# Agile Estimation: Beyond the Myths Part 2

Andy Berner

Quantitative Software Management, Inc.

QSM® The Intelligence behind Successful Software Projects

# Agenda

- Review from Part 1
  - Some Key Agile Principles
  - What do we estimate?
- Configuration for estimation:
  - Milestones
  - Productivity and Effort
- Project Control
- Questions

# Agenda for Part 1, available as replay

- Some Key Agile Principles
- What Do We Estimate? The importance of planning a release in agile organizations
- Configuration for estimation:
  - Sizing
  - Shape of the work
- Questions

# Some Key Agile Principles

***"The fundamental things apply, as time goes by"***

*By Herman Hupfeld,*
*made famous in "Casablanca"*

QSM® The Intelligence behind Successful Software Projects

# 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
  - Definition of ready/groom the backlog
- Working software is the primary measure of progress

# 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® The Intelligence behind Successful Software Projects

# 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.

# “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!

# “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

# In summary

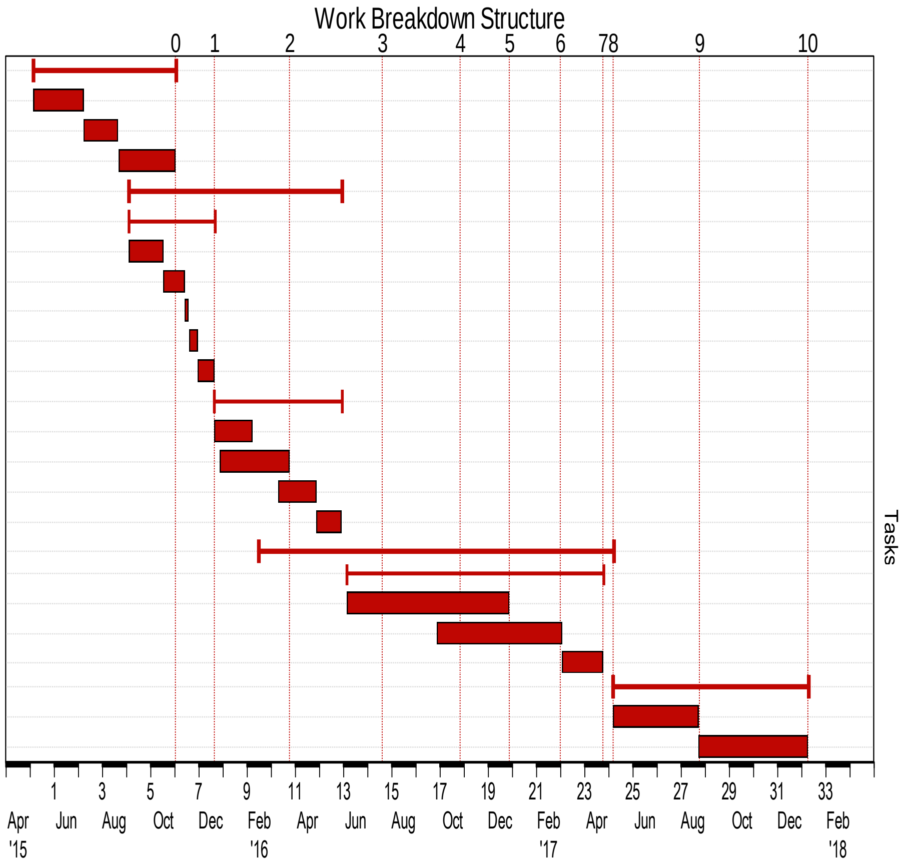
- Releases must provide consumable value.
- Unlike sprints within a project, projects to develop releases are not always time-boxed.
- Team members should make sprint level decisions, but estimating a release requires tools and methods that take into account:
  - Overall goals: Balance “minimal” with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

# Milestones

> *"Life isn't a matter of milestones, but of moments"*
>
> *Rose Kennedy*

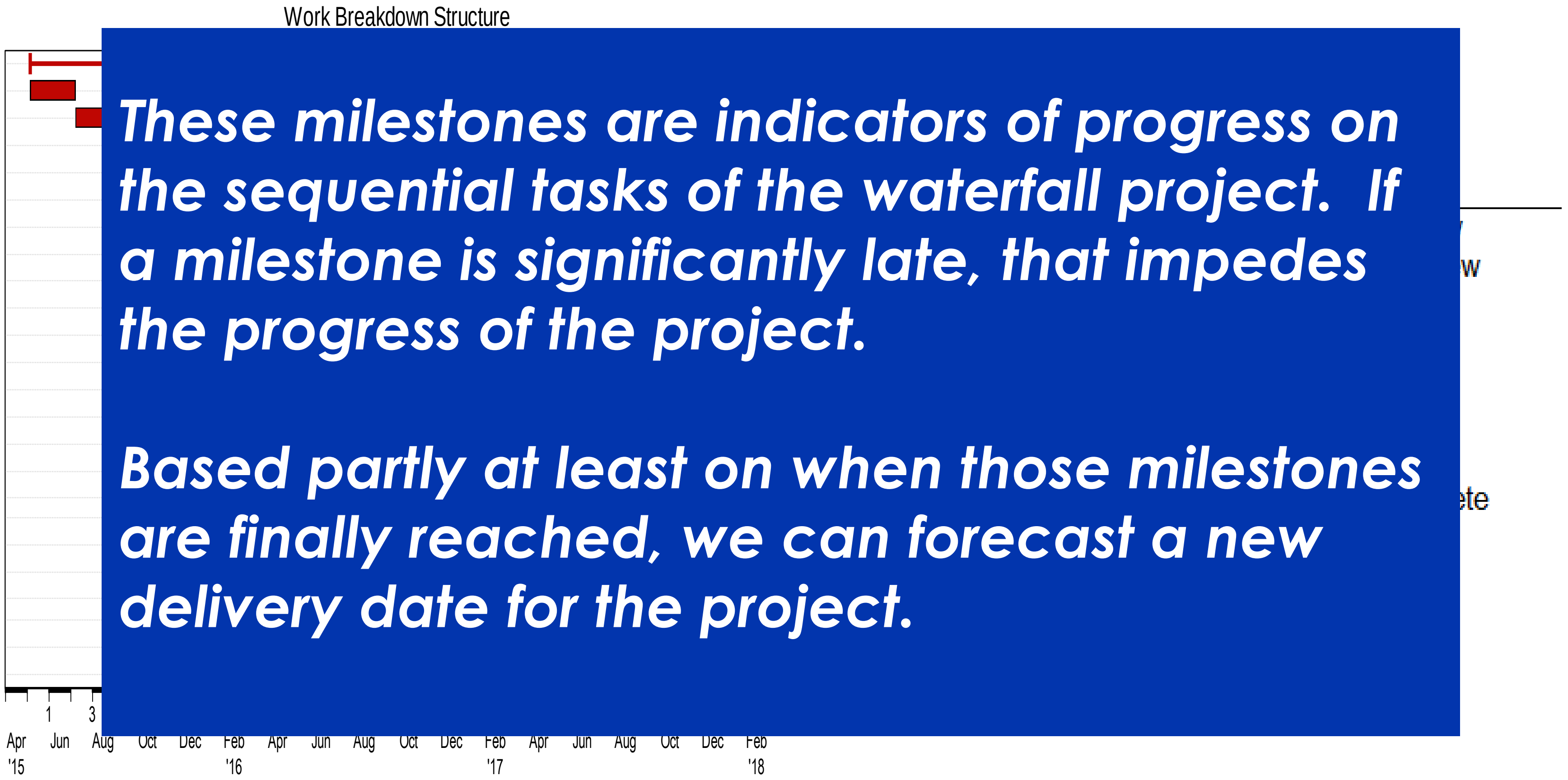
The Intelligence behind
Successful Software Projects

# 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 |

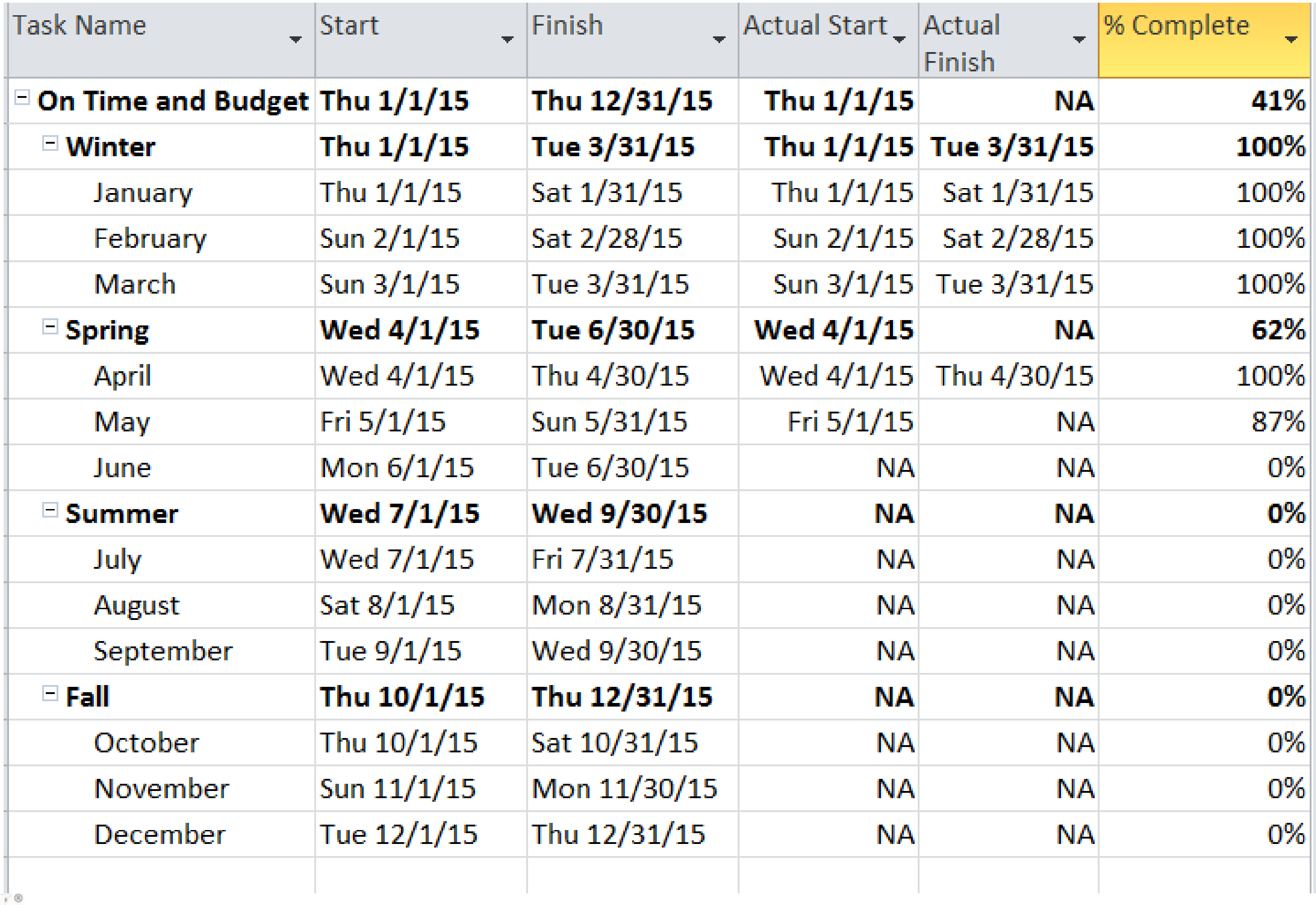
# Waterfall is managed through milestones



***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** | **41%** |
| ⊟ **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** | **NA** | **62%** |
| 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 | NA | 87% |
| June | Mon 6/1/15 | Tue 6/30/15 | NA | NA | 0% |
| ⊟ **Summer** | **Wed 7/1/15** | **Wed 9/30/15** | **NA** | **NA** | **0%** |
| July | Wed 7/1/15 | Fri 7/31/15 | NA | NA | 0% |
| August | Sat 8/1/15 | Mon 8/31/15 | NA | NA | 0% |
| September | Tue 9/1/15 | Wed 9/30/15 | NA | 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

# In agile projects, working software is the measure of progress

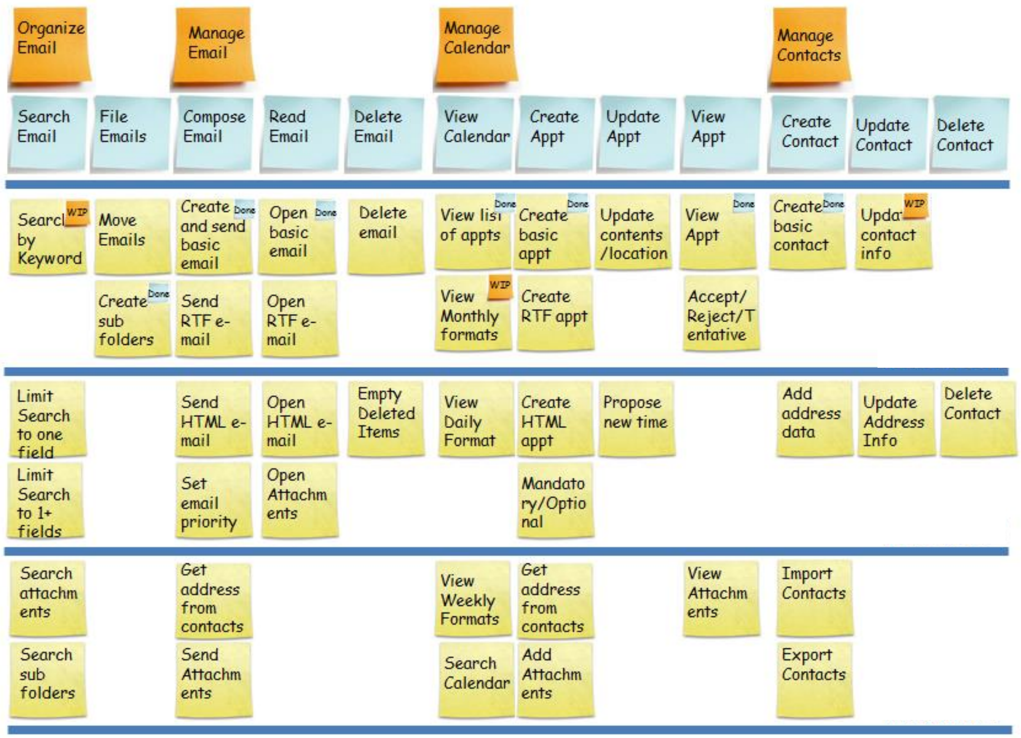
- Sprint end does not work as a milestone:
  - It's timeboxed!  It can neither end early nor late!
  - "Sprint is the new month"
- Since most types of work are done concurrently in an agile project, there are few "intermediate" deliverables that act as milestones
- We measure progress by the size of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

QSM® The Intelligence behind Successful Software Projects

# What can we use for agile milestones?

- We have found a useful milestones:
  - “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

