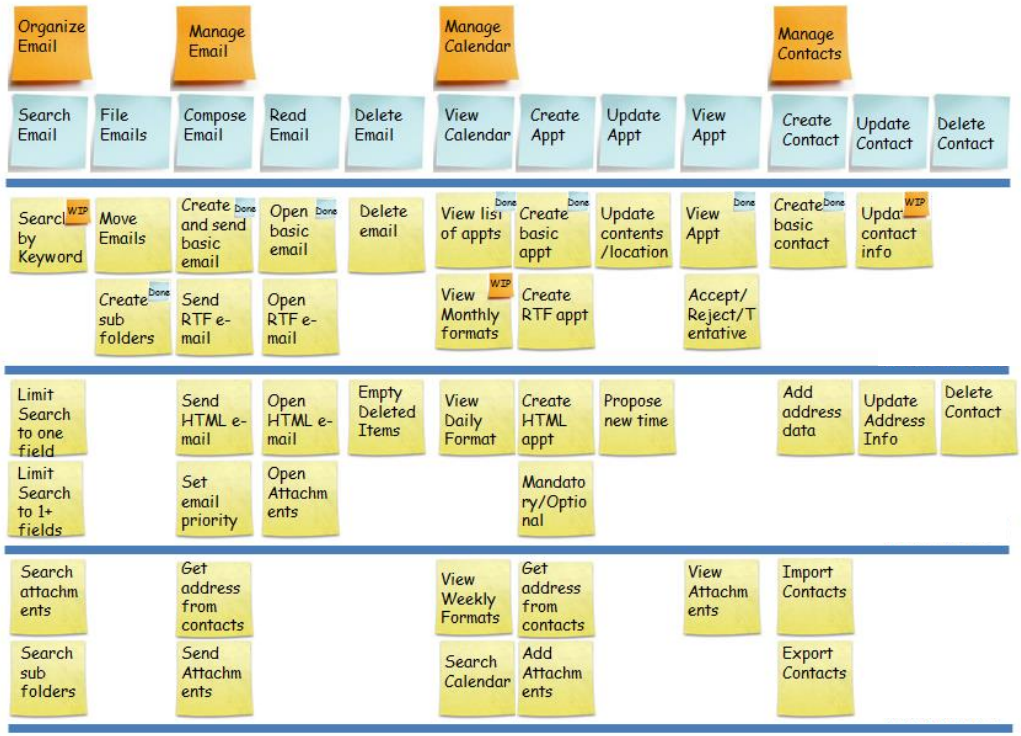
In agile projects, working software is the measure of progress

- Sprint end does not work as a milestone:
  - It's timeboxed! It can neither end early nor late!
  - "Sprint is the new month"
- Since most types of work are done concurrently in an agile project, there are few "intermediate" deliverables that act as milestones
- We measure progress by the size of the stories that meet the "definition of done" at each sprint

What can we use for agile milestones?

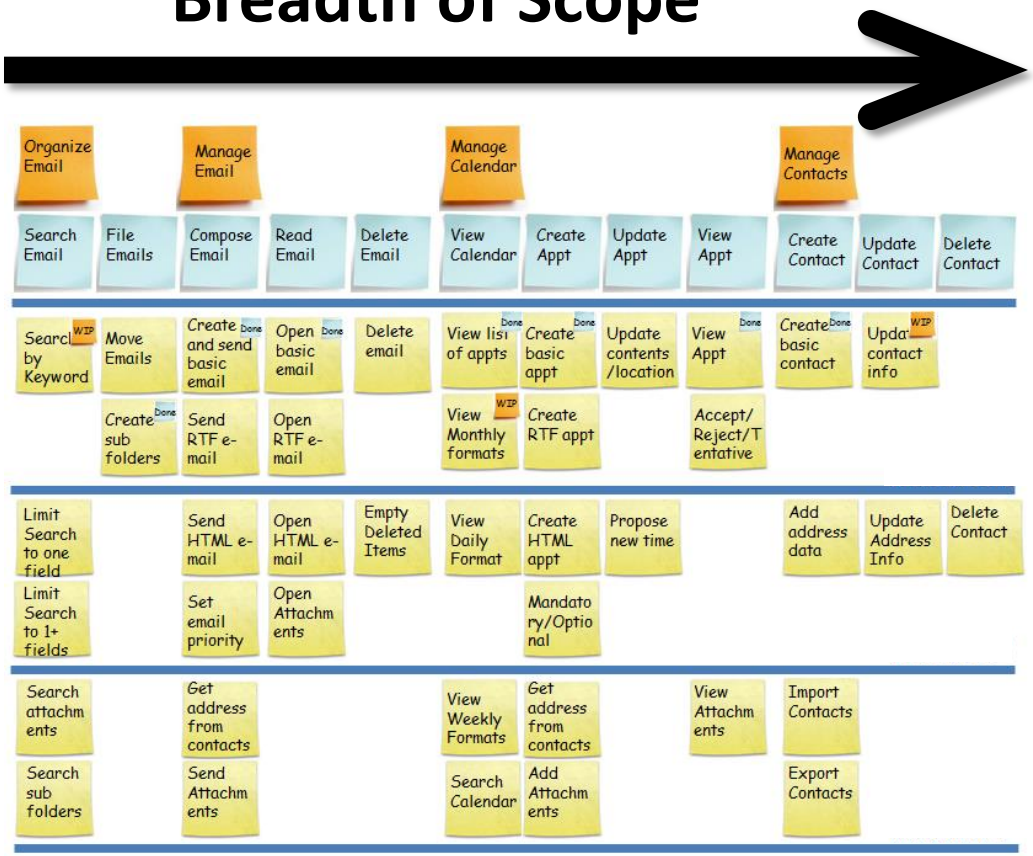
- We have found a useful milestone:
  - "Minimal Viable Features Stable"
    - The upper levels of the user story map are stable
    - Leading indicator
- It's about the work of grooming the backlog!
- Note: The name may vary based on your terminology

User Story Map (or other representation)



Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

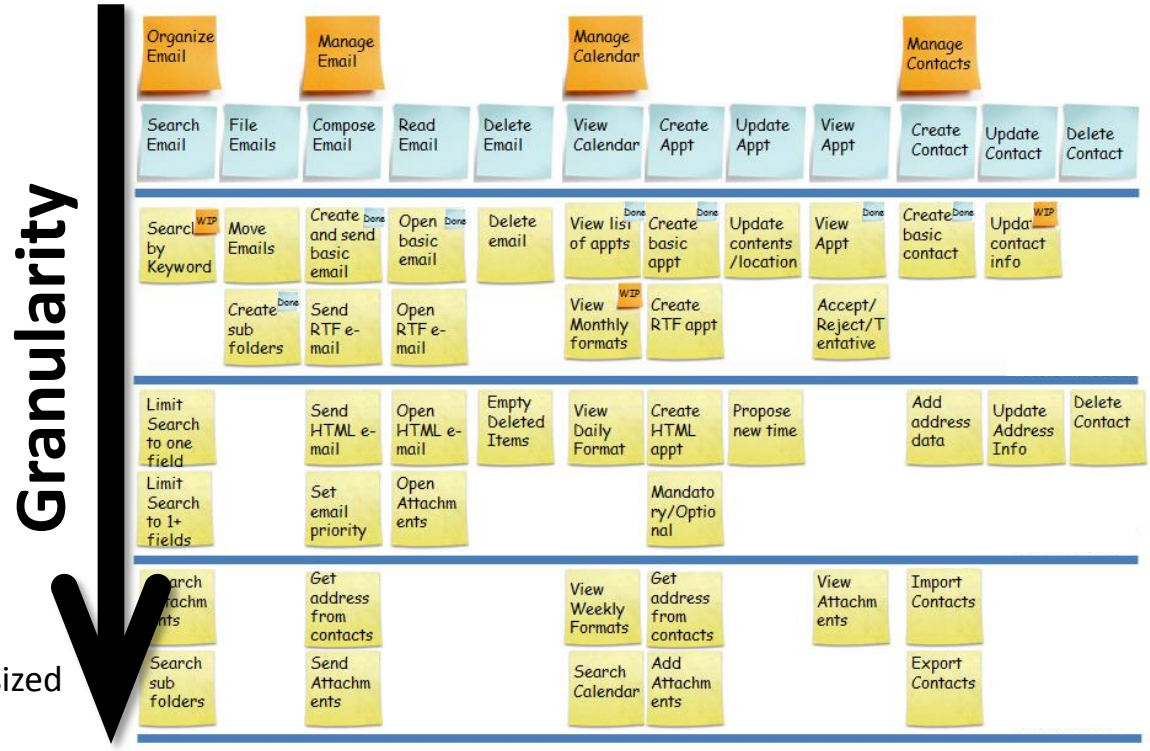
User Story Map (or other representation)



Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

User Story Map (or other representation)

User-sized epics



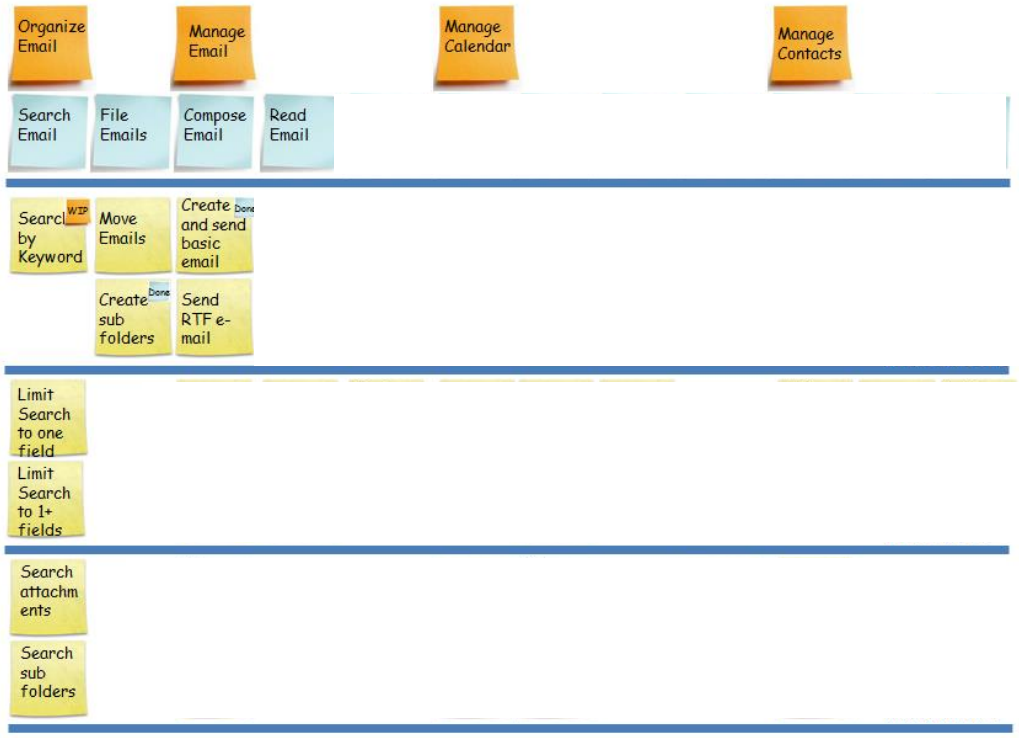
Developer-sized bites

Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Story map emerges as project progresses

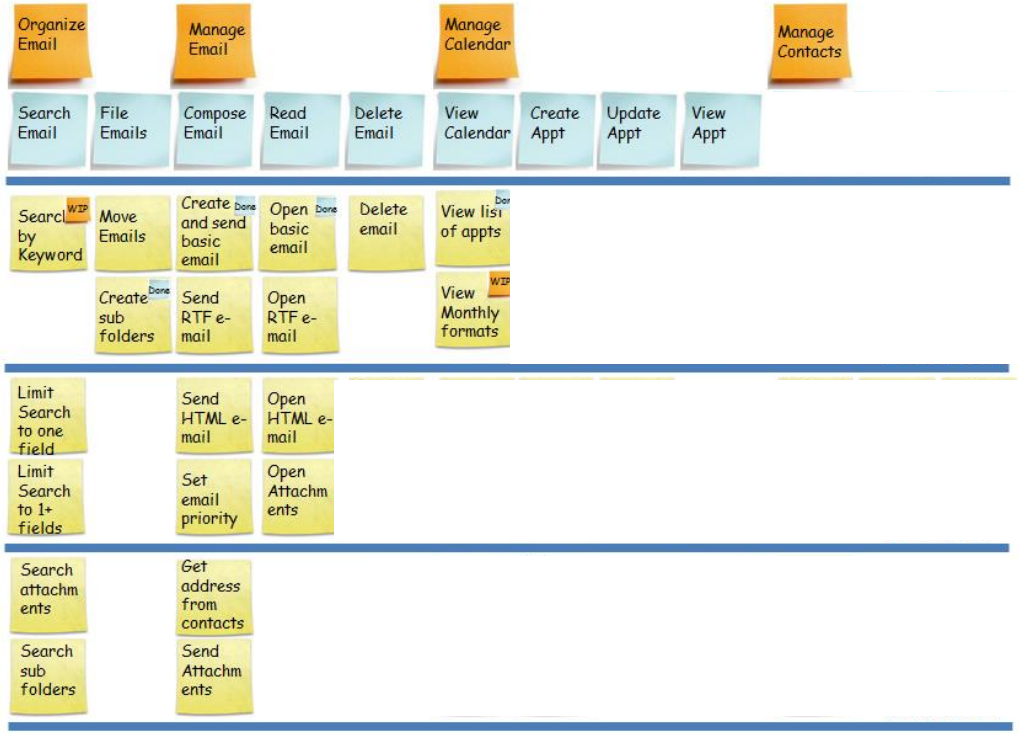


Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

QSM®

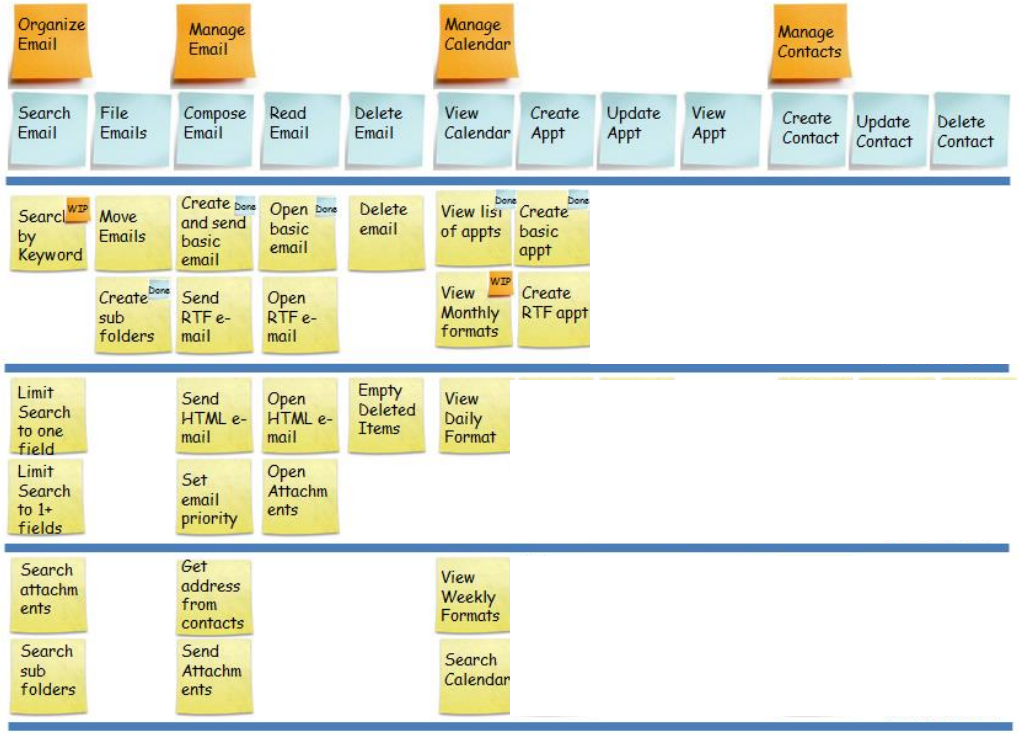
CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