# Minimal Viable Features Identified

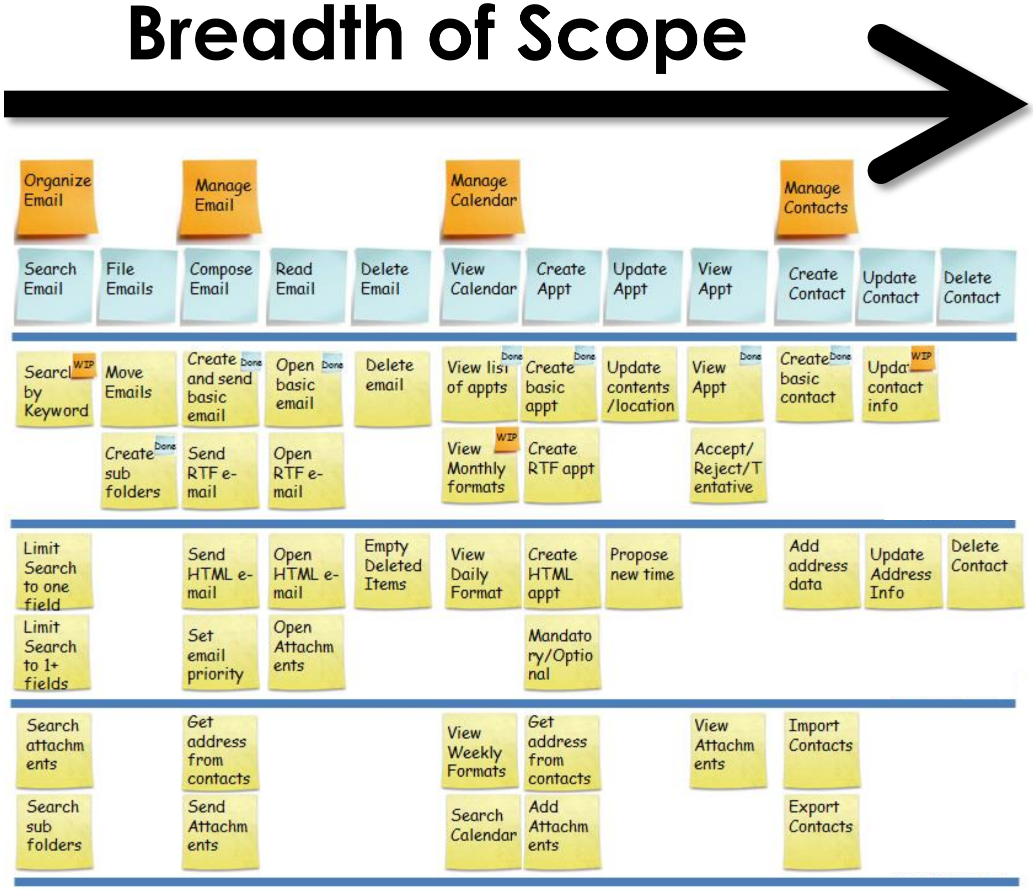


| Organize Email | | Manage Email | | | Manage Calendar | | | | Manage Contacts | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Search Email | File Emails | Compose Email | Read Email | Delete Email | View Calendar | Create Appt | Update Appt | View Appt | Create Contact | Update Contact | Delete Contact |
| Search by Keyword WIP | Move Emails | Create and send basic email Done | Open basic email Done | Delete email | View list of appts Done | Create basic appt Done | Update contents /location | View Appt Done | Create basic contact Done | Update contact info WIP | |
| | Create sub folders Done | Send RTF e-mail | Open RTF e-mail | | View Monthly formats WIP | Create RTF appt | | Accept/ Reject/Tentative | | | |
| Limit Search to one field | | Send HTML e-mail | Open HTML e-mail | Empty Deleted Items | View Daily Format | Create HTML appt | Propose new time | | Add address data | Update Address Info | Delete Contact |
| Limit Search to 1+ fields | | Set email priority | Open Attachments | | | Mandatory/Optional | | | | | |
| Search attachments | | Get address from contacts | | | View Weekly Formats | Get address from contacts | | View Attachments | Import Contacts | | |
| Search sub folders | | Send Attachments | | | Search Calendar | Add Attachments | | | Export Contacts | | |

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

The Intelligence behind Successful Software Projects

# User Story Map (or other representation)



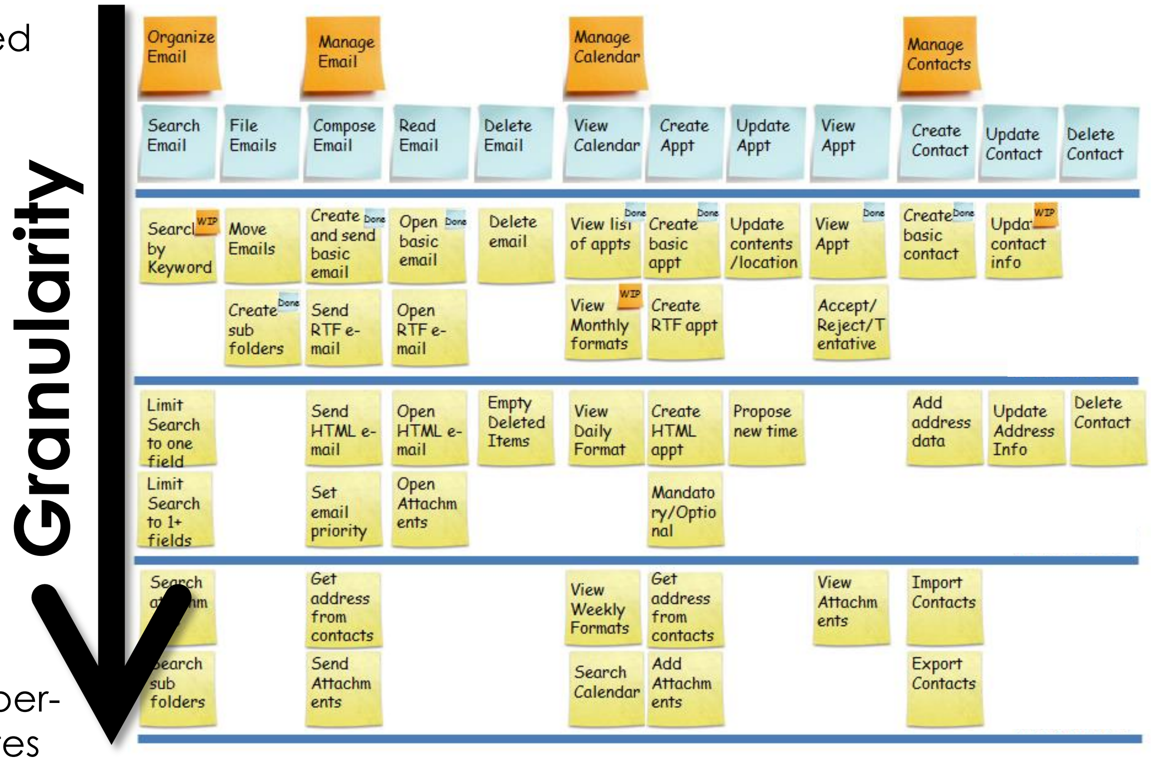
Breadth of Scope

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Organize Email | | Manage Email | | | Manage Calendar | | | | Manage Contacts | | |
| Search Email | File Emails | Compose Email | Read Email | Delete Email | View Calendar | Create Appt | Update Appt | View Appt | Create Contact | Update Contact | Delete Contact |
| Search by Keyword<br>WIP | Move Emails | Create and send basic email<br>Done | Open basic email<br>Done | Delete email | View list of appts<br>Done | Create basic appt<br>Done | Update contents /location | View Appt<br>Done | Create basic contact<br>Done | Update contact info<br>WIP | |
| | Create sub folders<br>Done | Send RTF e-mail | Open RTF e-mail | | View Monthly formats<br>WIP | Create RTF appt | | Accept/ Reject/Tentative | | | |
| Limit Search to one field | | Send HTML e-mail | Open HTML e-mail | Empty Deleted Items | View Daily Format | Create HTML appt | Propose new time | | Add address data | Update Address Info | Delete Contact |
| Limit Search to 1+ fields | | Set email priority | Open Attachments | | | Mandatory/Optional | | | | | |
| Search attachments | | Get address from contacts | | | View Weekly Formats | Get address from contacts | | View Attachments | Import Contacts | | |
| Search sub folders | | Send Attachments | | | Search Calendar | Add Attachments | | | Export Contacts | | |

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

# User Story Map (or other representation)

User-sized epics

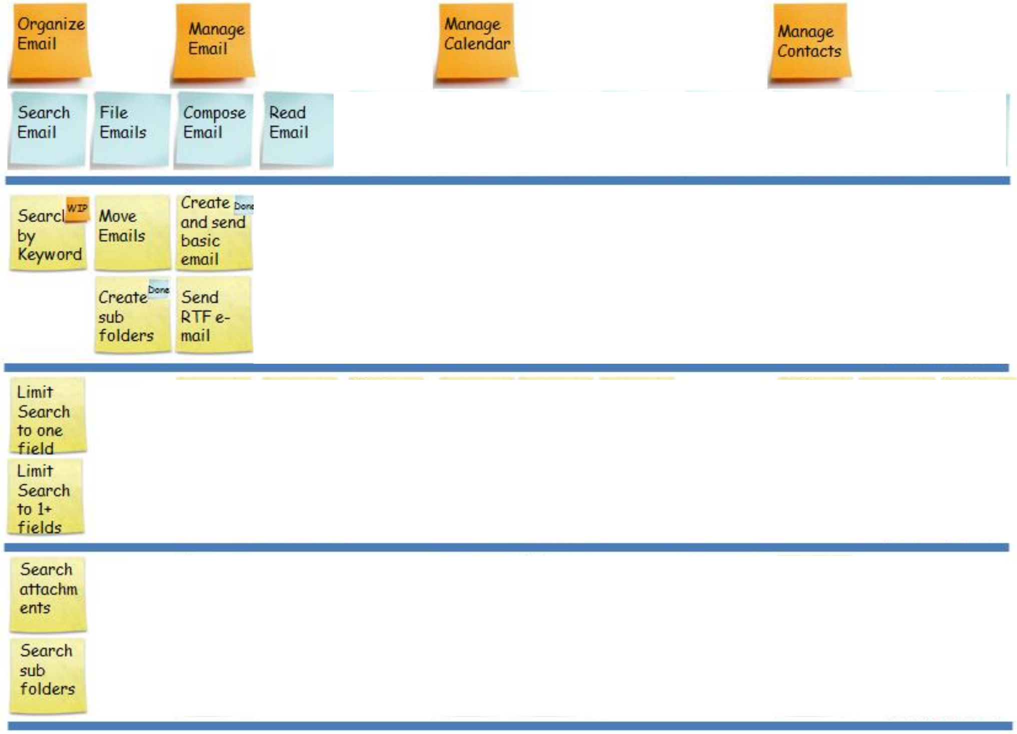


Developer-sized bites

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

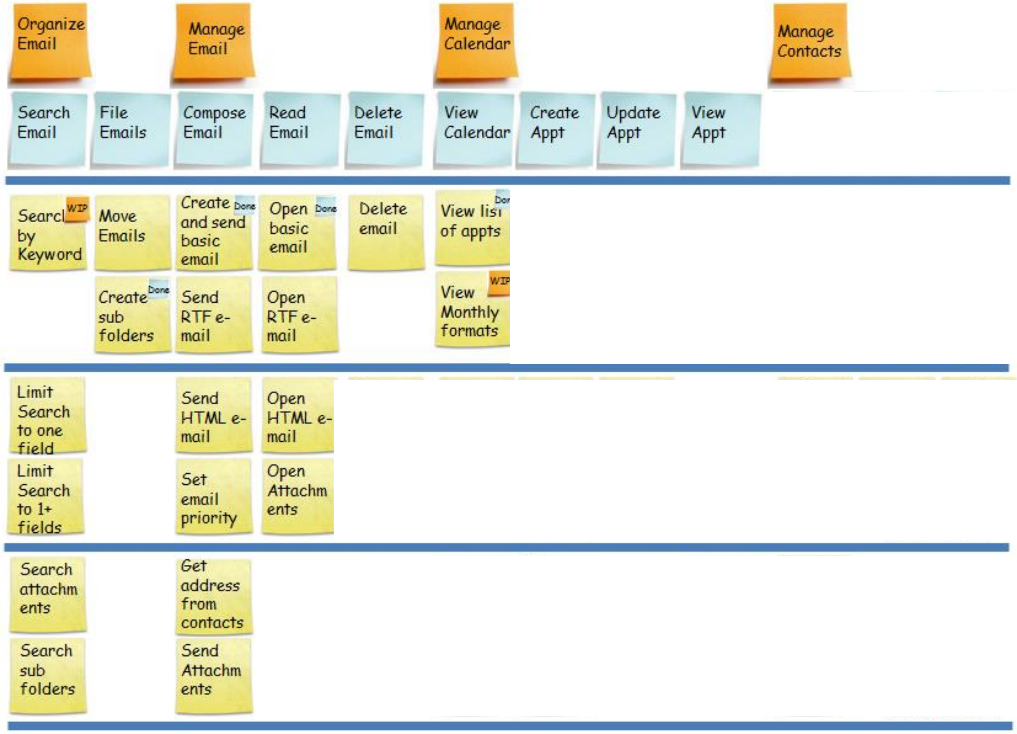
QSM® The Intelligence behind Successful Software Projects

# Story map emerges as project progresses



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

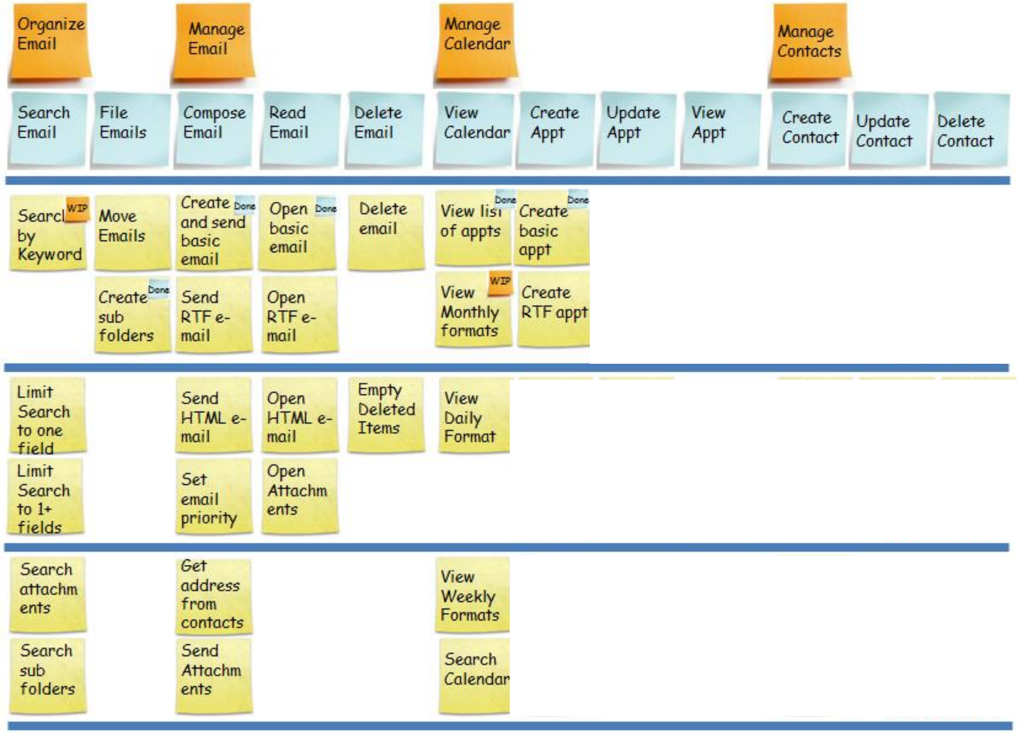
# A few sprints later, more has emerged



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

QSM® The Intelligence behind Successful Software Projects

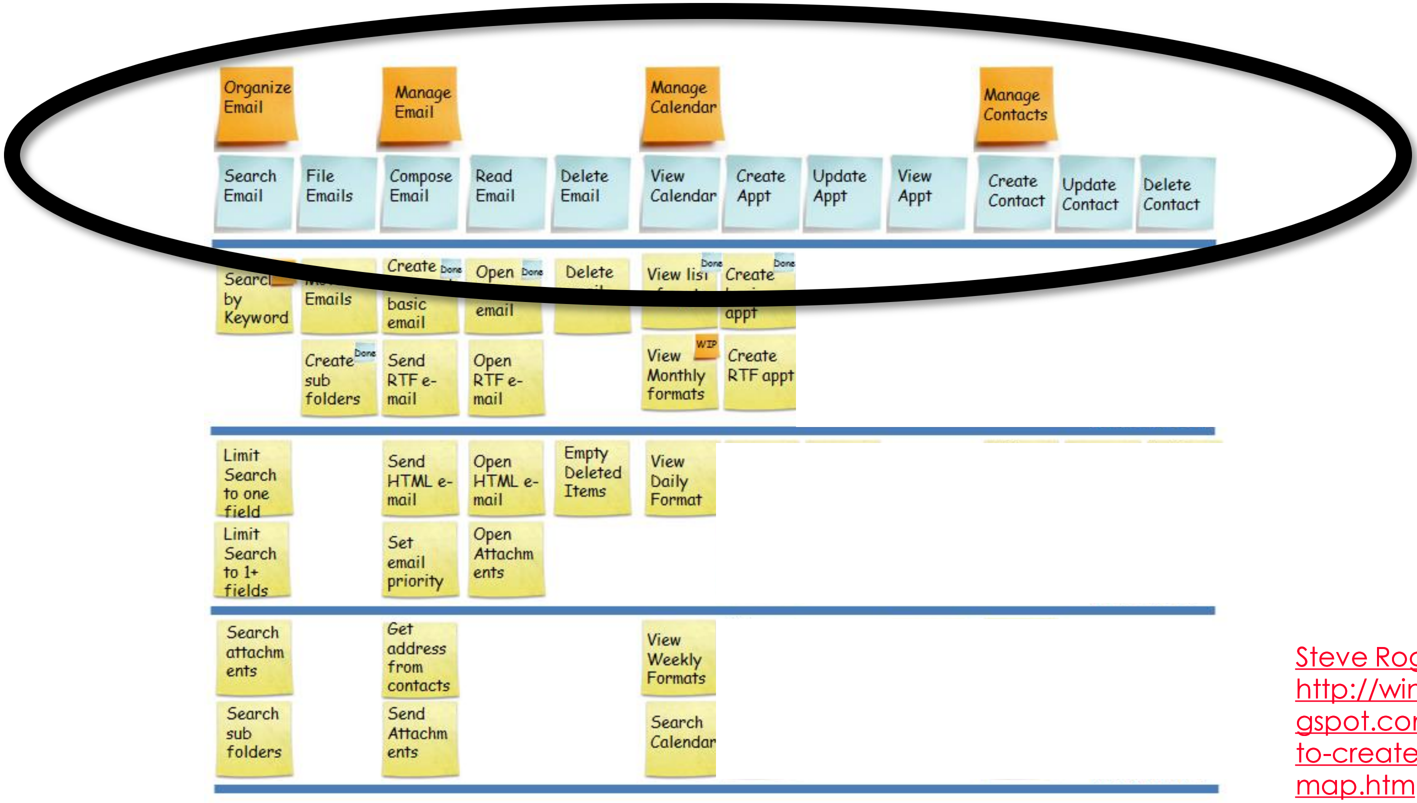
# And still later…



| Organize Email | | Manage Email | | | Manage Calendar | | | | Manage Contacts | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Search Email | File Emails | Compose Email | Read Email | Delete Email | View Calendar | Create Appt | Update Appt | View Appt | Create Contact | Update Contact | Delete Contact |
| Search by Keyword WIP | Move Emails | Create and send basic email Done | Open basic email Done | Delete email | View list of appts Done | Create basic appt Done | | | | | |
| | Create sub folders Done | Send RTF e-mail | Open RTF e-mail | | View Monthly formats WIP | Create RTF appt | | | | | |
| Limit Search to one field | | Send HTML e-mail | Open HTML e-mail | Empty Deleted Items | View Daily Format | | | | | | |
| Limit Search to 1+ fields | | Set email priority | Open Attachments | | | | | | | | |
| Search attachments | | Get address from contacts | | | View Weekly Formats | | | | | | |
| Search sub folders | | Send Attachments | | | Search Calendar | | | | | | |

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® The Intelligence behind Successful Software Projects

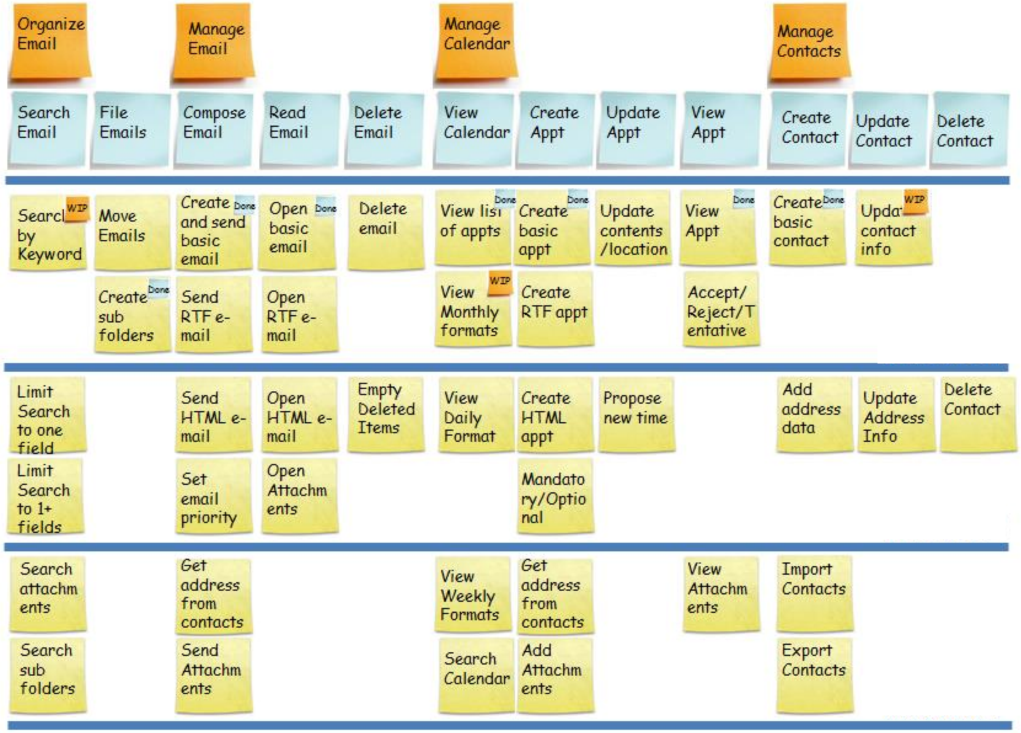
# Minimal Viable Features Stable

***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.***

ilist.blo
3/how-
y-
map.htm

# There’s still more grooming to do



| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Organize Email | | Manage Email | | | Manage Calendar | | | | Manage Contacts | | |
| Search Email | File Emails | Compose Email | Read Email | Delete Email | View Calendar | Create Appt | Update Appt | View Appt | Create Contact | Update Contact | Delete Contact |
| Search by Keyword WIP | Move Emails | Create and send basic email Done | Open basic email Done | Delete email | View list of appts Done | Create basic appt Done | Update contents /location | View Appt Done | Create basic contact Done | Update contact info WIP | |
| | Create sub folders Done | Send RTF e-mail | Open RTF e-mail | | View Monthly formats WIP | Create RTF appt | | Accept/Reject/Tentative | | | |
| Limit Search to one field | | Send HTML e-mail | Open HTML e-mail | Empty Deleted Items | View Daily Format | Create HTML appt | Propose new time | | Add address data | Update Address Info | Delete Contact |
| Limit Search to 1+ fields | | Set email priority | Open Attachments | | | Mandatory/Optional | | | | | |
| Search attachments | | Get address from contacts | | | View Weekly Formats | Get address from contacts | | View Attachments | Import Contacts | | |
| Search sub folders | | Send Attachments | | | Search Calendar | Add Attachments | | | Export Contacts | | |

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

QSM® The Intelligence behind Successful Software Projects

# 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® The Intelligence behind Successful Software Projects

# 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

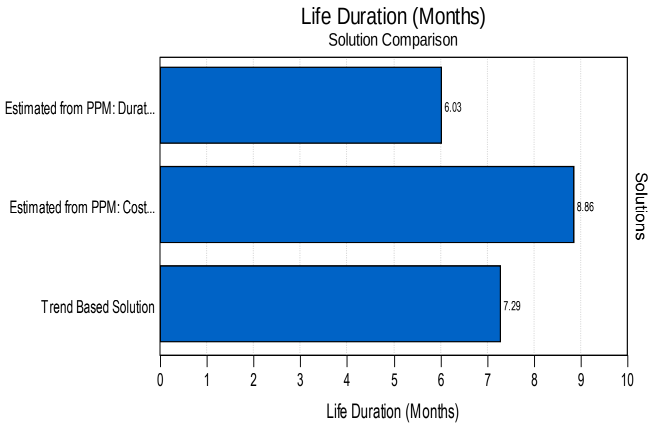
# The Mythical Person-Month

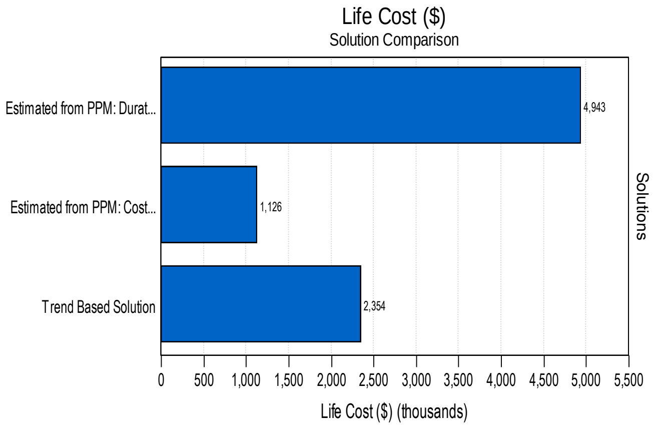
- **While duration clearly varies with team size, it's tempting to think that effort is constant,**
-
-

***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!***

# 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® The Intelligence behind Successful Software Projects

# 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 affects the productivity of a team?

- Many, MANY factors affect productivity of a team working on a release:

*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.*

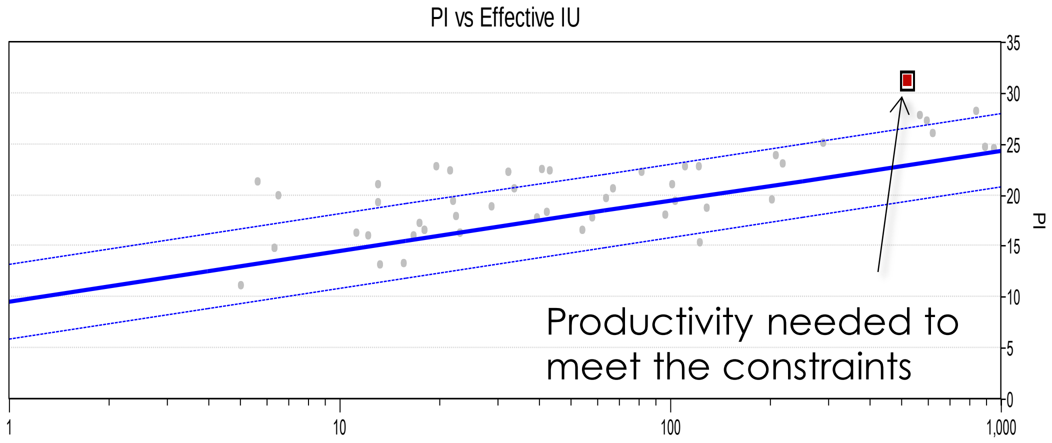
way of measuring this

# 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

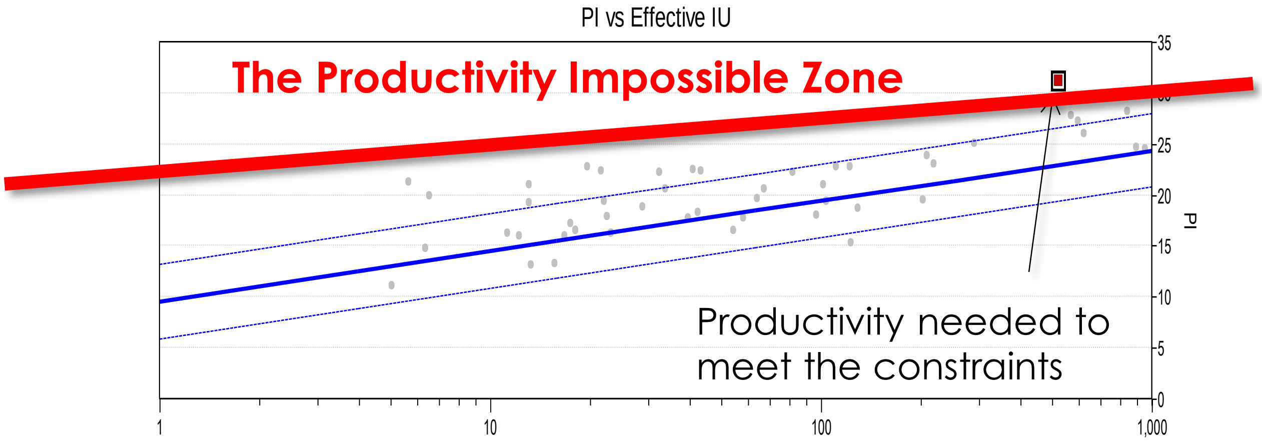
- **We've all seen management put arbitrary constraints on a project:**
  - **"That scope is great and will bring our customers the value they want.  Now get it done in eight months for a million dollars, OK?"**