A few sprints later, more has emerged



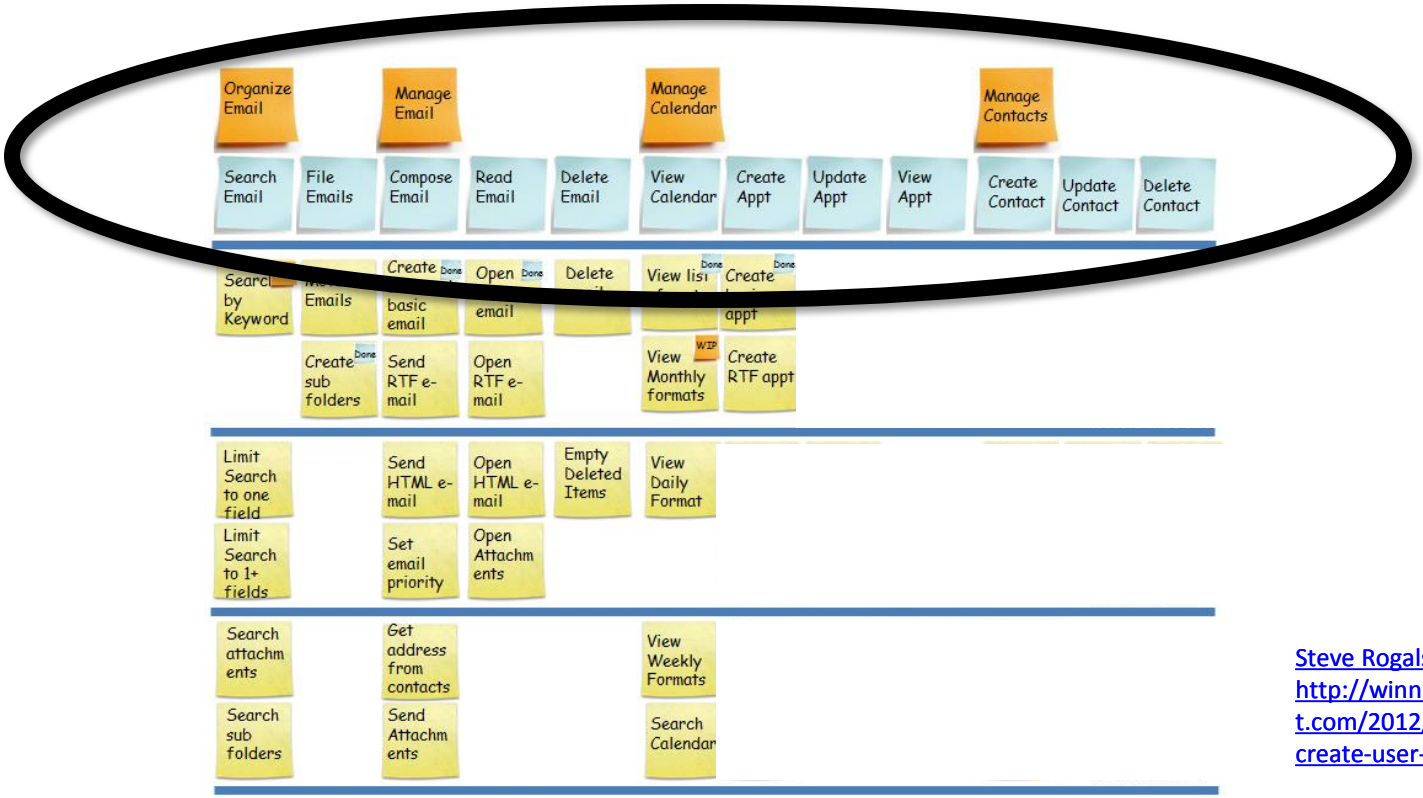
Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

And still later…



Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

Minimal Viable Features Stable



Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Minimal Viable Features Stable

The upper and middle levels should settle in somewhat early while the joint work of story definition and story development are proceeding along sprint by sprint.

Otherwise, it will slow down progress later in the project. You will churn on what should be in those upper levels, and thus what leaves on the tree need to be developed.

.blogspo
o-
.htm

There's still more grooming to do

Steve Rogalski
http://winnipegagilist.blogspot.com/2012/03/how-to-create-user-story-map.htm

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

QSM®

What about “embrace change”?

- Change is good. Churn is not.
- The goal is that the upper levels are stable, but still subject to some change as we discover and learn from completed software
- If the upper levels keep churning, you can still continue developing
  - It’s not a lagging indicator
- Completed work based on lower levels will be thrown out and new details added limiting progress to completion

Productivity and Effort

"The bearing of a child takes nine months, no matter how many women are assigned"

From "The Mythical Man-Month" by Fred Brooks

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

QSM®

The Mythical Person-Month

- While duration clearly varies with team size, it's tempting to think that effort is constant.
- Fred Brooks ("The Mythical Man-Month") pointed out that adding more people to a late project often makes it later!
- Larry Putnam, Sr. derived the "Software Production Equation" that says there is a fourth power relationship when you trade people for time

QSM®

The Mythical Person-Month

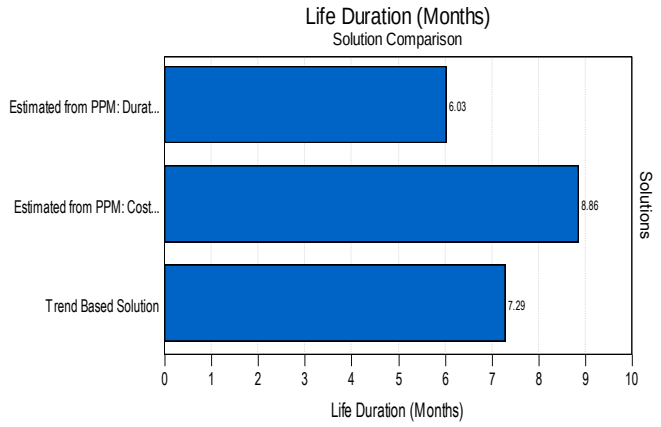
- W … ng t…
- F… hat a… er!
- L… B… o w…

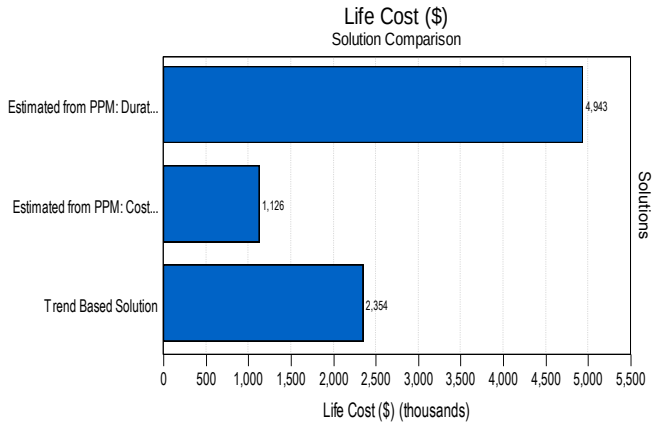
In practical terms, this means that to shorten a software project even a small amount by adding more people, the total effort and therefore the total cost will go up by a LOT!

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Labor cost is just one factor

- Cost of Delay – sooner is better
  - Loss of value that would be received
  - First to market advantages
  - Loss of market share to competitors
- Weigh tradeoffs of schedule and cost





QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

What determines what effort is needed?

- Project size
- Desired duration

What determines what effort is needed?

- Project size
- Desired duration
- Productivity
  - What is this?
  - How is it used?
  - How is it measured?

What is team productivity?

- We know it when we see it!
  - We can observe that some people or teams are more productive than others, but we don't have any "natural" way of measuring this
- Many, MANY factors affect productivity of a team working on a release:
  - Experience of the people
  - Work environment (physical environment, teamwork, management, etc., etc., etc.)
  - Tools and methods
  - Characteristics of the project
  - Etc., etc., etc.

What is team productivity?

- V
- M
  r

Notice that very few of factors that affect productivity are in the control of an individual team member!

QSM joins with most agile methodologists in a strong caution against using any measures of productivity we describe for the purpose of evaluation of personnel.

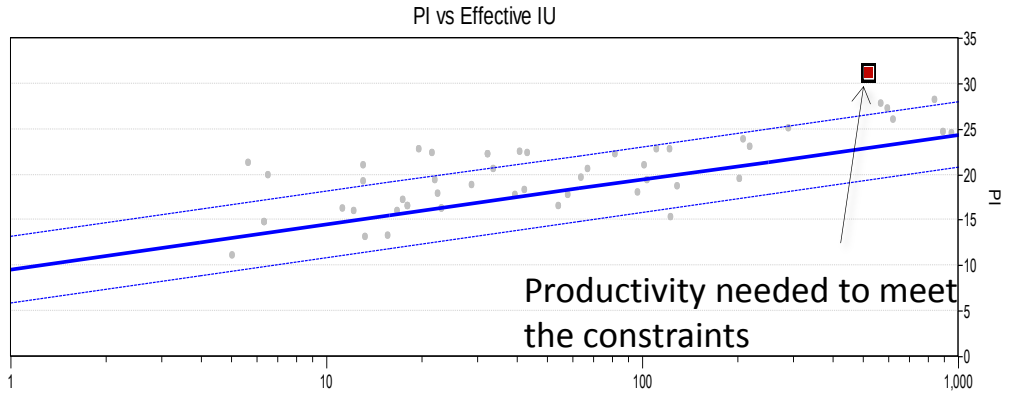
QSM®

Why measure productivity?

- Determine if constraints of time and effort for a release are feasible
- Assess the schedule/effort tradeoff
  - Unless you tell a team, “get done whatever you can, we’ll be fine”, you need to evaluate possible plans to get the best balance among viability of the release, cost of development, and time of release.
  - It’s not an “iron triangle”, but it’s not totally elastic either
  - Pick an expected level of productivity to assess tradeoff between duration and effort

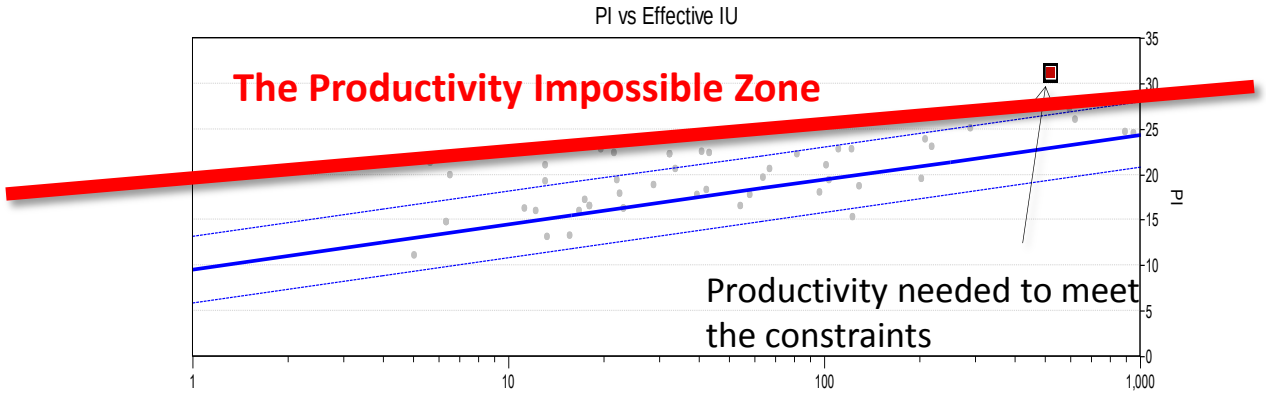
Productivity can help assess feasibility

- We've all seen management put arbitrary constraints on a project:
  - "That scope is great and will bring our customers the value they want. Now get it done in eight months for a million dollars, OK?"



Productivity can help assess feasibility

- We've all seen management put arbitrary constraints on a project:
  - "That scope is great and will bring our customers the value they want. Now get it done in eight months for a million dollars, OK?"



QSM®

Productivity can help assess feasibility

- W
  a

The Impossible Zone: Nobody ever has achieved that level of productivity on a project of that type and magnitude!

"Try hard and do your best" won't help!

You have history on your side to renegotiate the plan.

on

ey

K?"

QSM®

Pr

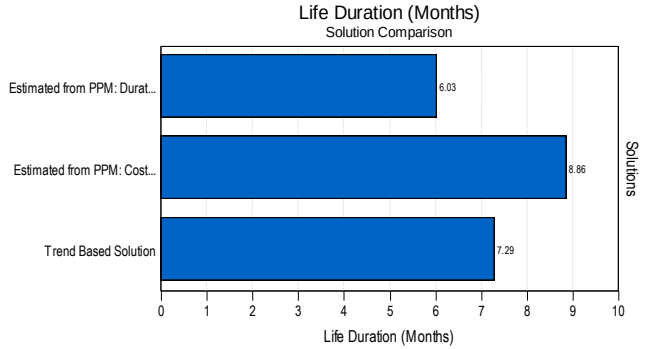
- W on a

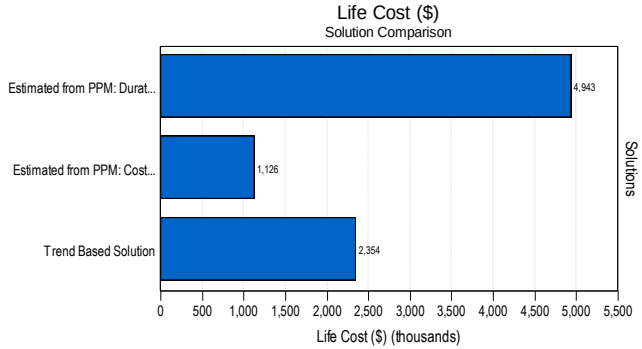
ey K?”

Catching plans that are in the impossible zone is the best medicine for ailing software development organizations.

Determine Schedule/Effort Tradeoff

- Need two metrics:
  - Expected size (story points, function points, stories or some other way of measuring size)
  - Expected productivity
- Determines possible tradeoffs between schedule and cost





Measuring productivity for estimation

- The only useful way to quantify productivity for use in estimates is to use data from previous projects.
- You can collect your own organizational data.
- You can benchmark against industry wide data.
- You “reverse engineer” the actual data to get a value for productivity that would have predicted the result. That forms the basis for future estimates.
- We will discuss this more in the section of this webinar on data collection.

Two measures of productivity

- Team Velocity
  - Story points per unit of time: Measure the number of story points your team develops in each iteration.
- QSM Productivity Index
  - A number that takes into account characteristics of the team, the environment, and the project that can be used to determine feasibility of constraints and the tradeoff of size and effort

QSM®

How is velocity supposed to help estimate?

- From history of similar projects previously completed, determine the historical velocity in “story points per iteration”
  - Almost all agile tracking tools have this data, so it’s easy to collect.
  - What “similar projects” means is problematic
- Estimate the size of the upcoming release in story points
- Divide size (in story points) by velocity (in story points per iteration) to compute expected duration (number of time boxed iterations)

That sounds easy! BUT…..

"For every complex problem there is an answer that is clear, simple, and wrong."

H. L. Mencken

- This method of using velocity to estimate ignores several critical aspects of productivity
  - You need to find the average from comparable projects
  - The velocity of a team varies throughout a project…software is not produced at a constant rate
  - Velocity varies based on team size in a non-linear way…you cannot just add velocities
  - Velocity depends on the size of the project

QSM®

Averages from comparable projects

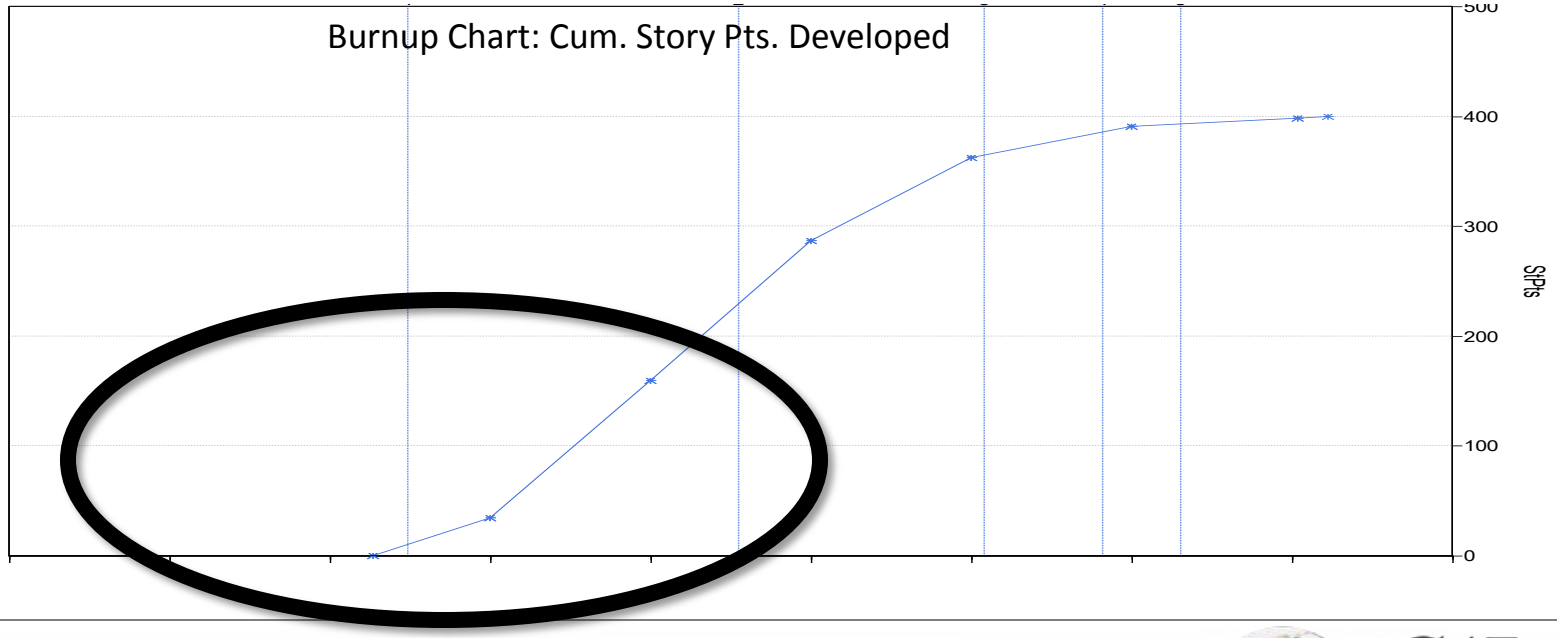
- Nobody expects velocity to be exactly the same for each iteration on each project.
- To use it for prediction, you're going to use an average value (well, the statistician in you may look at both mean and median), but average of what?
- Easy part: Group projects in "like types"
  - You don't expect the same productivity for a website that just displays information that you do for the software in a medical device
  -

Average MPH Per Tankful???

- Some driving was in the city, between 20-30 mph.
- Some was on the highway, driving between 55-65 mph.
- I almost never drive near 40 mph.

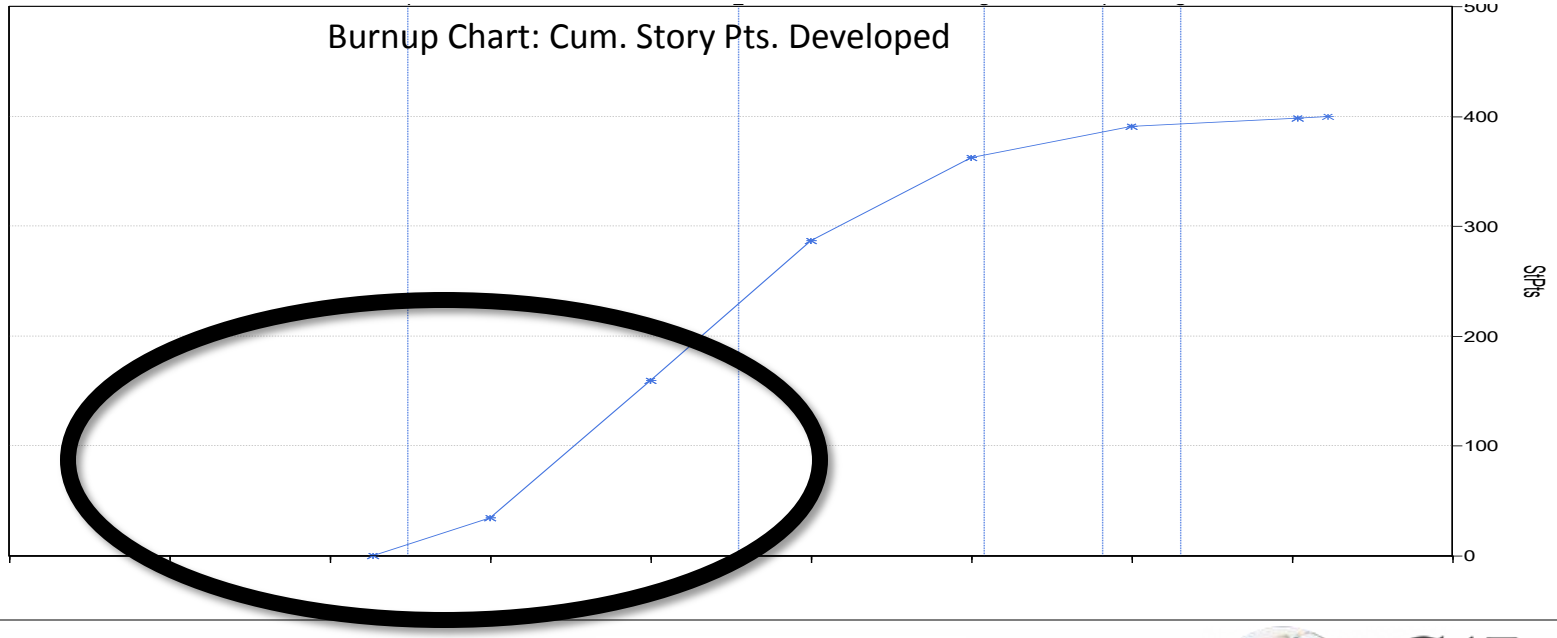
Velocity is not constant during a project

- Most people know velocity starts off lower at the beginning of a project



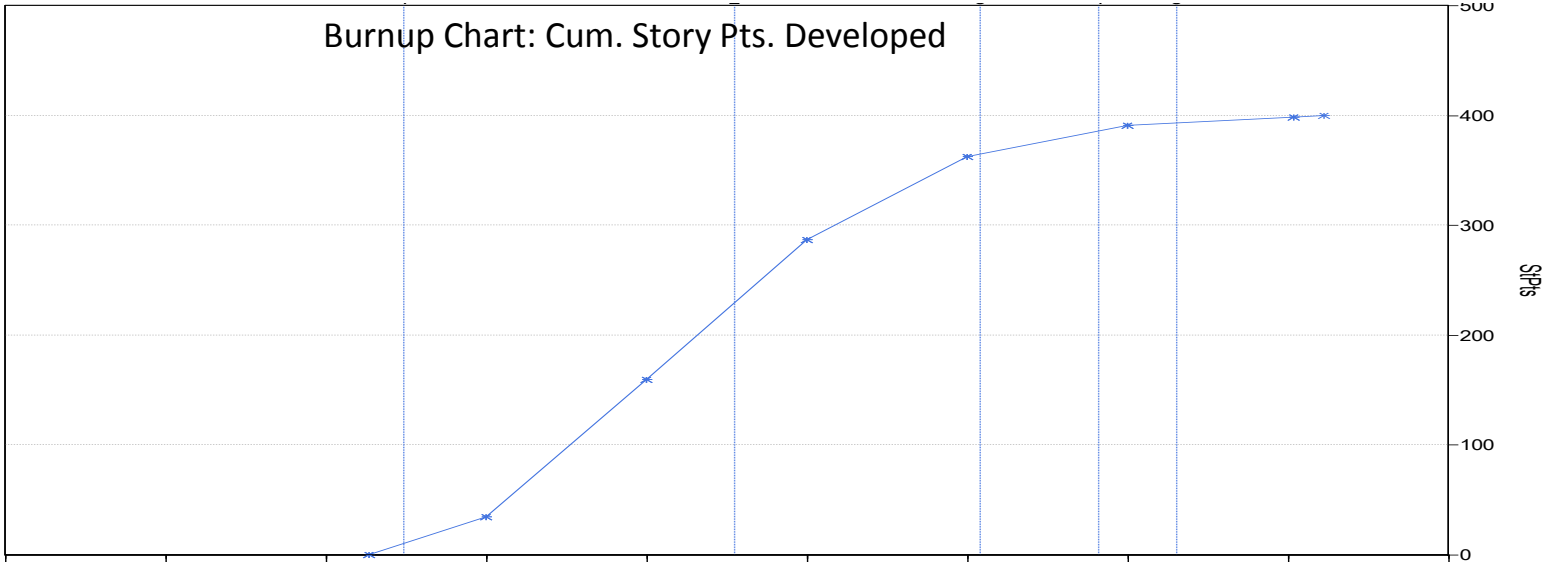
Velocity is not constant during a project

- Most people know velocity starts off lower at the beginning of a project
  - Often chalked up to “the team has to get to know each other”
  - It’s more systematic than that (Putnam-Norden-Rayleigh curve)



Velocity is not constant during a project

- So what historical data should you average?
  - Average velocity of each sprint across multiple projects?
  - Compute average velocity for each project, then look at the trend of those averages?



Velocity varies based on team size

- Of course that's true…
  - You expect the number of story points a 12 person team develops in two weeks to be more than the number of story points a 6 person team develops in the same two weeks.
- But how much more?

We learned it all in high school

- "If John can paint a room in 8 hours by himself, and Patty could paint the room in 6 hours by herself, how long will it take John and Patty to paint the room together."
- The high school algebra solution assumes "effort is constant"
  - Working together it will take about three and half hours (3 3/7 to be precise).

We learned it all in high school

- "If John can paint a room in 8 hours by himself, and Patty could paint the room in 6 hours by herself, how long will it take John and Patty to paint the room together."
- The high school algebra solution assumes "effort is constant"
  - Working together it will take about three and half hours (3 3/7 to be precise).
- Everyone knows the real answer is 10 hours!
  - "Where should we go for dinner?"
  - "I want the ladder first" "No, I get it first"

Velocity varies based on team size

- Of course that's true…
  - You expect the number of story points a 12 person team develops in two weeks to be more than the number of story points a 6 person team develops in the same two weeks.
- But it's a non-linear relationship (Mythical Person Month)
  - If Team A has a velocity of 50 story points per sprint and Team B has a velocity of 60 points per sprint, the combined velocity is
    - Probably bigger than either the 50 or 60
    - But NOT 110 = 50 + 60! It will be less.
- Effort is NOT constant as team size changes!