Productivity needed to meet the constraints

# 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?”


PI vs Effective IU

The Productivity Impossible Zone

Productivity needed to meet the constraints

The Intelligence behind Successful Software Projects

# Productivity can help assess feasibility

- We've all seen management put arbitrary constraints on a project:

ers

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

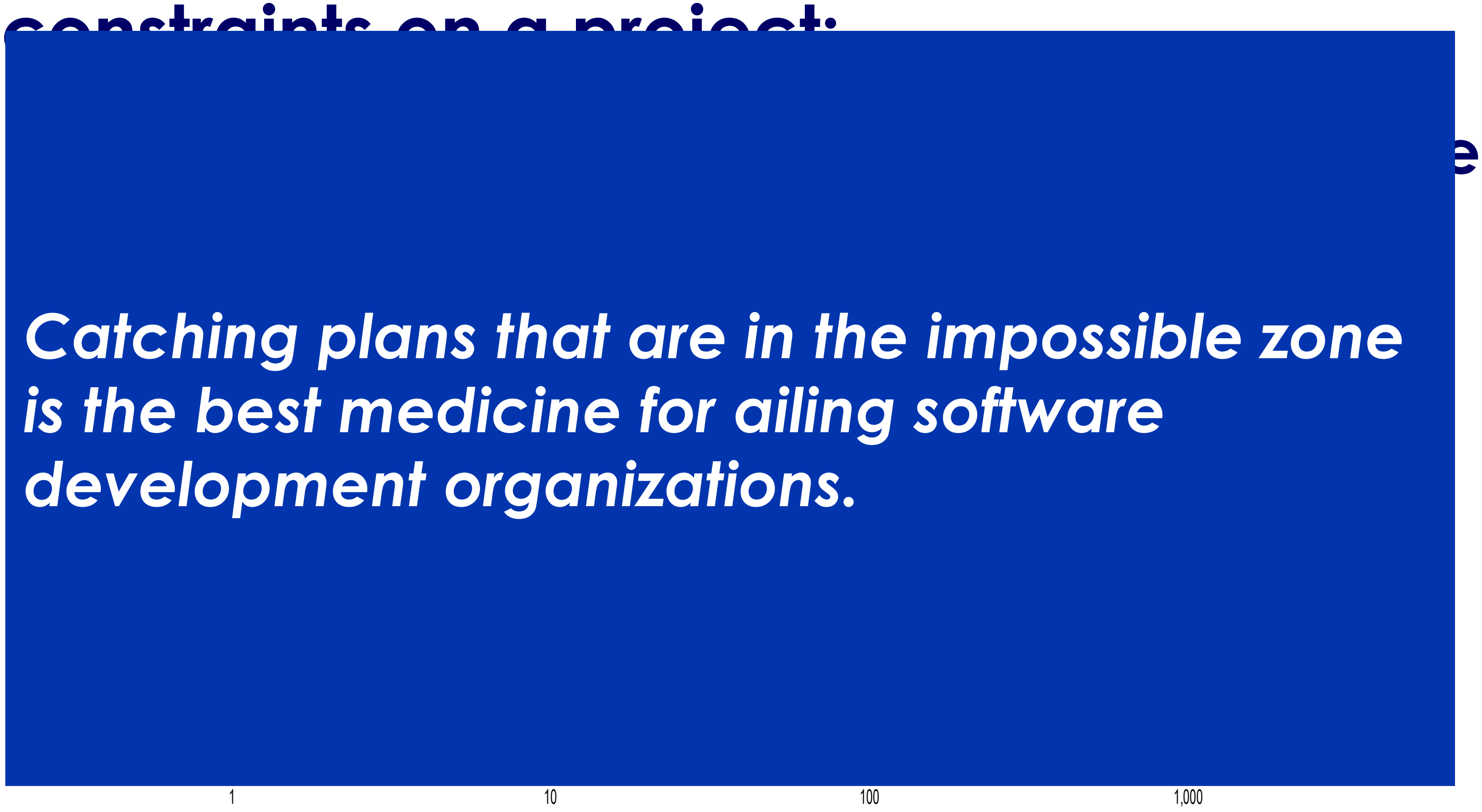
*"Try hard and do your best" won't help!*

*You have history on your side to renegotiate the plan.*

# Productivity can help assess feasibility

- We’ve all seen management put arbitrary constraints on a project:

ers

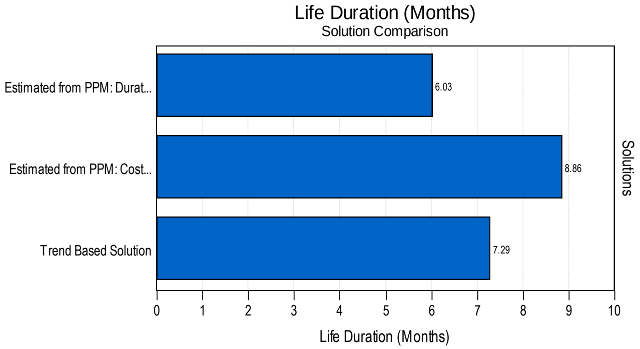


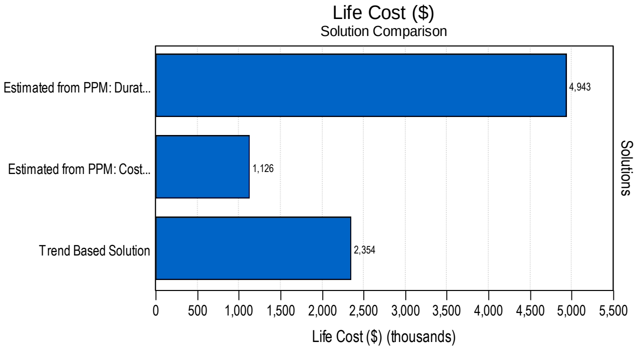
*Catching plans that are in the impossible zone is the best medicine for ailing software development organizations.*

QSM® The Intelligence behind Successful Software Projects

# 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

# 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."

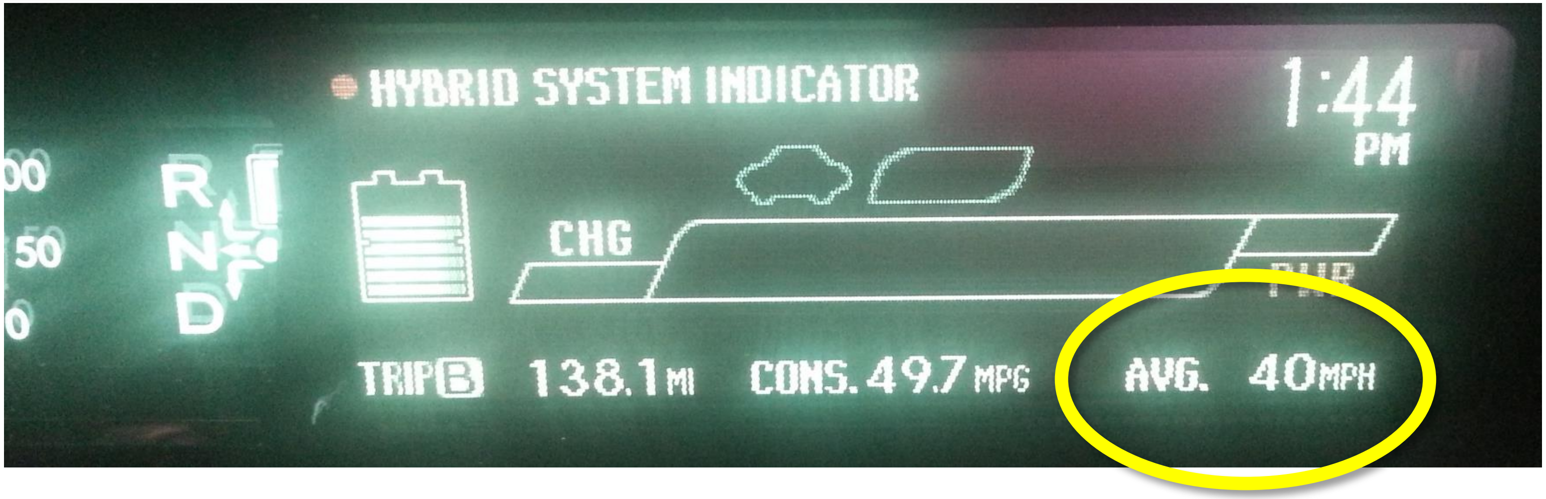
H. L. Mencken

- **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**

# 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???

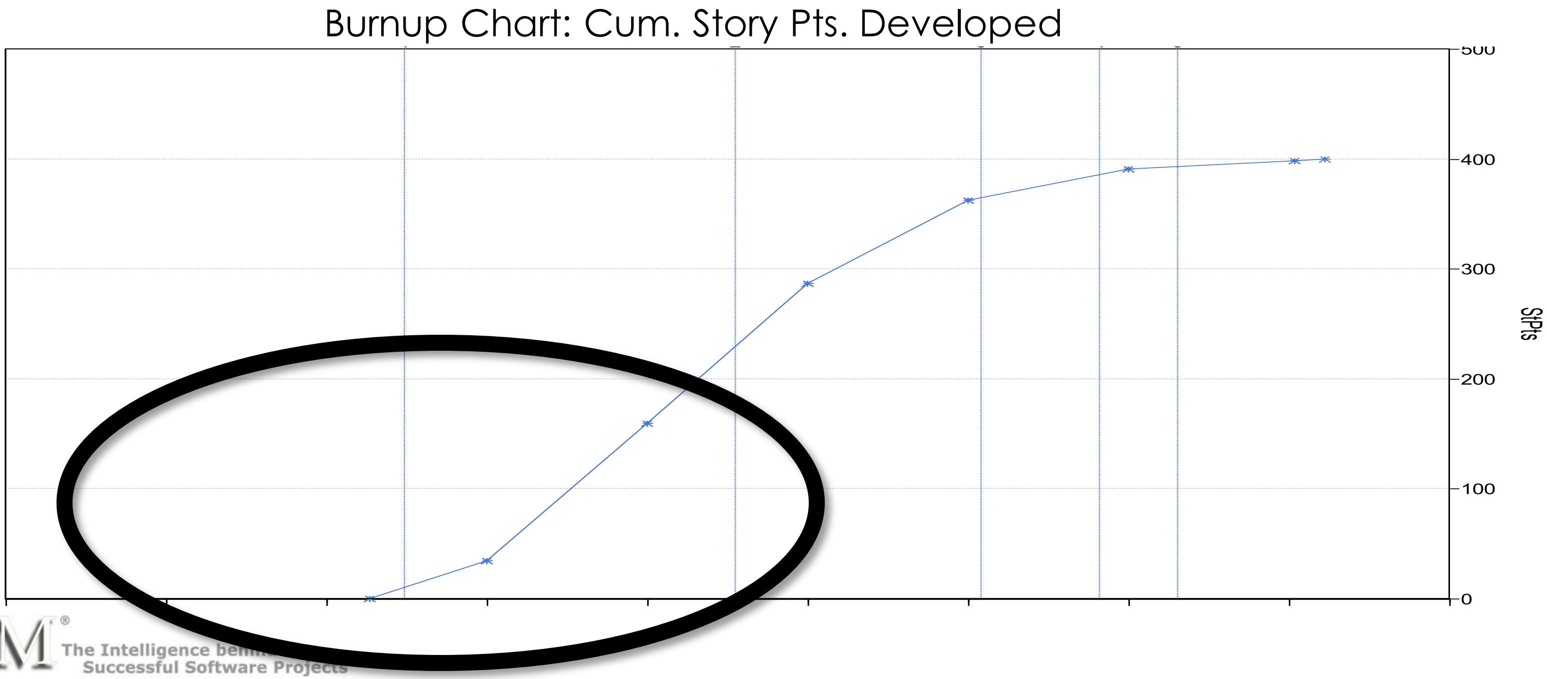


- Some driving  was in the city, between 20-30 mph.
- Some was on the highway, driving between 55-65 mph.
- I almost never drive near 40 mph.

The Intelligence behind
Successful Software Projects

# Velocity is not constant during a project

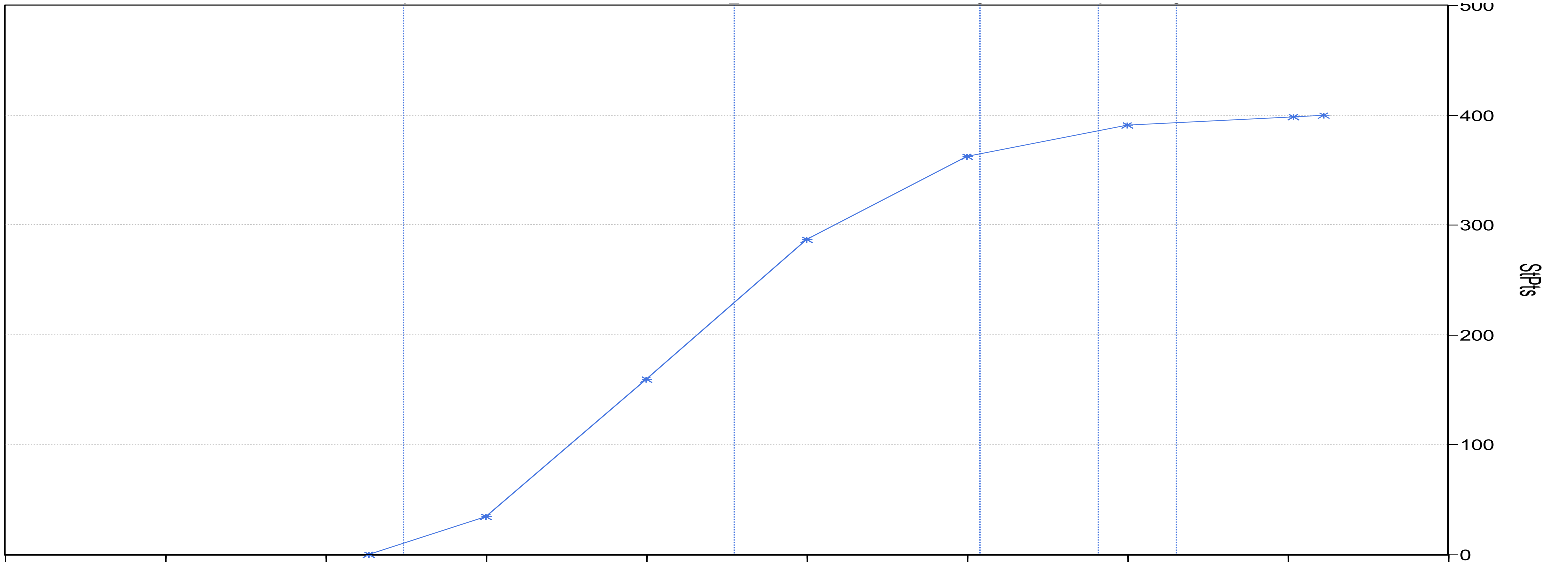
- **Most people know velocity starts off lower at the beginning of a project**



The Intelligence behind Successful Software Projects

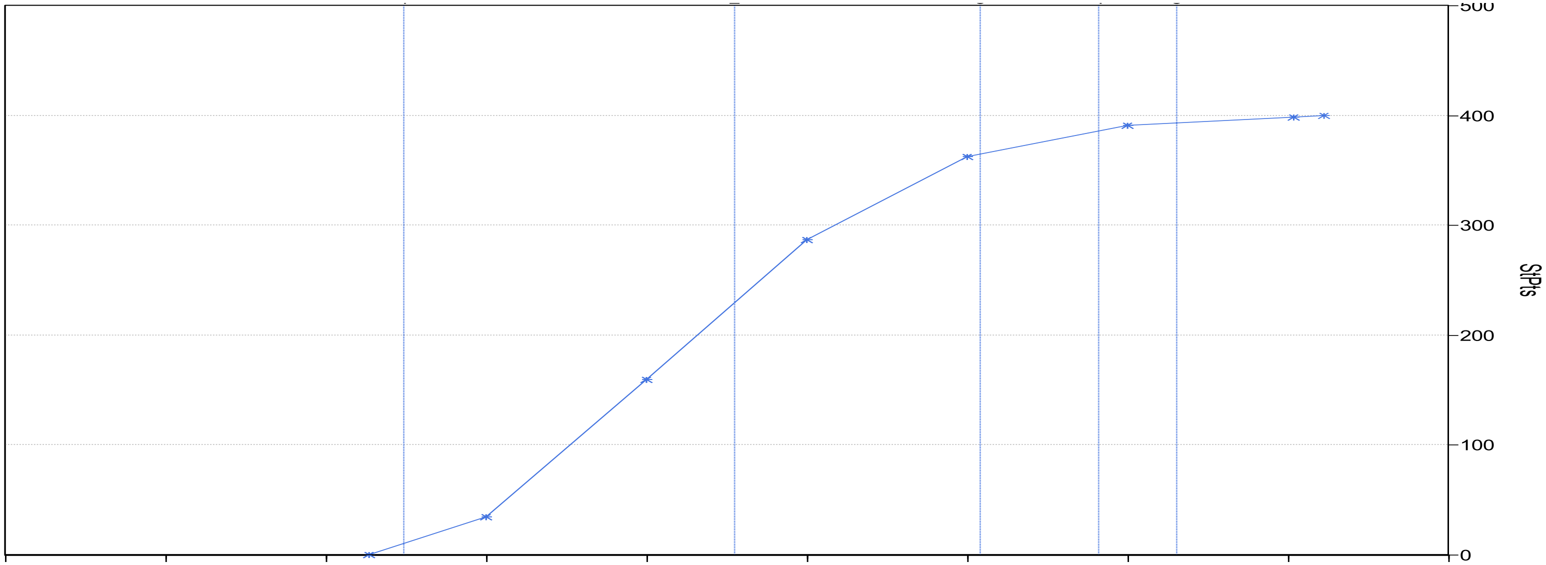
# Velocity is not constant during a project

- **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!

# 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.

# What’s the alternative?

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

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

k is derived from Size and Productivity Index

# 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”

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

k is derived from Size and Productivity Index

# 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

# How do agile methods fit in?

- The values used for Productivity Index come from a trend calibrated from historical data of similar

ts

- gile

*The QSM SLIM-Estimate Agile Template uses the QSM Business Agile trend, based on agile projects contributed to the QSM Database.*

*You can use this industry wide data or substitute a trend from your own data.*

*QSM continues to collect data and periodically updates our industry-wide trends.*

# 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….

QSM® The Intelligence behind Successful Software Projects

# Velocity isn't all bad!

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

QSM® The Intelligence behind Successful Software Projects

# Project Control

> ***"The shortest distance between two points is under construction"***
>
> *Noelie Altito*

QSM® The Intelligence behind Successful Software Projects

# 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 The Intelligence behind Successful Software Projects

# Some reasons projects are late

- Lower productivity than expected

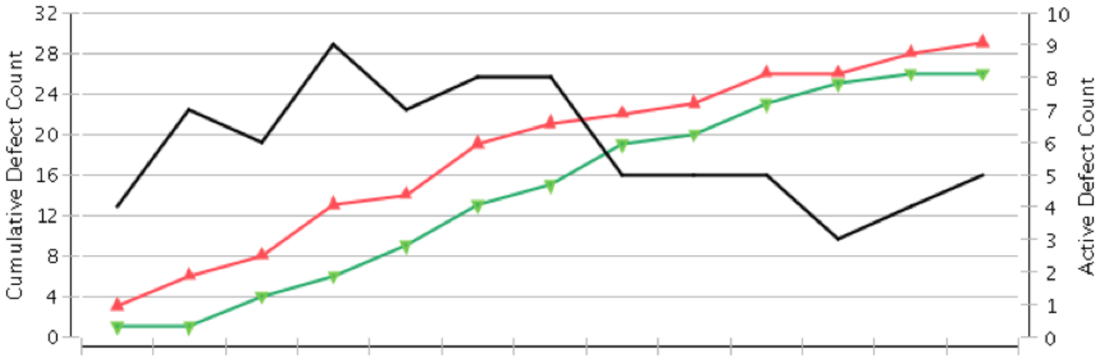
ect

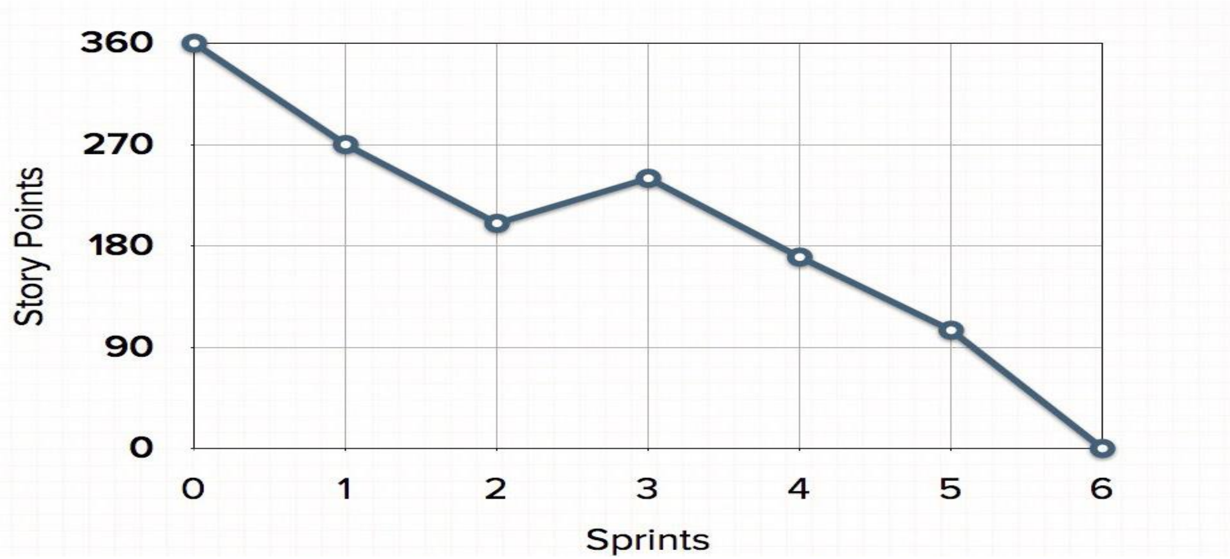
*You can't necessarily see these problems day to day.*

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

QSM® The Intelligence behind Successful Software Projects

# Agile teams collect metrics



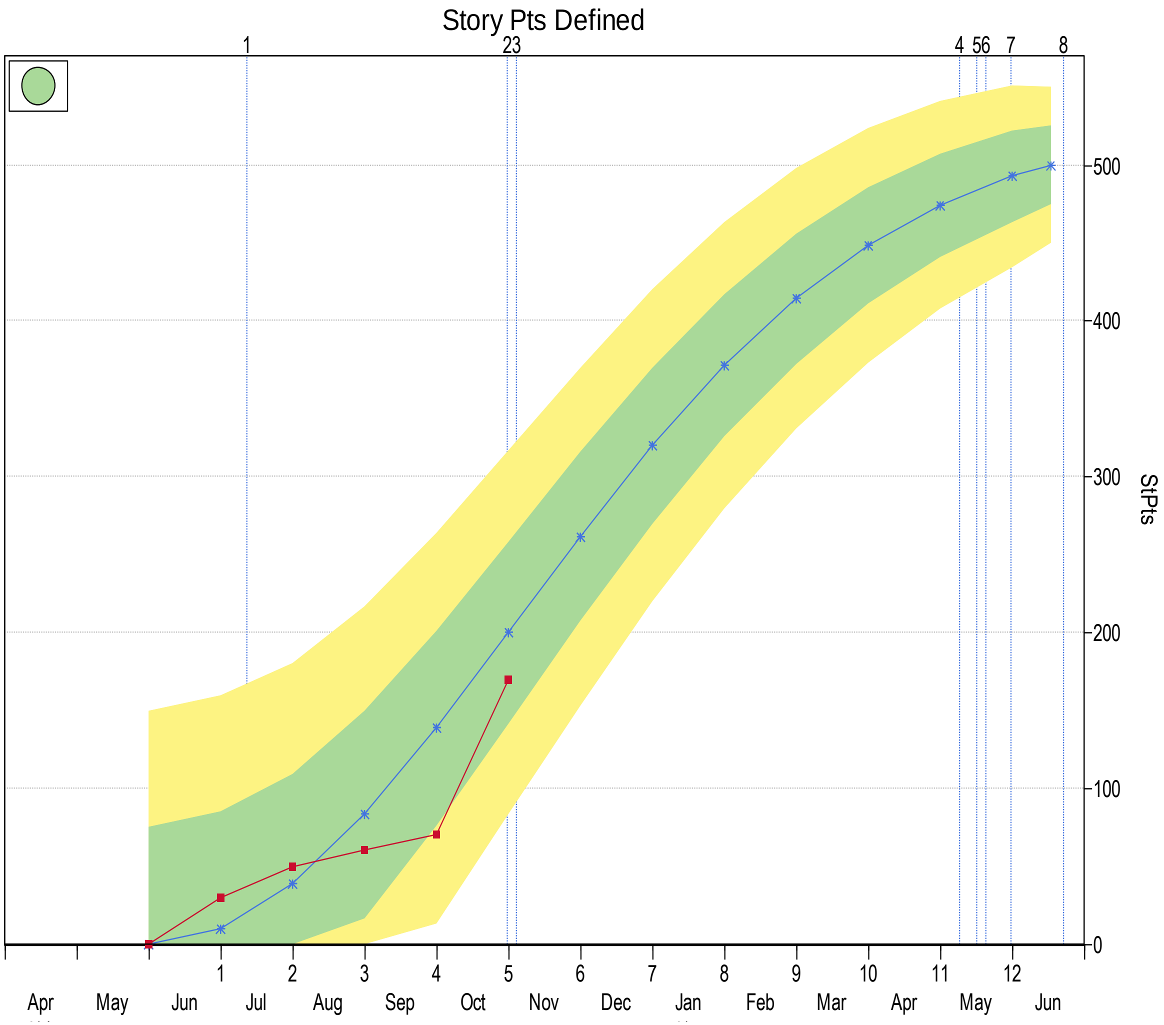


The Intelligence behind
Successful Software Projects

# Purpose of metrics

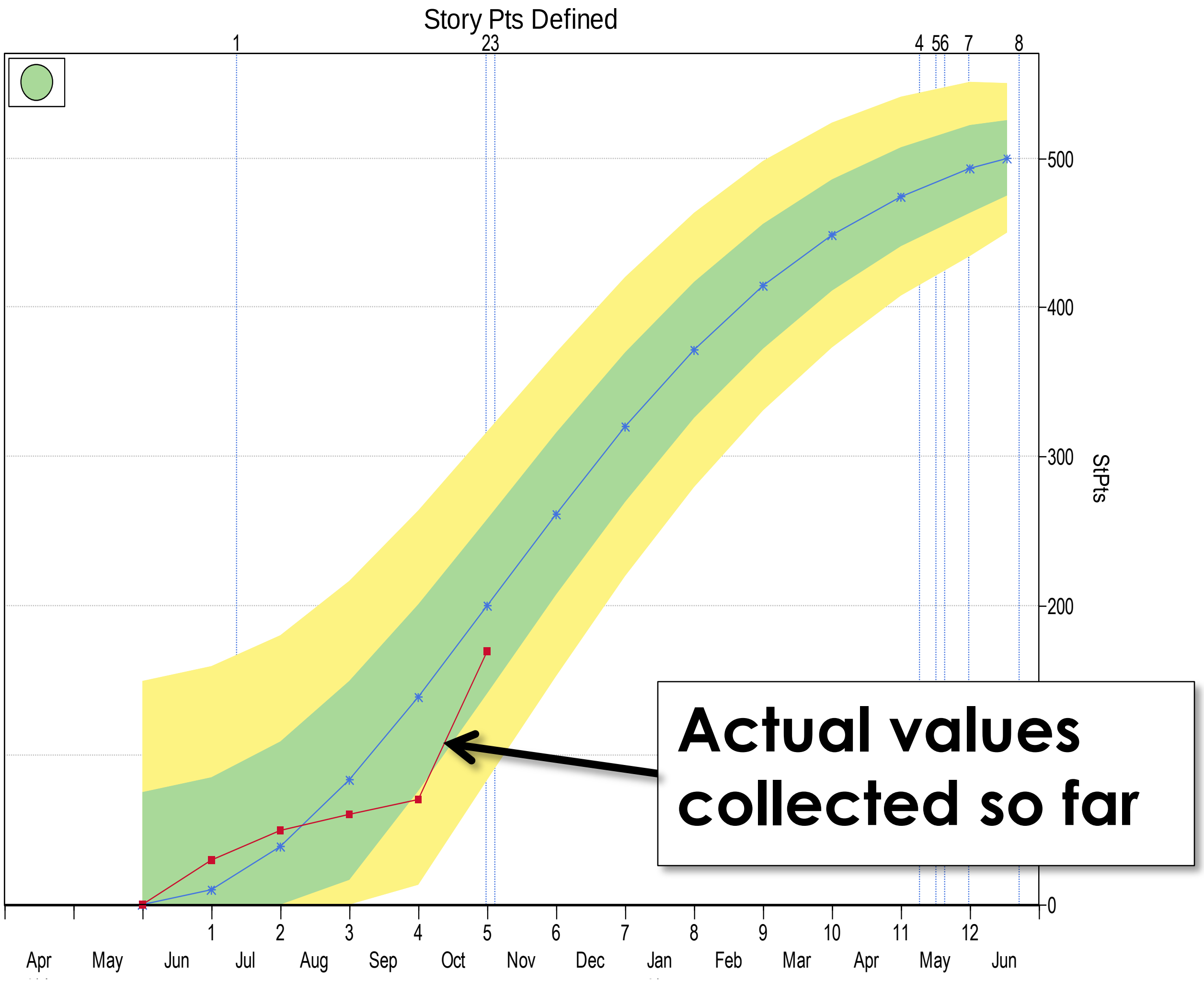
- Quick course corrections
- Spot potential trouble before it happens
- Replan if needed
- Collect historical data for use in future estimates and process improvement activities