QSM®

Velocity depends on software size

- We already found several complexities to know what “average velocity” to use for predicting duration:
  - You have to adjust for type of project (but that’s relatively easy to do) and take the proper average (hmmm….)
  - You have to adjust for the non-linear change in velocity as projects progress
  - You have to adjust somehow for the non-linear effect of team size
- There’s yet another non-linearity: productivity depends on software size (how many story points).
  - Even with a fixed team, doubling the size does not double the duration.

QSM®

What's the alternative?

- The Putnam Software Equation relates:
  - Size
  - Duration
  - Effort
  - Productivity

$$Size = k * duration^{4/3} * effort^{1/3}$$

k is derived from Size and Productivity Index

QSM®

What's the alternative?

- SLIM-Estimate uses a value called Productivity Index as an input to the Software Equation.
  - Can be used to tradeoff size, duration, and effort
  - PI trend values are derived by "reverse engineering" actual values of size, duration, and effort from previous projects using the Software Equation to "Solve for PI".
  - "Solve for PI" and comparing to historical data also helps assess feasibility of desired constraints for a new project
    - See if you're in the "impossible zone"

QSM®

How do agile methods fit in?

- The values used for Productivity Index come from a trend calibrated from historical data of similar projects
  - You can collect your own organizational data of projects actually completed
  - You can benchmark against industry wide data
- Computing trends from collections of previous agile projects captures
  - Similar sizing methods (e.g. Story Points)
  - Productivity gained through agile techniques

Velocity isn't all bad!

- All estimates have a degree of uncertainty
  - Estimates should include the risk from uncertainty
- The simple calculation using velocity can get you in the ballpark….

Velocity isn't all bad!

- …if you don't mind a really big ballpark!

Project Tracking and Control

"The shortest distance between two points is under construction"

Noelie Altito

Webinar Sponsored by Computer Aid, Inc.

QSM®

CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

QSM®

Some reasons projects are late

- Lower productivity than expected
- Scope creep
  - Changing scope does not necessarily mean increasing scope
  - If your minimal viable release keeps growing, don't expect to meet your original estimate
  - Count "throwins" are part of the size
- Requirements churn
- Quality issues

QSM®

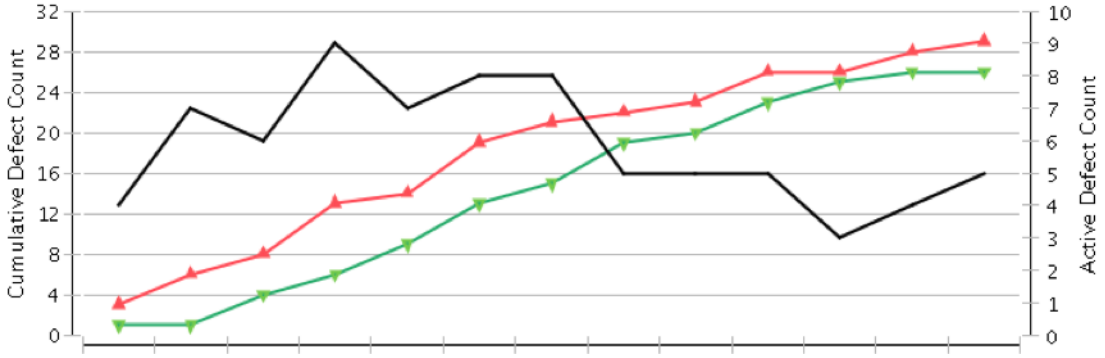
Some reasons projects are late

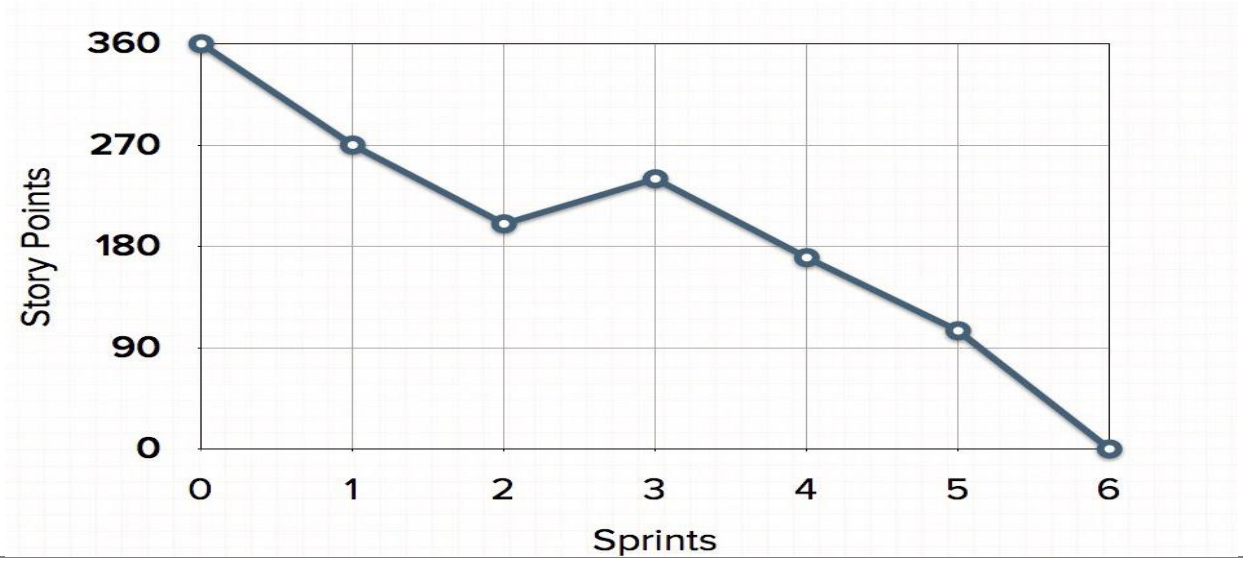
- L
- S
- C
- F

You can't necessarily see these problems day to day.

We can groom the backlog, develop to the "definition of done" sprint after sprint, and still fall victim to these problems.

Agile teams collect metrics



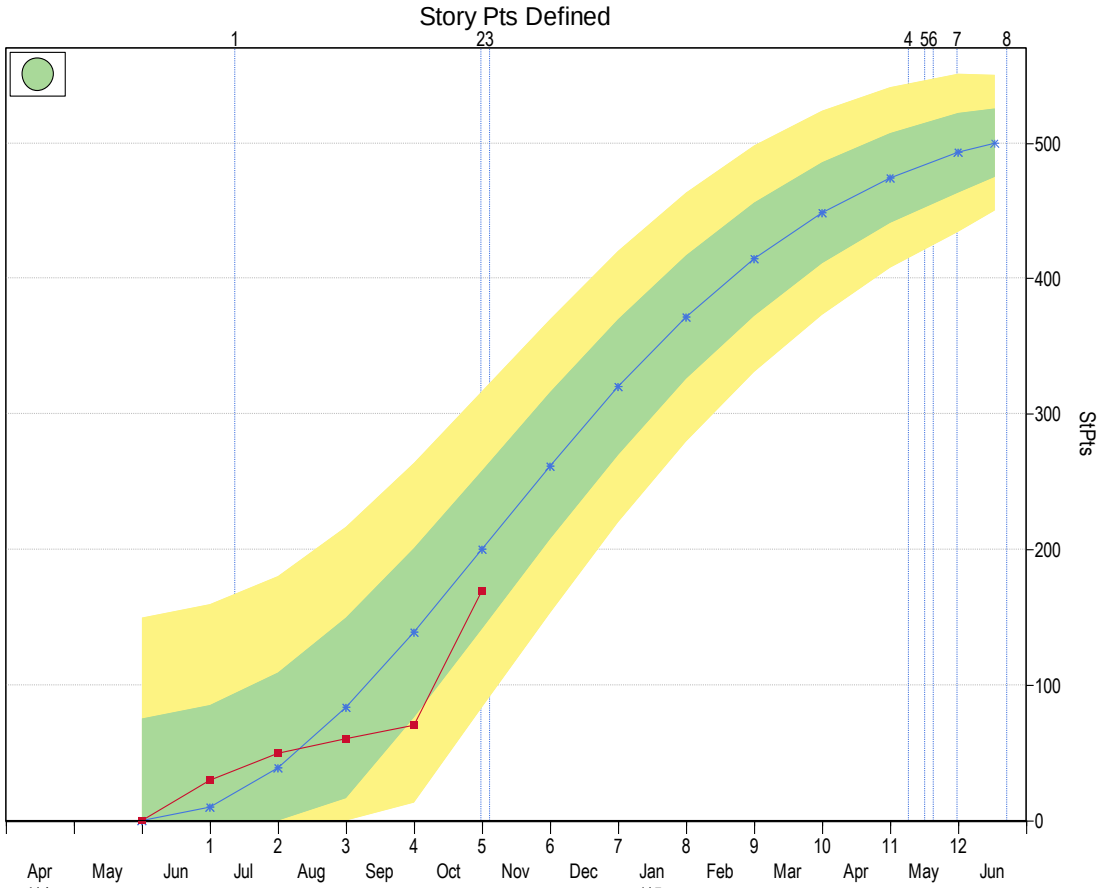


QSM®

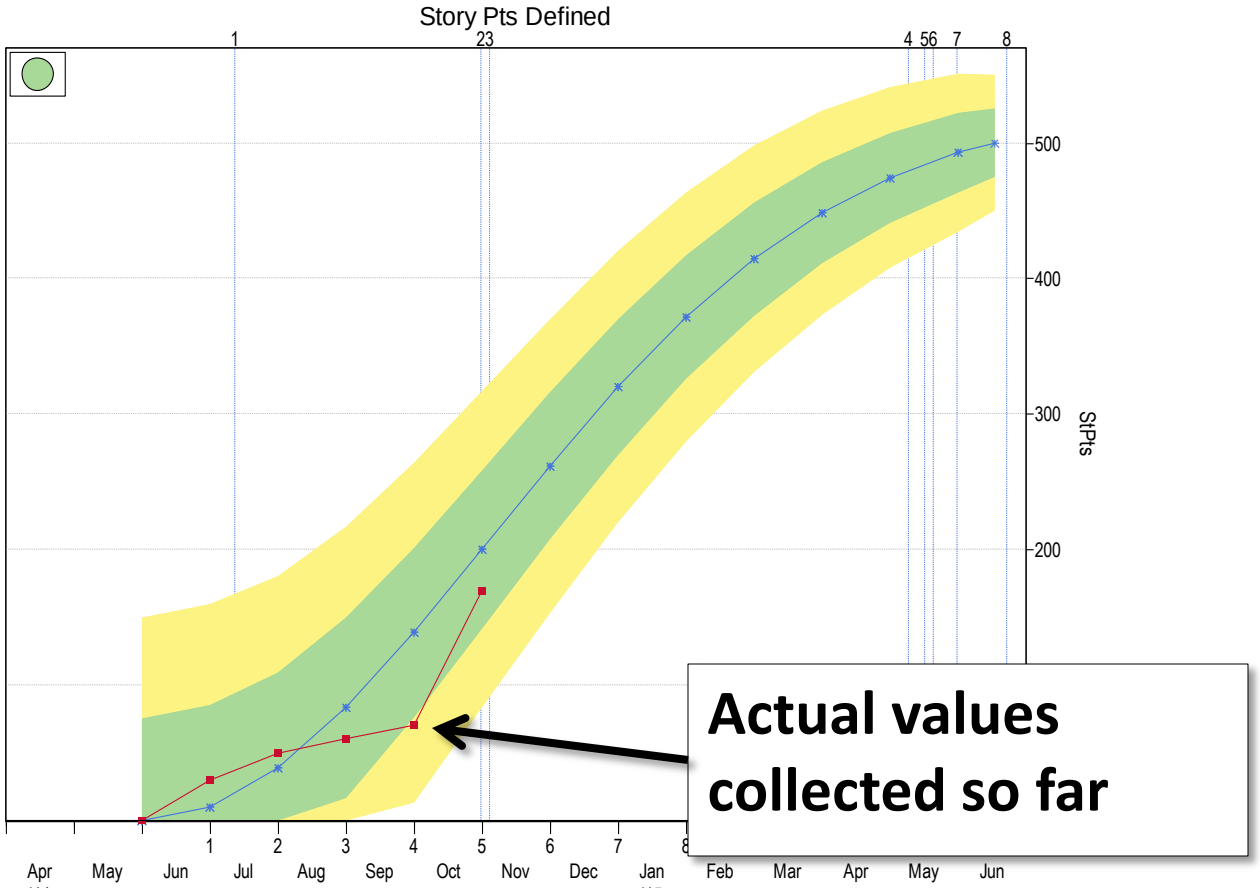
Purpose of metrics

- Quick course corrections
- Spot potential trouble before it happens
- Replan if needed
- Collect historical data for use in future estimates and process improvement activities