# Anatomy of a metric



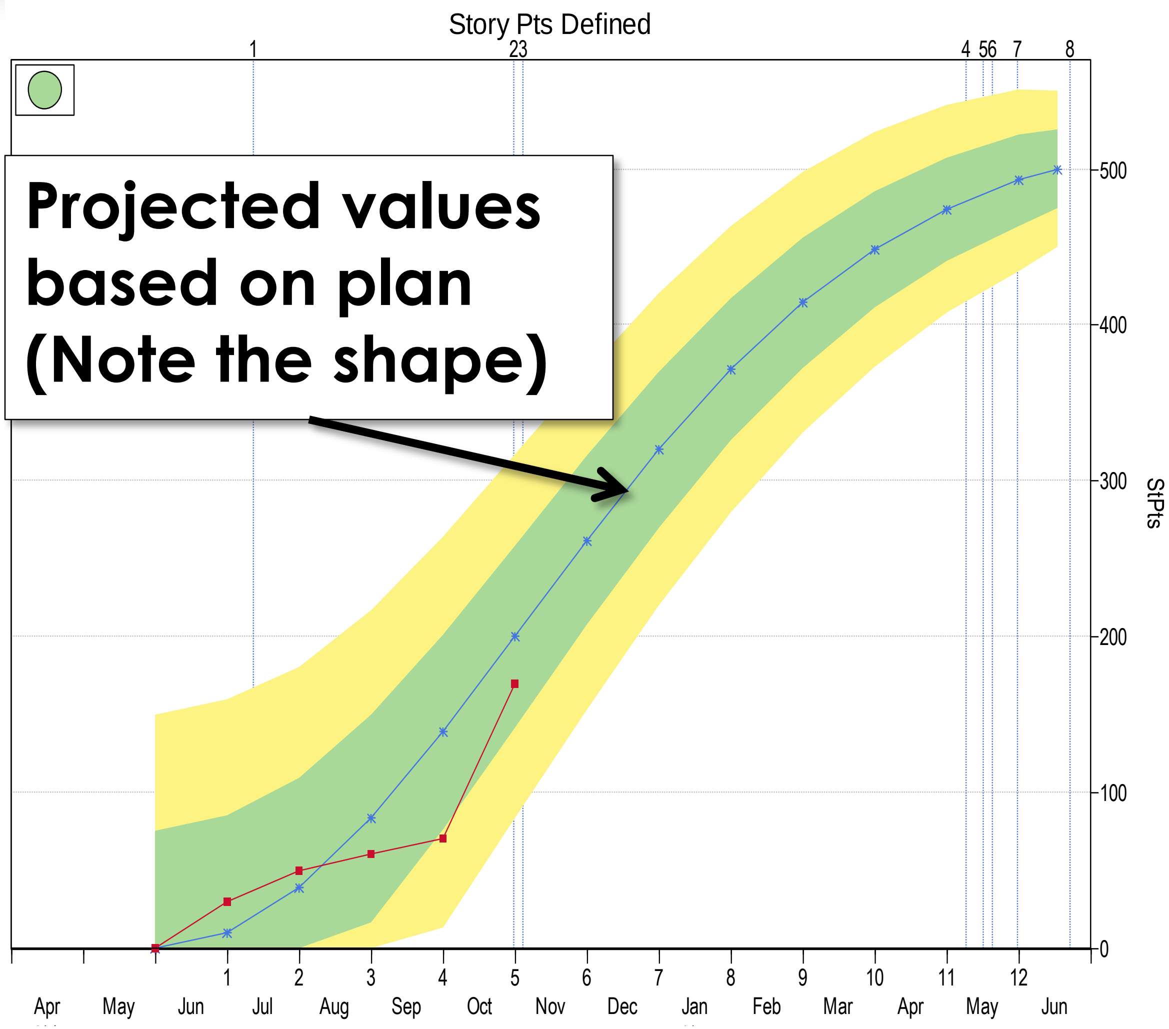
QSM® The Intelligence behind Successful Software Projects

# Anatomy of a metric



Actual values collected so far

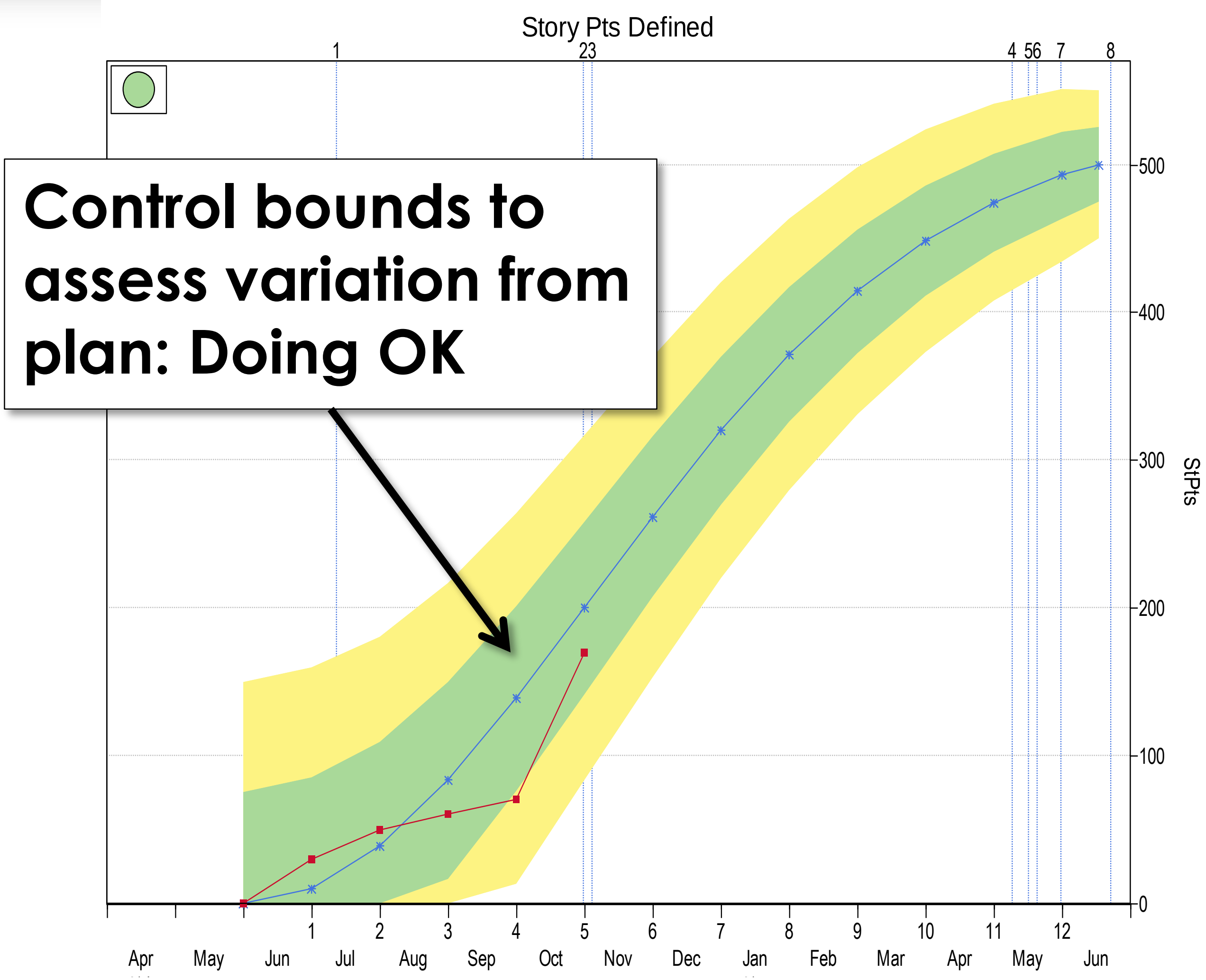
# Anatomy of a metric



**Projected values based on plan (Note the shape)**

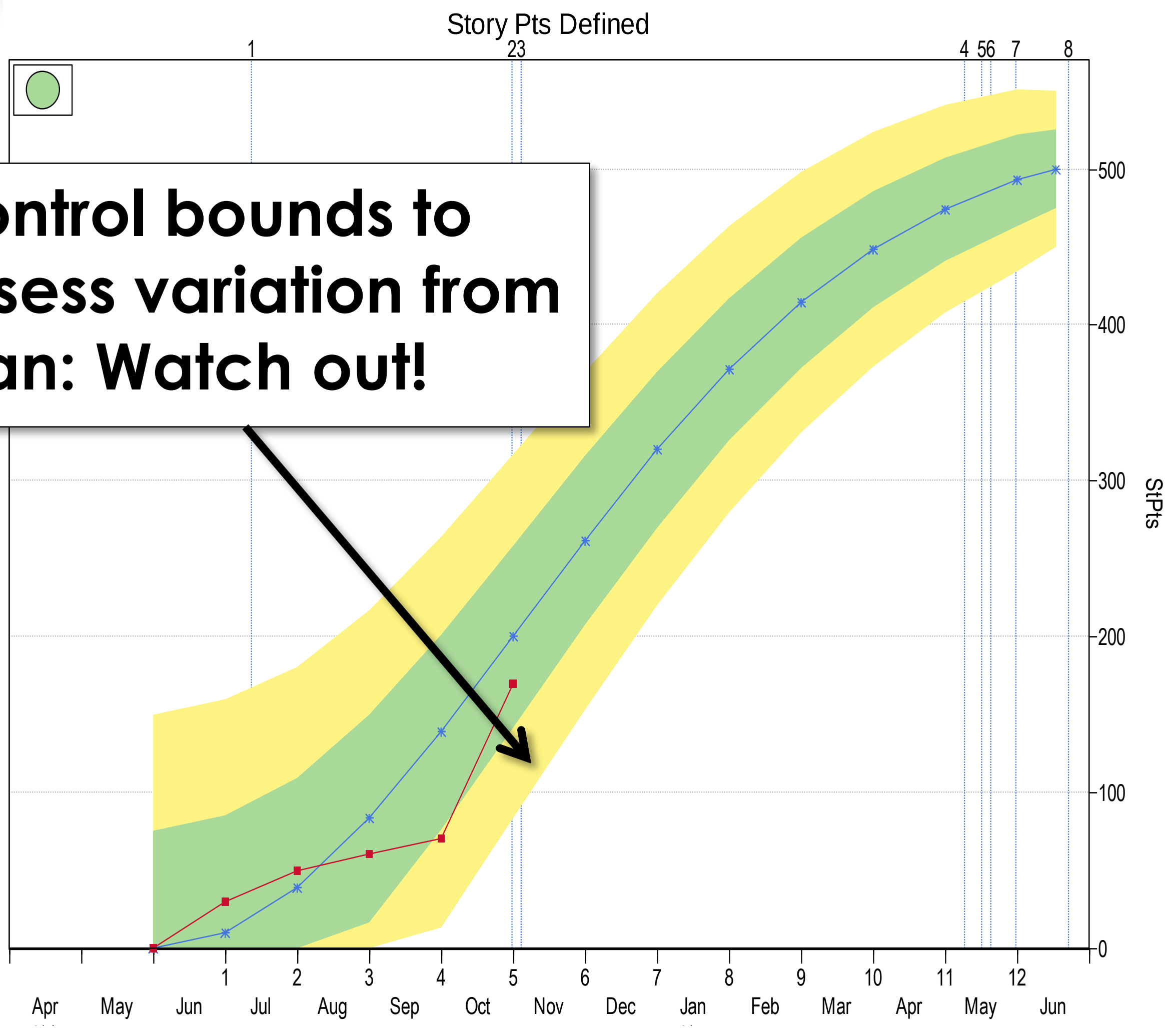
The Intelligence behind Successful Software Projects

# Anatomy of a metric



Control bounds to assess variation from plan: Doing OK

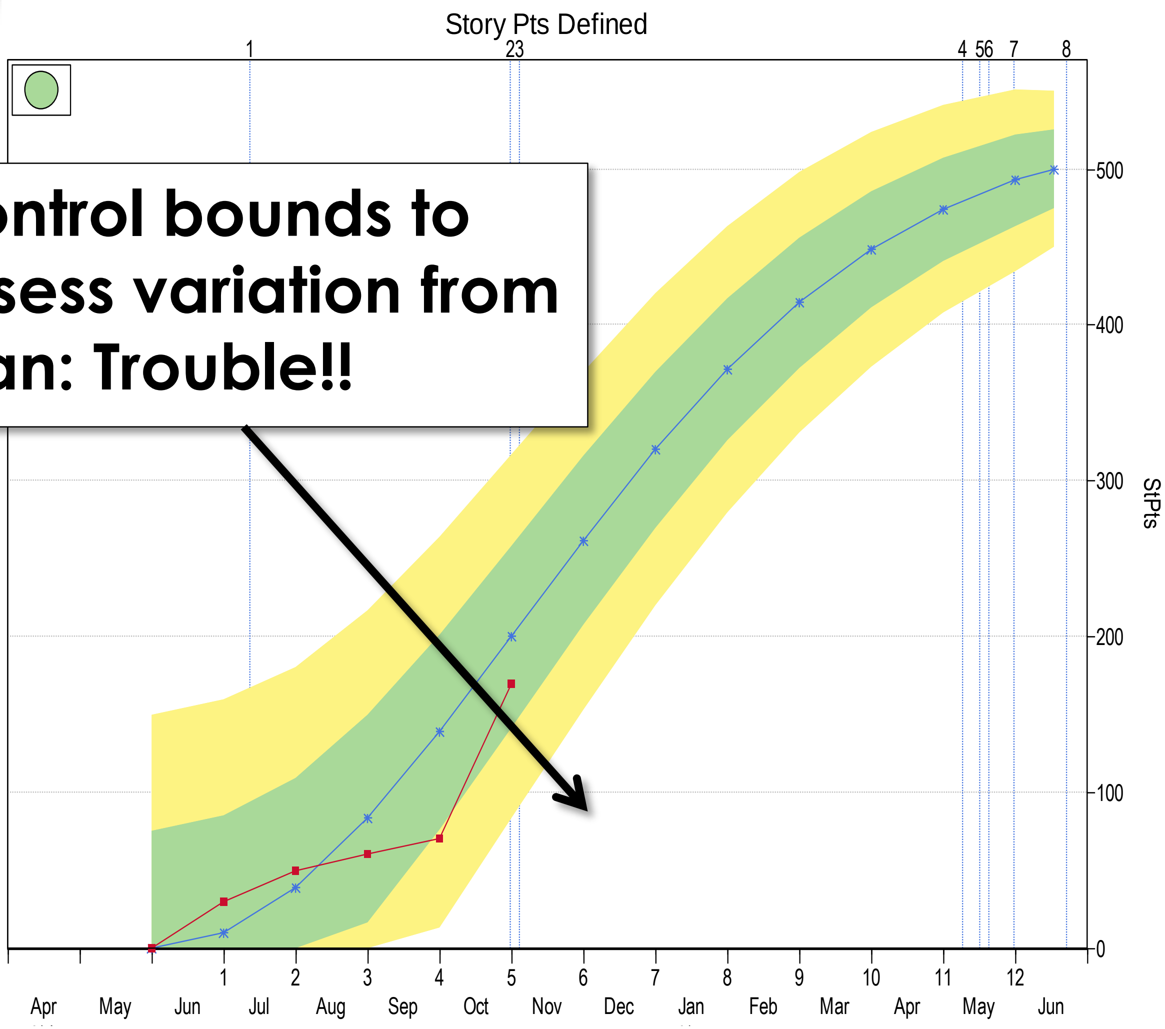
# Anatomy of a metric



Control bounds to assess variation from plan: Watch out!

QSM® The Intelligence behind Successful Software Projects

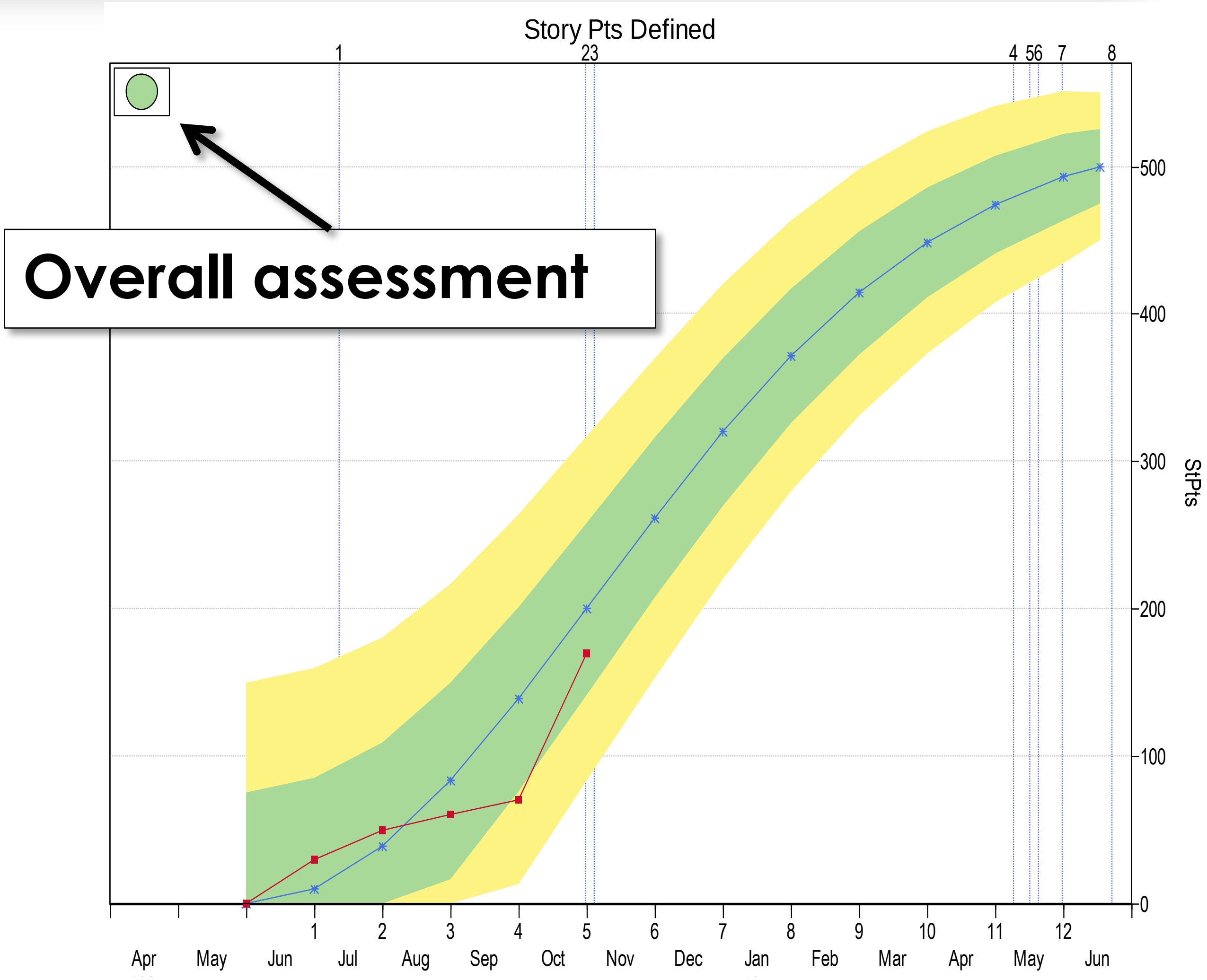
# Anatomy of a metric



**Control bounds to assess variation from plan: Trouble!!**

QSM® The Intelligence behind Successful Software Projects

# Anatomy of a metric



Overall assessment

# 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

# Combine metrics from various tools

- (This is not a definitive list)

*With QSM SLIM-Control, you can choose from our standard metrics and define your own custom metrics to build a metrics program that fits your organization.*

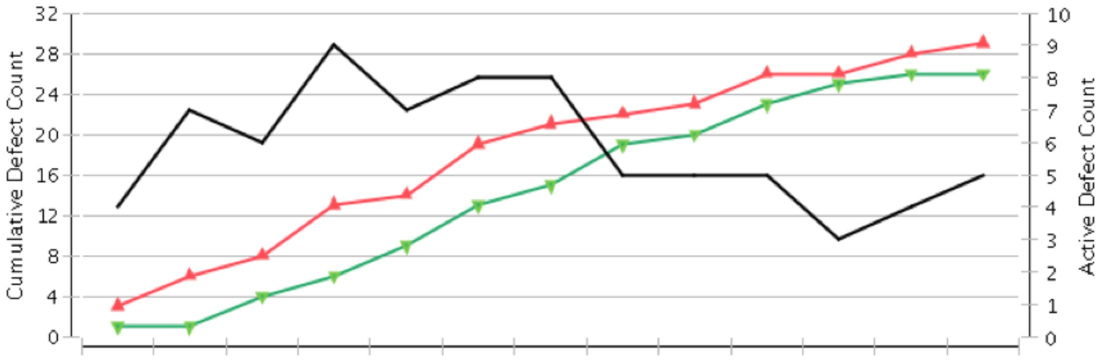
*You can compare actuals to plans and assess whether the variation from plan is ok or hints at a problem.*