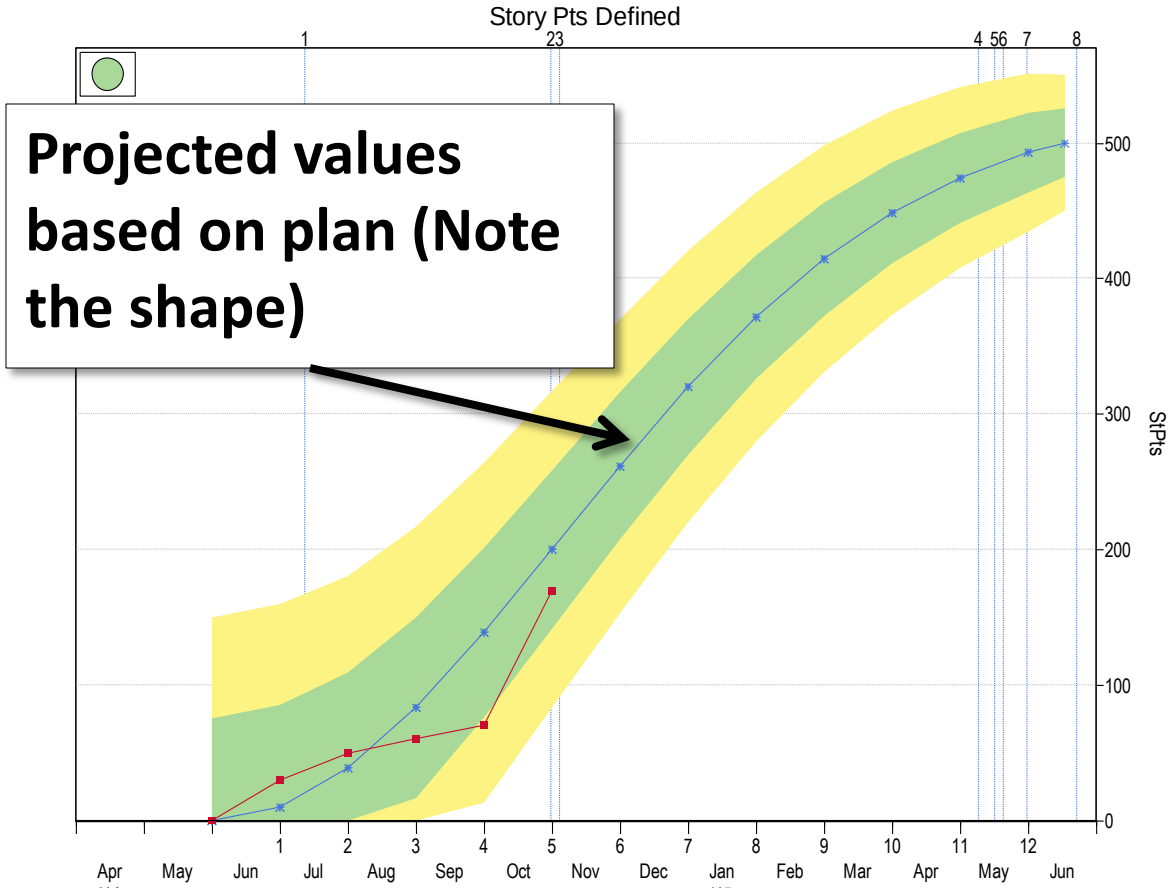
Anatomy of a metric



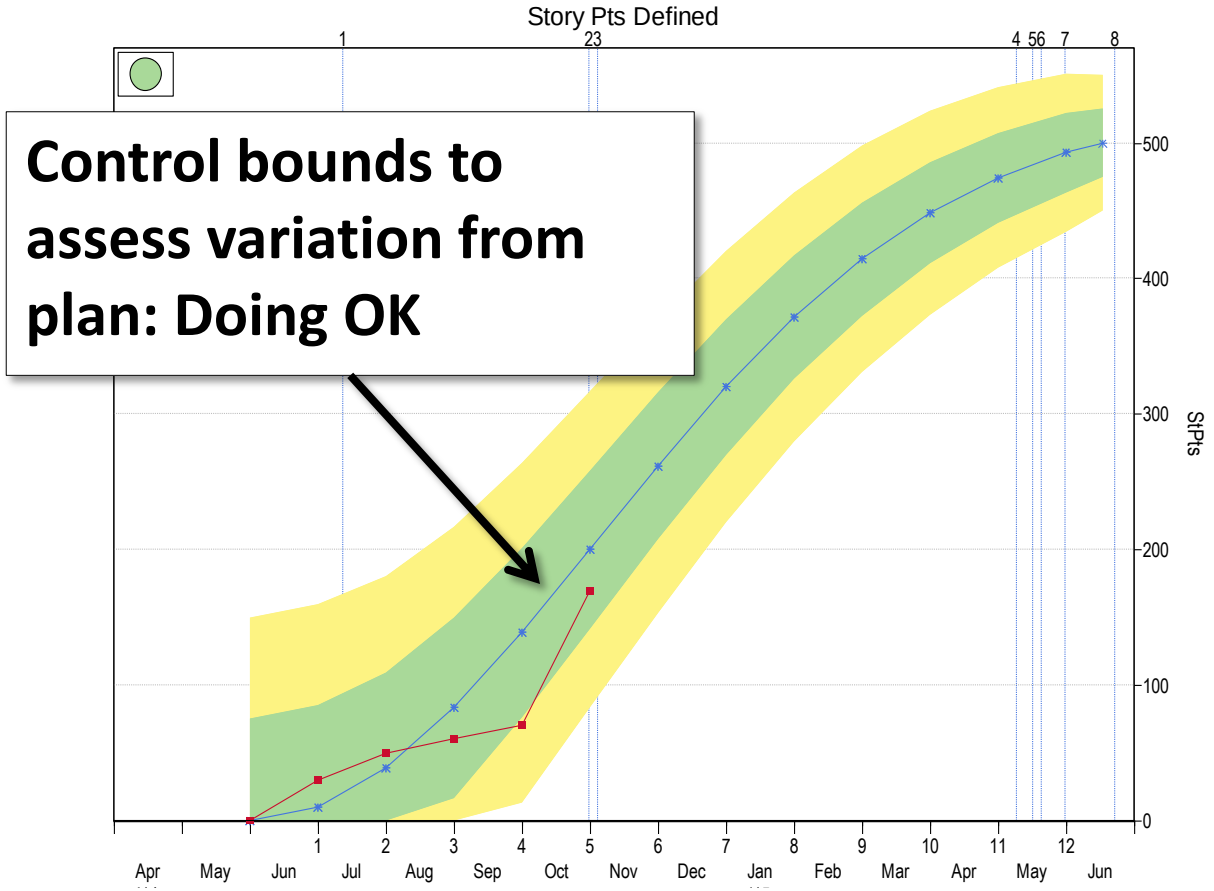
Anatomy of a metric



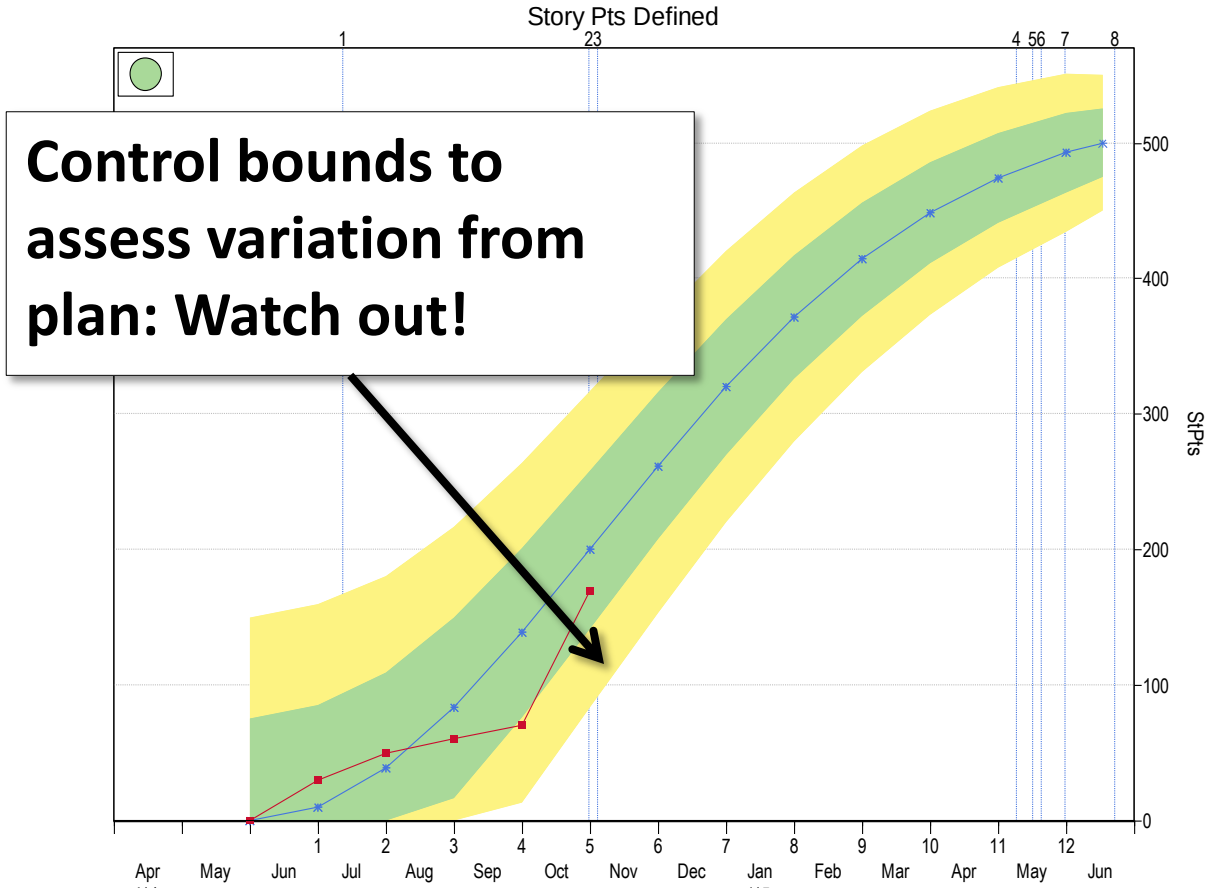
Anatomy of a metric



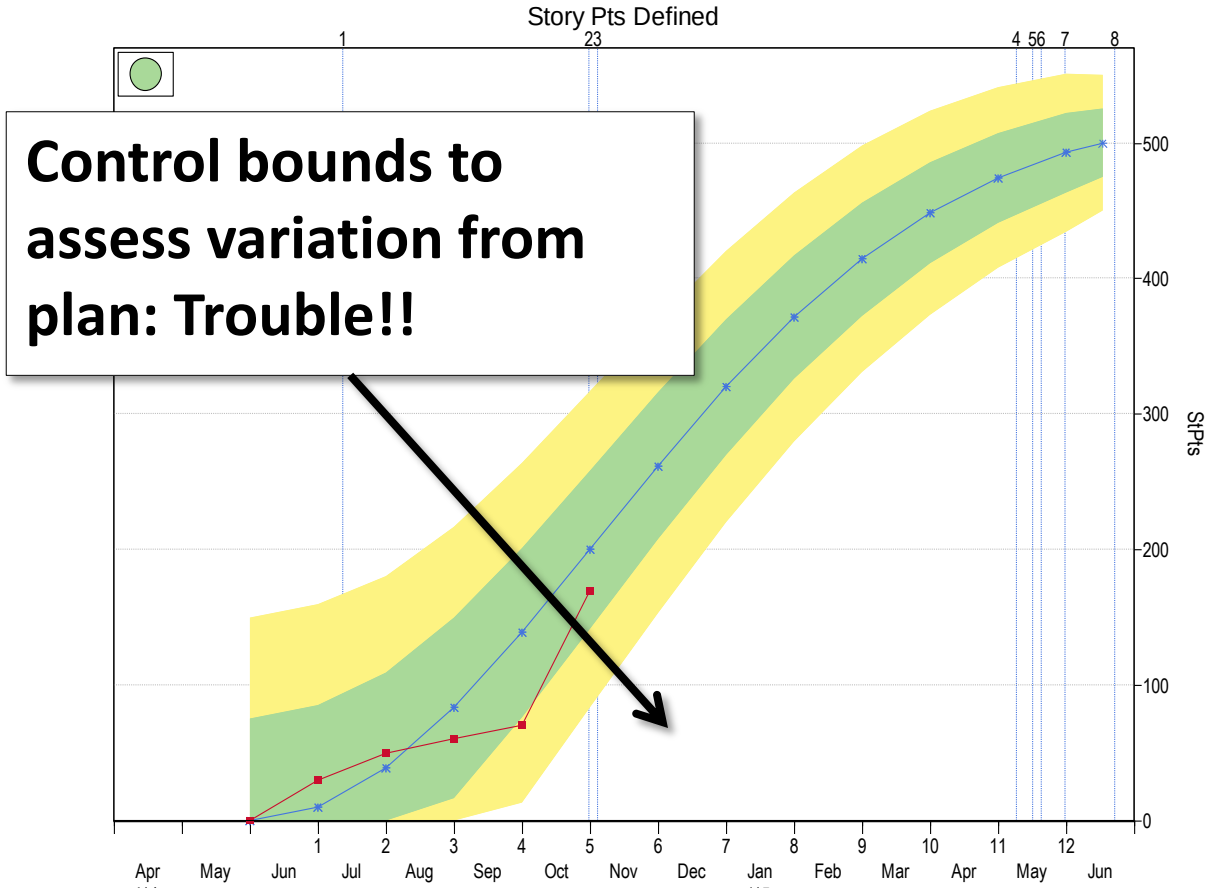
Anatomy of a metric



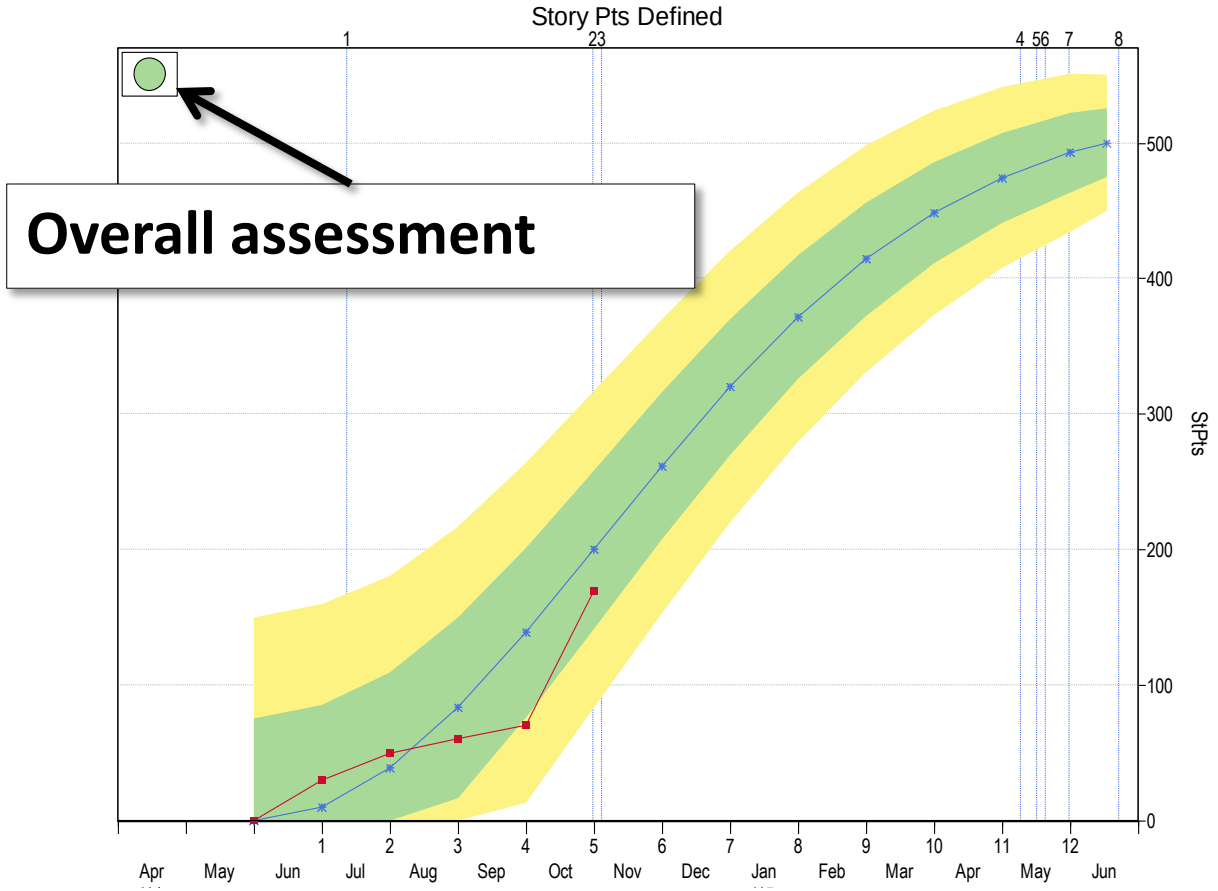
Anatomy of a metric



Anatomy of a metric



Anatomy of a metric

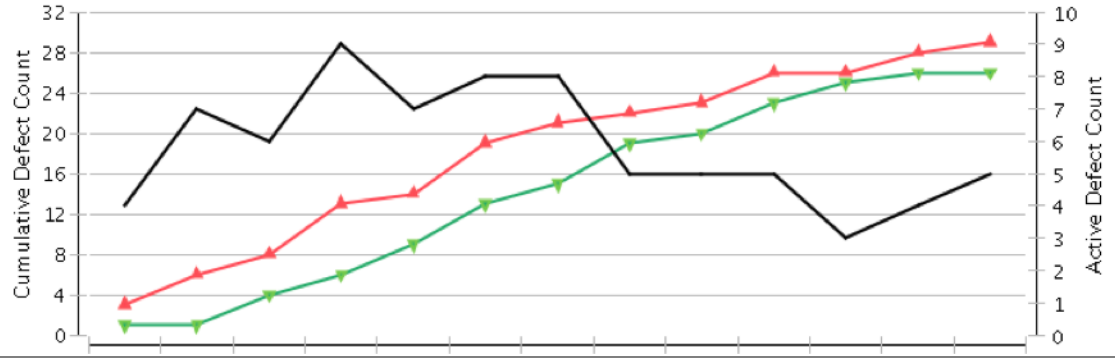


Combine metrics from various tools

- (This is not a definitive list)
- Project management metrics
  - Milestones reached
  - Effort expended
  - Stories defined, developed, and backlog remaining
- Technical metrics
  - Defects found
  - Defects corrected
  - Tests passed
- Assess each and reach overall assessment

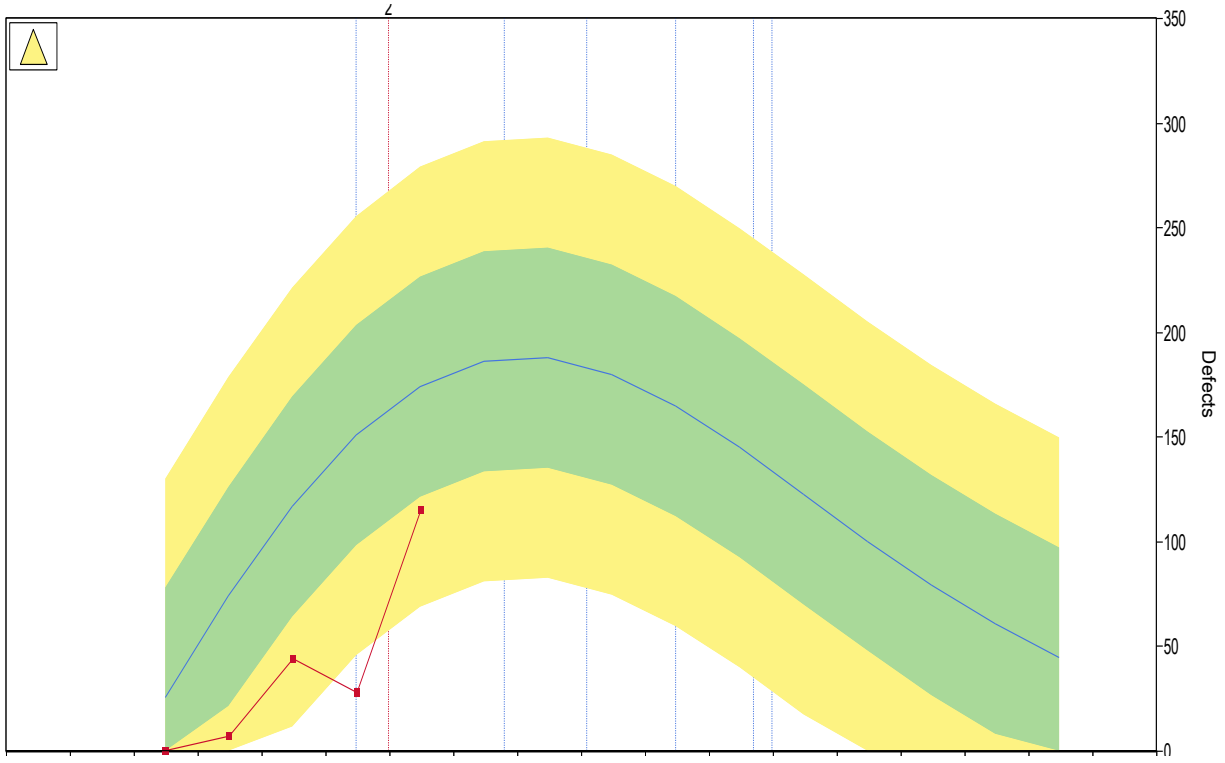
Defects and control bounds

- Most agile teams monitor defects found and defects fixed.
- If you're finding a lot but not fixing them, that's an obvious problem.



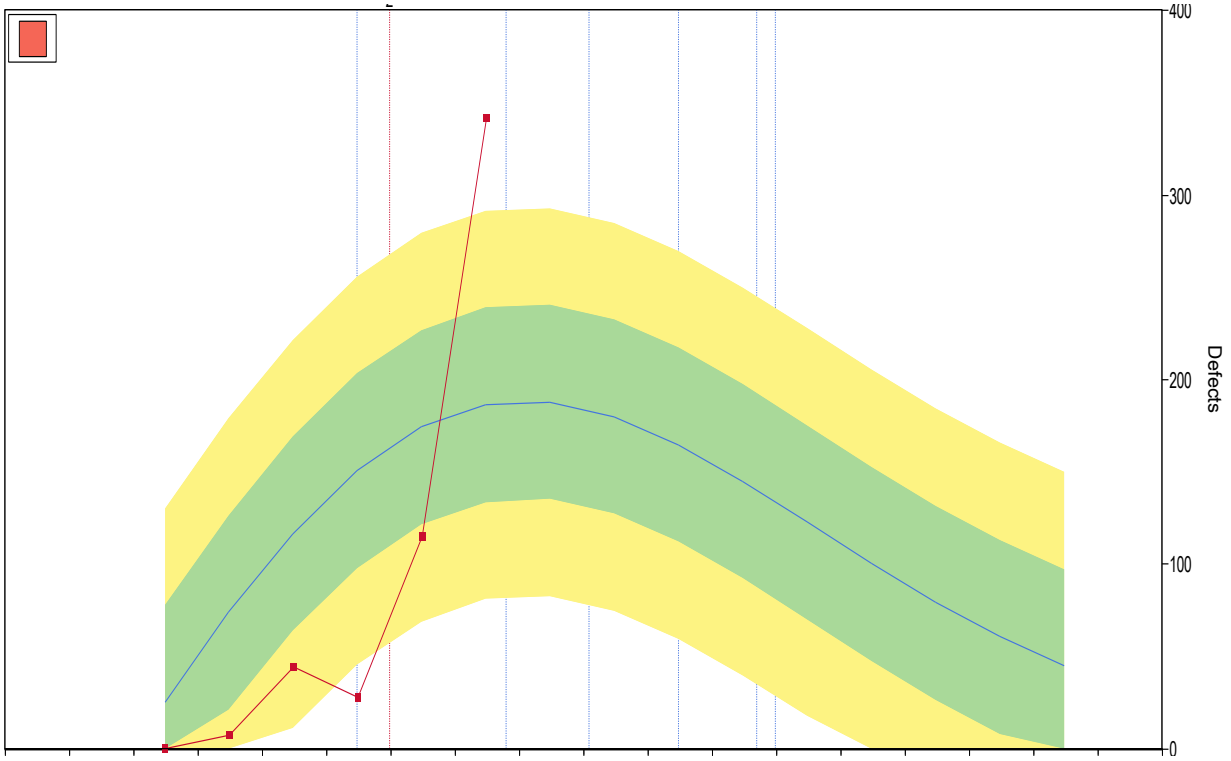
Defects and control bounds

- Too few is as bad as too many!



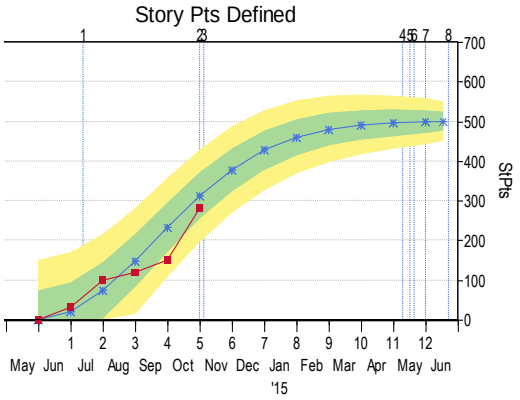
Defects and control bounds

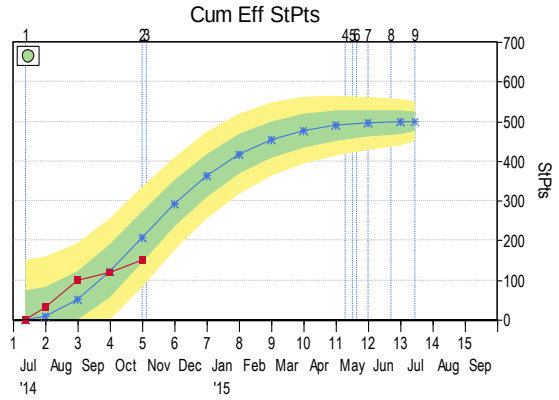
- Too few is as bad as too many!

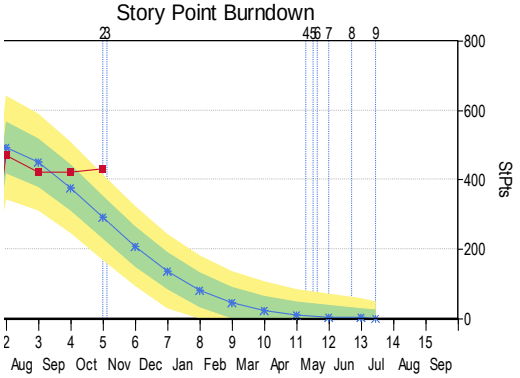


Monitoring story definition

Story Point Metrics







Phase	Start Date	End Date	From Project Start*)	From Project Start*)	Duration (Mos.)
Story Writing	6/1/2014	10/31/2014	0.00	5.00	5.00
Development/Test	7/13/2014	10/31/2014	1.42	5.00	3.58

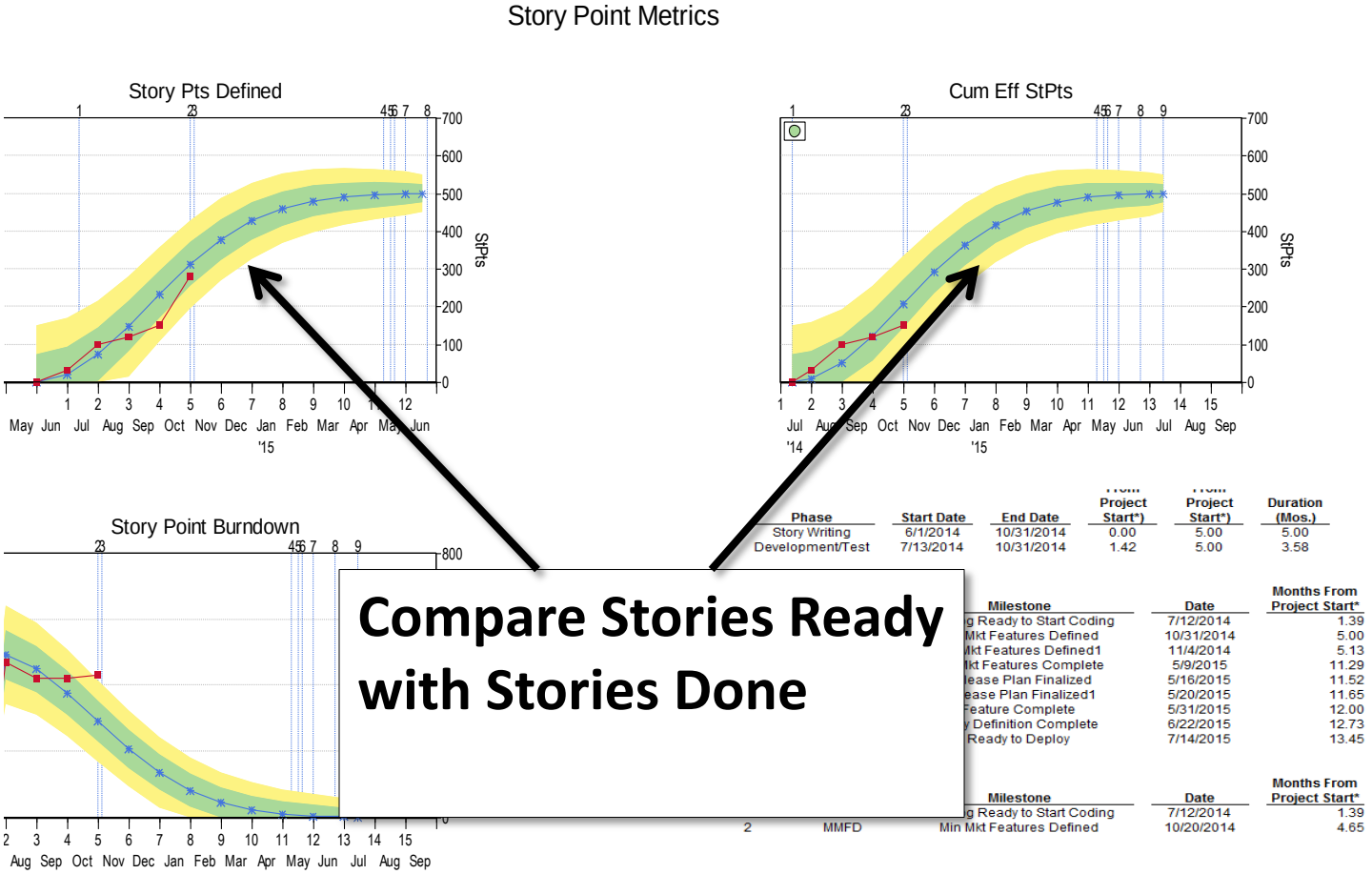
Milestones: Current Plan

Milestone ID	Milestone Acronym	Milestone	Date	Months From Project Start*
1	IT1	Backlog Ready to Start Coding	7/12/2014	1.39
2	MMFD	Min Mkt Features Defined	10/31/2014	5.00
3	MMFD1	Min Mkt Features Defined1	11/4/2014	5.13
4	MMFC	Min Mkt Features Complete	5/9/2015	11.29
5	RELPL	Release Plan Finalized	5/16/2015	11.52
6	RELP1	Release Plan Finalized1	5/20/2015	11.65
7	FC	Feature Complete	5/31/2015	12.00
8	SDCPL	Story Definition Complete	6/22/2015	12.73
9	READY	Ready to Deploy	7/14/2015	13.45

Milestones: Actual

Milestone ID	Milestone Acronym	Milestone	Date	Months From Project Start*
1	IT1	Backlog Ready to Start Coding	7/12/2014	1.39
2	MMFD	Min Mkt Features Defined	10/20/2014	4.65

Monitoring story definition

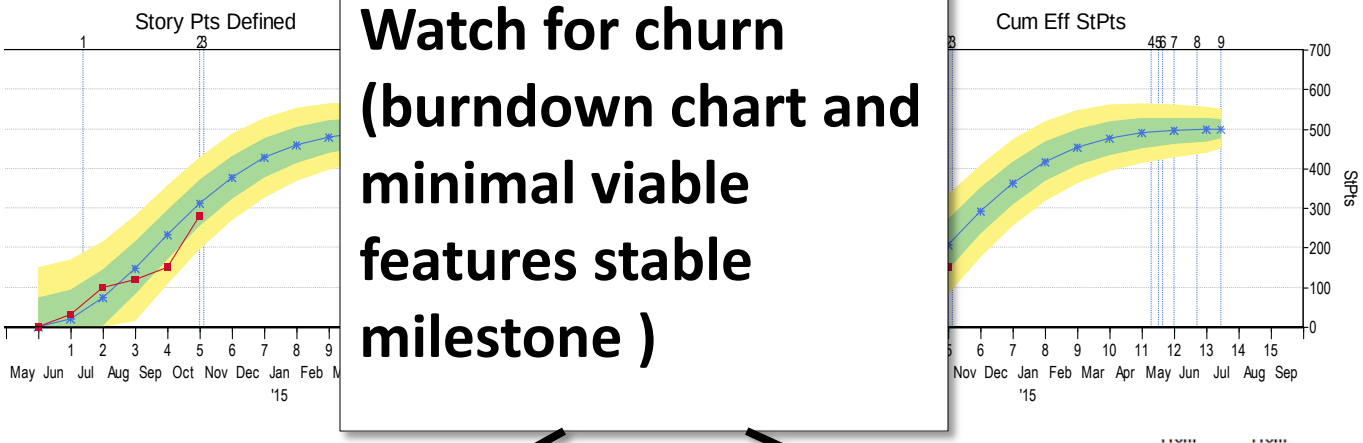


QSM®

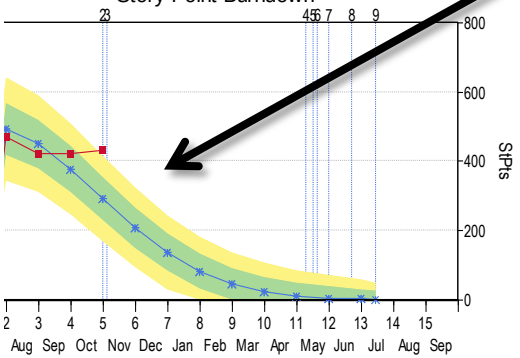
CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Monitoring story definition

Story Point Metrics



Watch for churn (burndown chart and minimal viable features stable milestone)



Phase	Start Date	End Date	From Project Start*)	From Project Start*)	Duration (Mos.)
Story Writing	6/1/2014	10/31/2014	0.00	5.00	5.00
Development Part	7/13/2014	10/31/2014	1.42	5.00	3.58

Milestones: Current Plan

Milestone ID	Milestone Acronym	Milestone	Date	Months From Project Start*
1	IT1	Backlog Ready to Start Coding	7/12/2014	1.39
2	MMFD	Min Mkt Features Defined	10/31/2014	5.00
3	MMFD1	Min Mkt Features Defined1	11/4/2014	5.13
4	MMFC	Min Mkt Features Complete	5/9/2015	11.29
5	RELPL	Release Plan Finalized	5/16/2015	11.52
6	RELP1	Release Plan Finalized1	5/20/2015	11.65
7	FC	Feature Complete	5/31/2015	12.00
8	SDCPL	Story Definition Complete	6/22/2015	12.73
9	READY	Ready to Deploy	7/14/2015	13.45

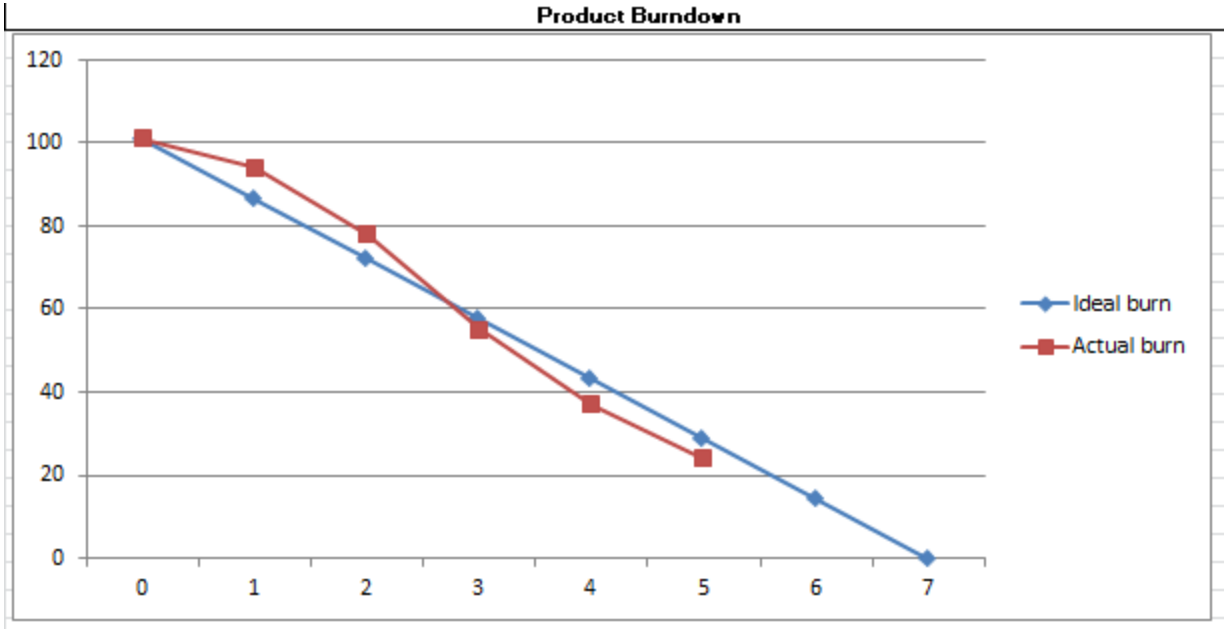
Milestones: Actual

Milestone ID	Milestone Acronym	Milestone	Date	Months From Project Start*
1	IT1	Backlog Ready to Start Coding	7/12/2014	1.39
2	MMFD	Min Mkt Features Defined	10/20/2014	4.65

QSM®

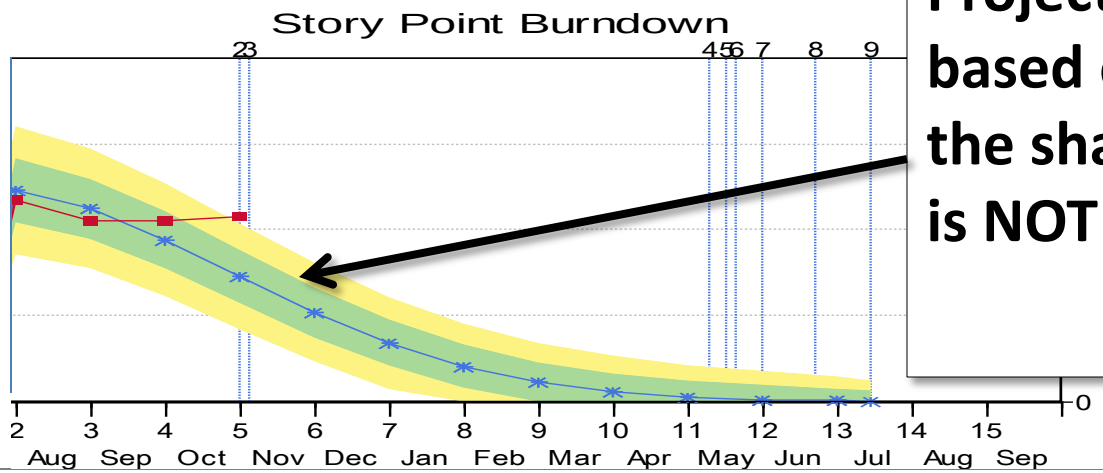
CAI Computer Aid, Inc.®
World Leader in IT Metrics and Productivity

Backlog Burndown Chart



Backlog Burndown Chart

- "Traditional" burndown chart but with plan and control bounds.
- The "real measure" of where we are compared to where we want to be!



Projected values based on plan (Note the shape—velocity is NOT constant)

QSM®

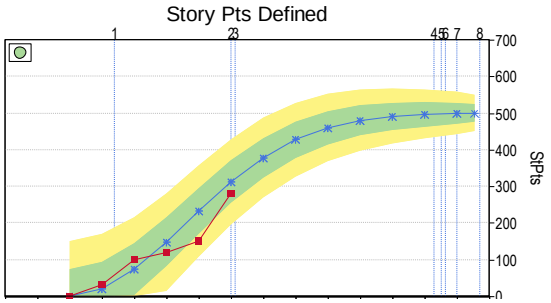
CAI Computer Aid, Inc.® World Leader in IT Metrics and Productivity

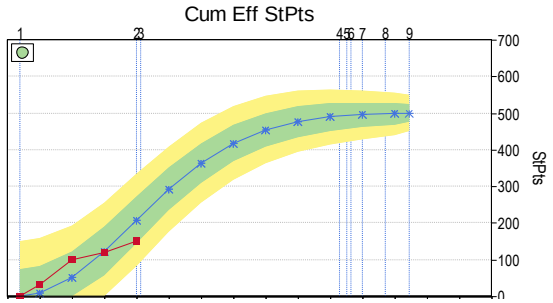
Burndown vs. Story Points Developed

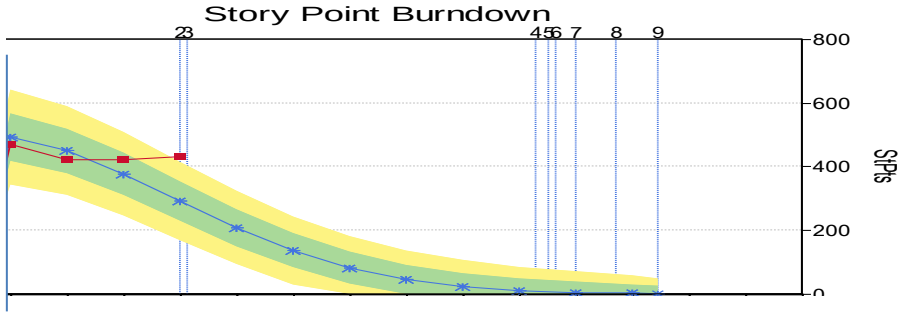
- Why isn't the Burndown Chart just the Story Points Developed "upside down"?
  - Equivalently, is this the right equation?
    - Backlog remaining = Plan – Story Points Developed?
- It isn't, because the Burndown Chart includes changes to the backlog
  - The equation is:
    - Backlog remaining = Plan – Story Points Developed + Story Points Added – Story Points Removed
- Stories removed may include stories already defined and developed!

Indication of Requirements Churn

- Too little “progress towards 0” on the burndown while Story Points Defined and Story Points Developed are going up nicely







QSM®

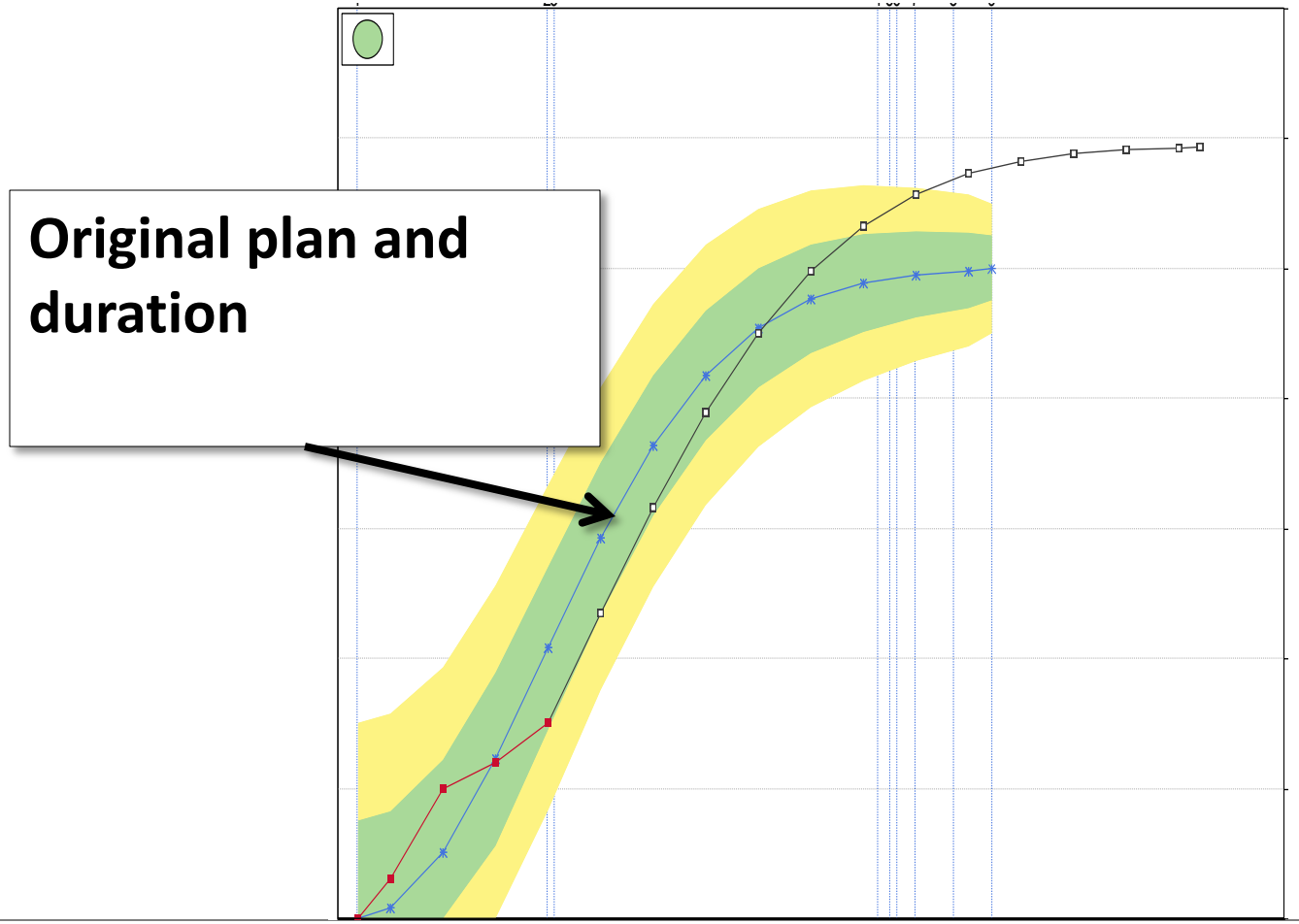
When metrics are “out of control”

- Control bounds show natural variation. Don’t expect “exact tracking to plan.”
  - Expect metrics to go above and below plan
  - If they stay in the “green zone” or occasionally dip into the “warning” zone, you’re ok
- If you understand why they are in the warning zone, you may be able to take some action.
  - Expect to see them back green soon

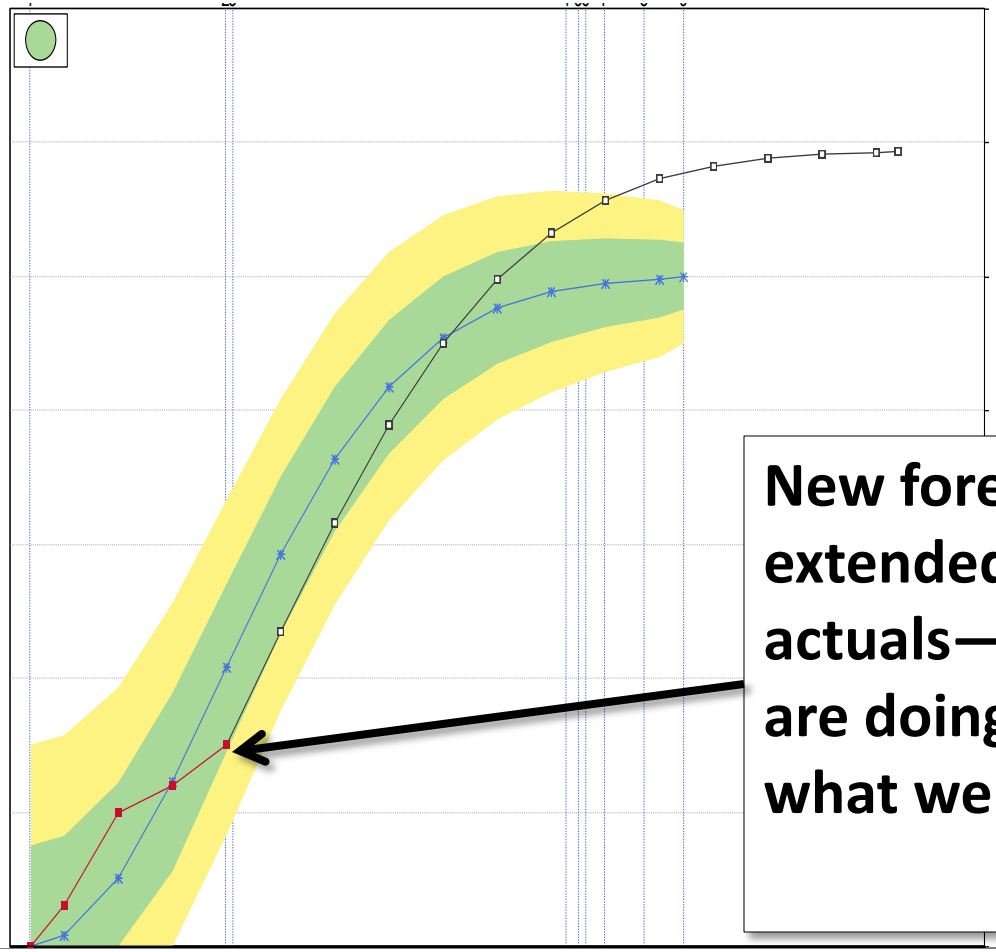
Reforecasting

- When actuals are consistently out of the “green zone” in the same direction and milestones are late, it’s time to reforecast.
- Use actuals to forecast to completion
  - Different metrics may give different answers
  - Weigh multiple metrics
  - Include plan changes

Anatomy of a forecast

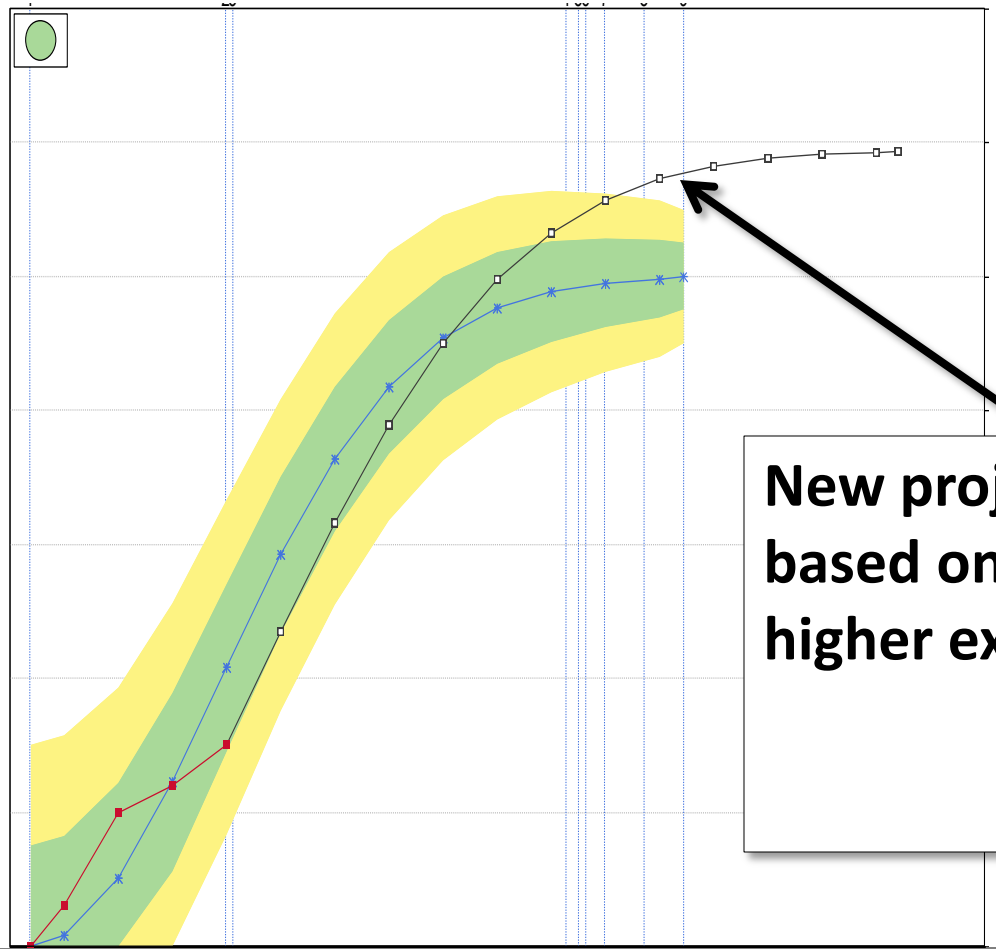


Anatomy of a forecast



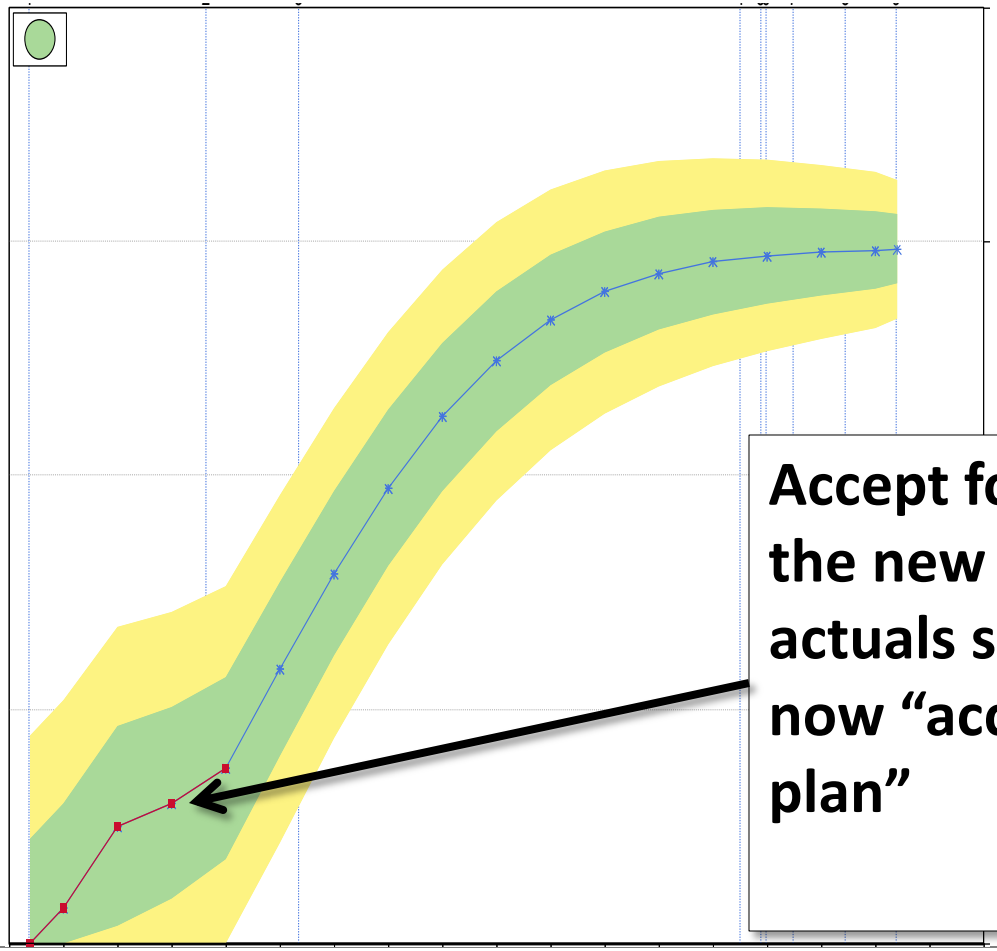
New forecast is extended from actuals—what we are doing "trumps" what we planned

Anatomy of a forecast



New projected plan based on actuals and higher expected total

Anatomy of a forecast



Accept forecast as the new plan. Note actuals so far are now "according to plan"

QSM®

Project Control Summary

- Don’t just collect metrics, use them to make decisions
- Combine metrics of various types from various sources
  - Story definition and story development
  - Project management and technical metrics
- Don’t just look at actuals
  - Compare to plan
  - See if actuals are within control bounds

Project Control Summary

- When metrics start to go out of control
  - Look for underlying reason
  - Look for mitigation to bring plan back under control
- When you see you cannot get back under control with your current plan, reforecast
  - Use the reality of actuals to reduce risk
  - Change the estimate assumptions based on what you've learned

Dr. Andy Berner
Sr. Software Engineer
Quantitative Software Management, Inc.
andy.berner@qsm.com

Hosted by:

Jessica Dahbour
ITMPI
Jessica_Dahbour@compaid.com

Webinar Sponsored by Computer Aid, Inc.