*You can get an overall assessment.*

*You can reforecast based on actual productivity and changed assumptions.*

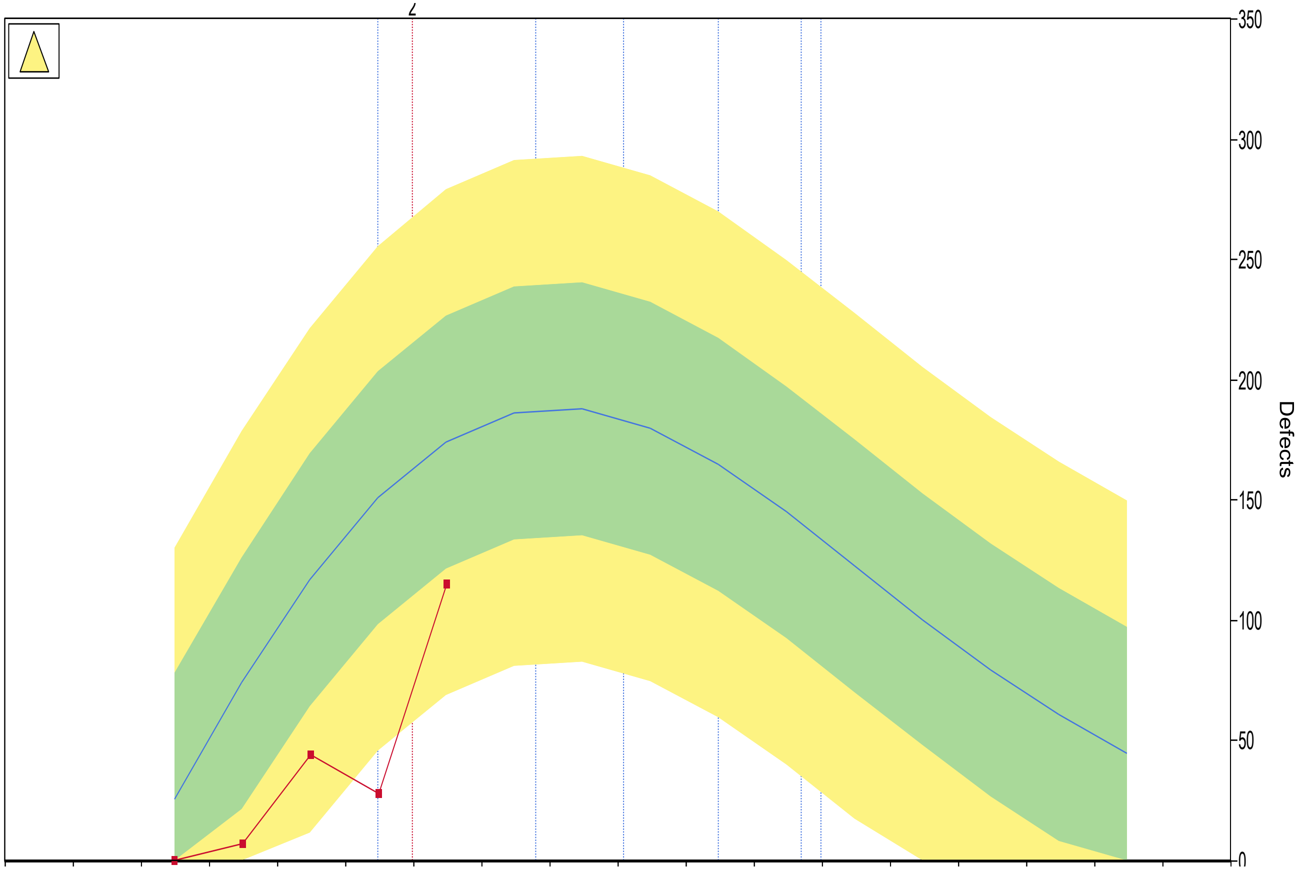
# Defects and control bounds

- **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

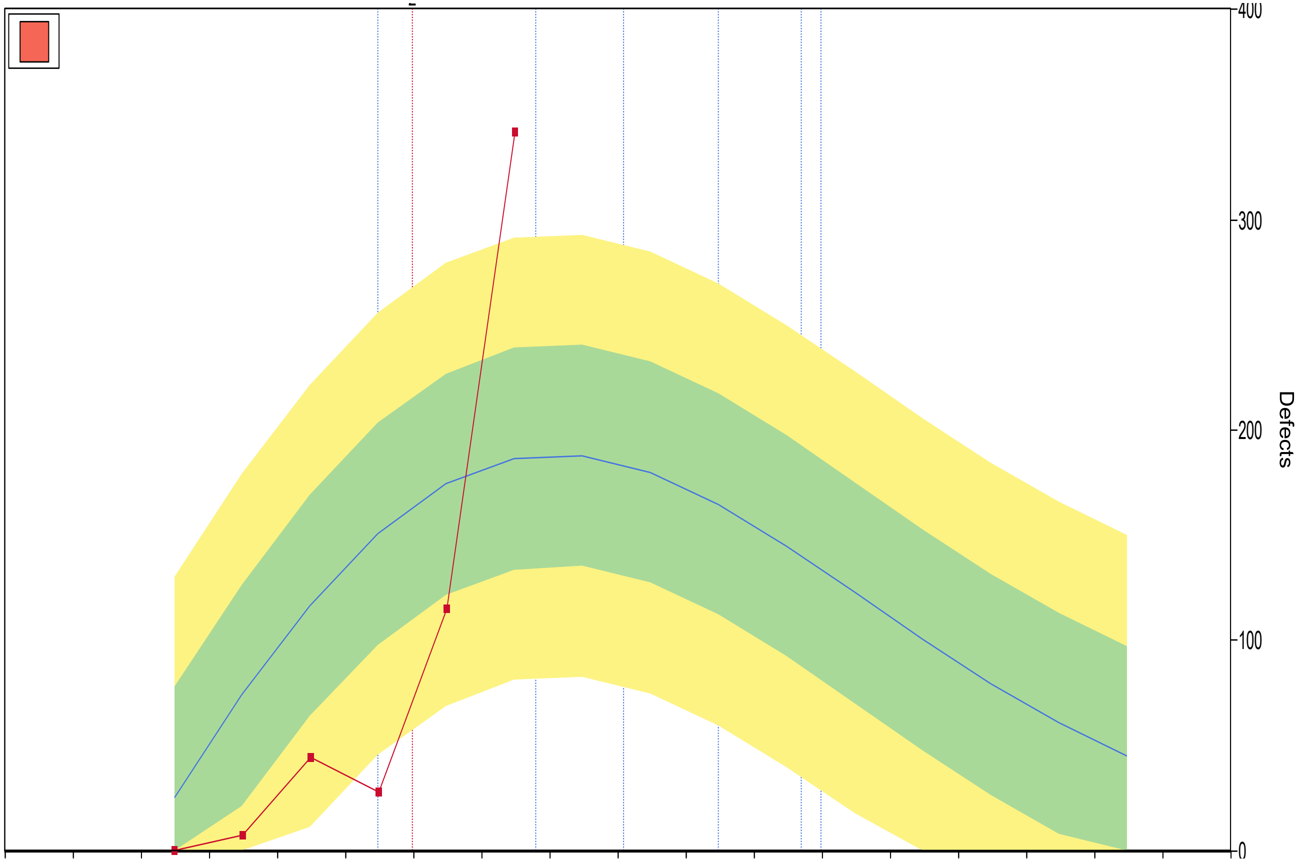
- **Too few is as bad as too many!**



The Intelligence behind Successful Software Projects

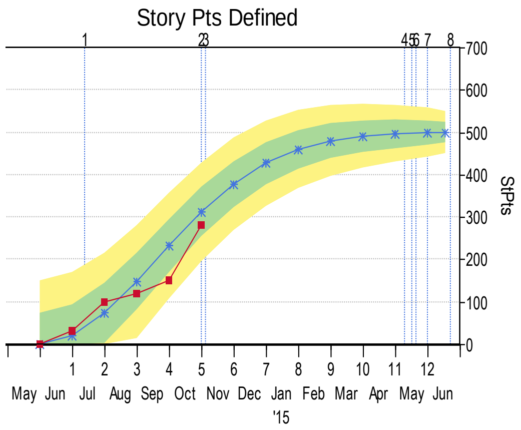
# Defects and control bounds

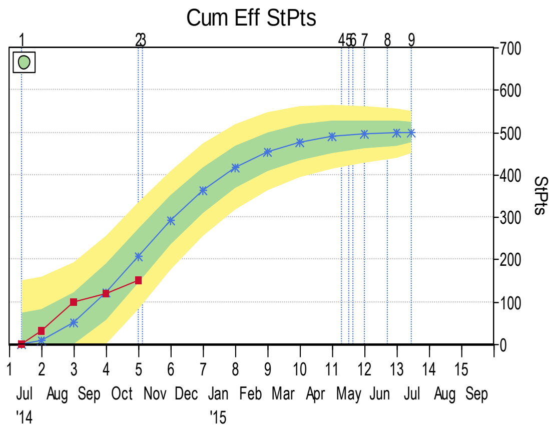
- **Too few is as bad as too many!**

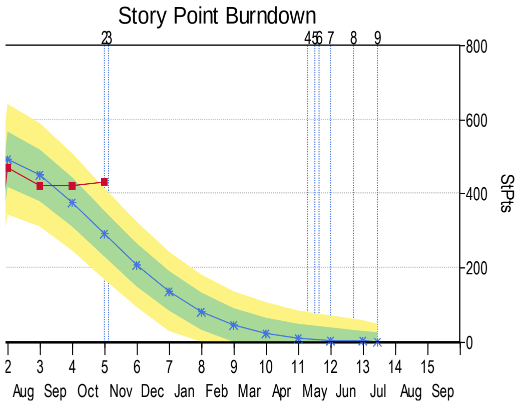


# Monitoring story definition

## 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

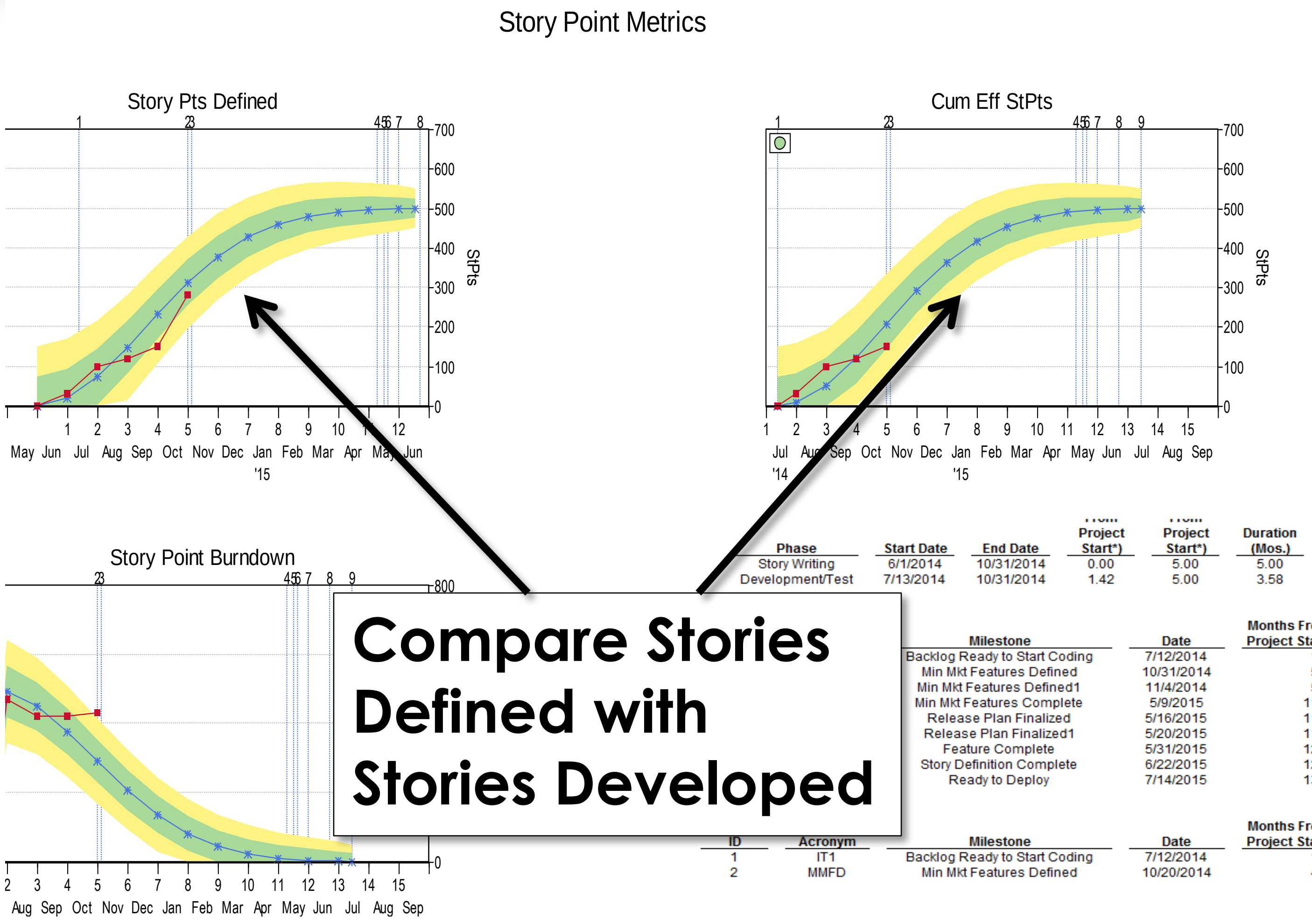
| Milestone ID | Milestone Acronym | Milestone | Date | Months From Project Start* |
|---|---|---|---|---|
| 1 | IT1 | Backlog Ready to Start Coding | 7/12/2014 | 1.39 |
| 2 | MMFD | Min Mkt Features Defined | 10/31/2014 | 5.00 |
| 3 | MMFD1 | Min Mkt Features Defined1 | 11/4/2014 | 5.13 |
| 4 | MMFC | 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 |

QSM® The Intelligence behind Successful Software Projects

# Monitoring story definition



| Phase | Start Date | End Date | Project Start*) | 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 |

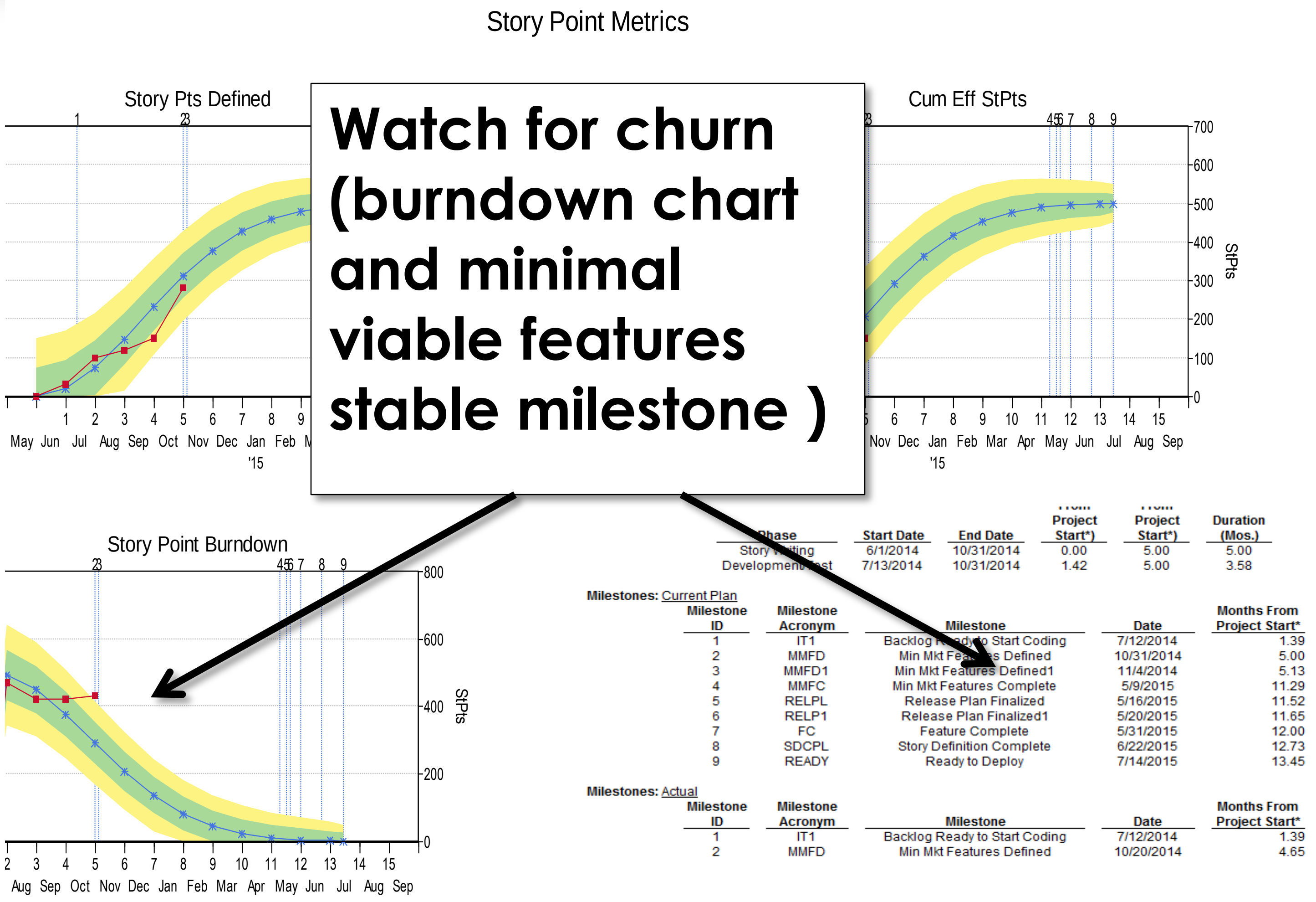
**Compare Stories Defined with Stories Developed**

| Milestone | Date | Months From Project Start* |
|---|---|---|
| Backlog Ready to Start Coding | 7/12/2014 | 1.39 |
| Min Mkt Features Defined | 10/31/2014 | 5.00 |
| Min Mkt Features Defined1 | 11/4/2014 | 5.13 |
| Min Mkt Features Complete | 5/9/2015 | 11.29 |
| Release Plan Finalized | 5/16/2015 | 11.52 |
| Release Plan Finalized1 | 5/20/2015 | 11.65 |
| Feature Complete | 5/31/2015 | 12.00 |
| Story Definition Complete | 6/22/2015 | 12.73 |
| Ready to Deploy | 7/14/2015 | 13.45 |

| ID | 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® The Intelligence behind Successful Software Projects

# Monitoring story definition



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

| Phase | Start Date | End Date | From<br>Project<br>Start*) | From<br>Project<br>Start*) | Duration<br>(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

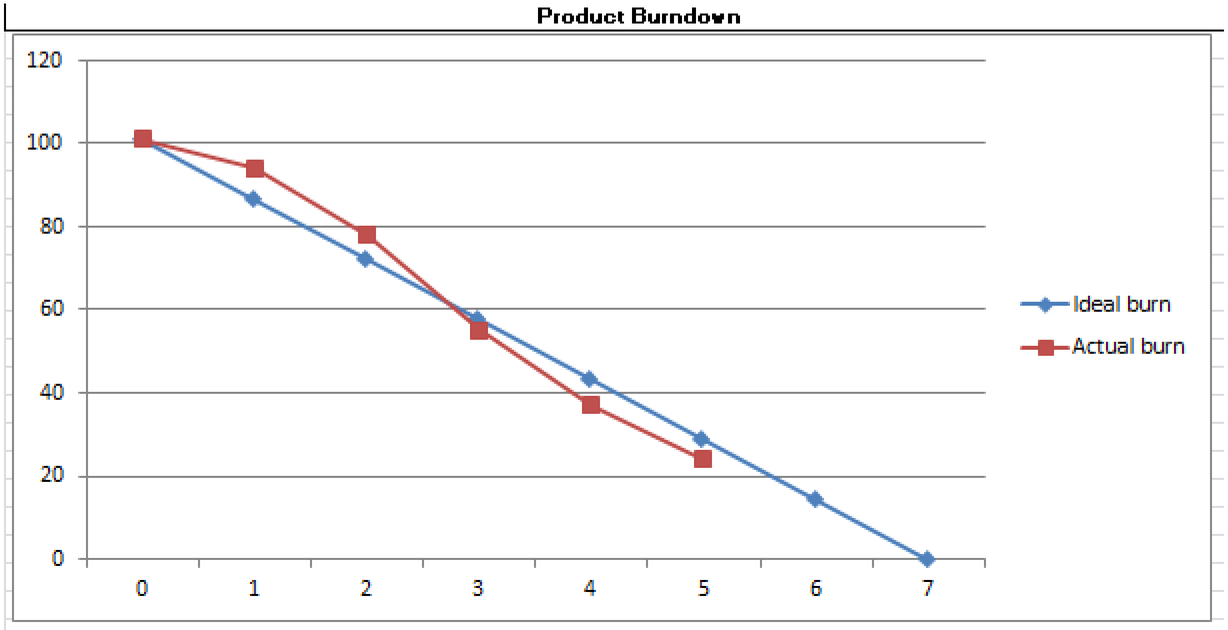
| Milestone ID | Milestone Acronym | Milestone | Date | Months From Project Start* |
|---|---|---|---|---|
| 1 | IT1 | Backlog Ready to Start Coding | 7/12/2014 | 1.39 |
| 2 | MMFD | Min Mkt Features Defined | 10/31/2014 | 5.00 |
| 3 | MMFD1 | Min Mkt Features Defined1 | 11/4/2014 | 5.13 |
| 4 | MMFC | 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 |

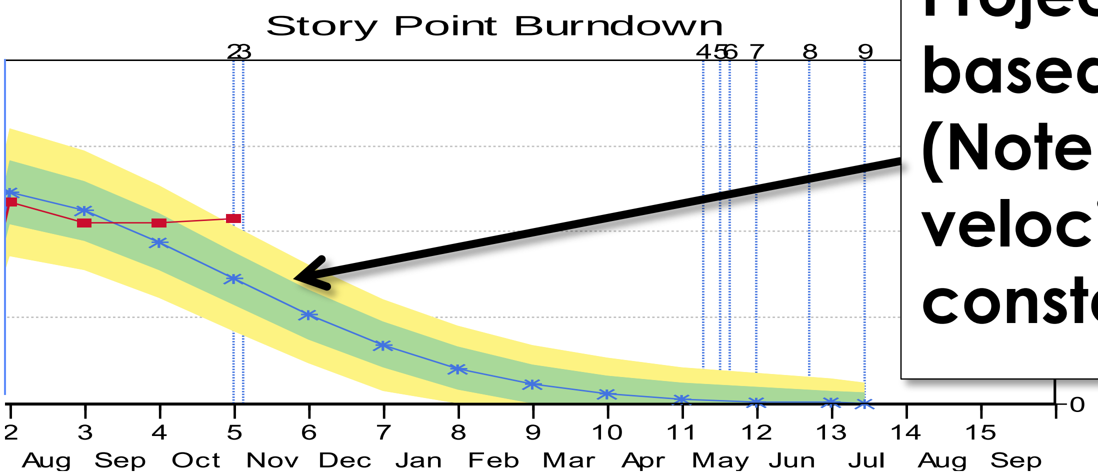
QSM® The Intelligence behind Successful Software Projects

# 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!



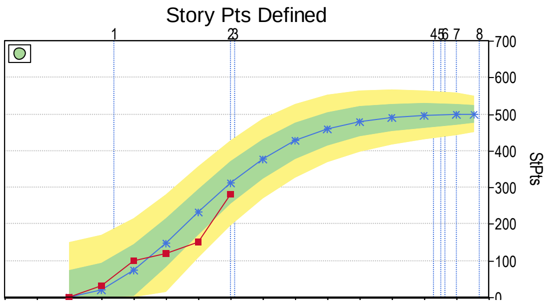
Projected values based on plan (Note the shape—velocity is NOT constant)

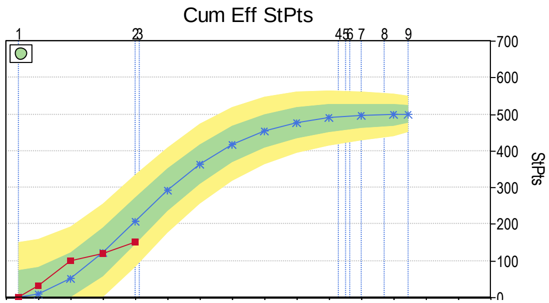
# Burndown vs. Story Points Developed

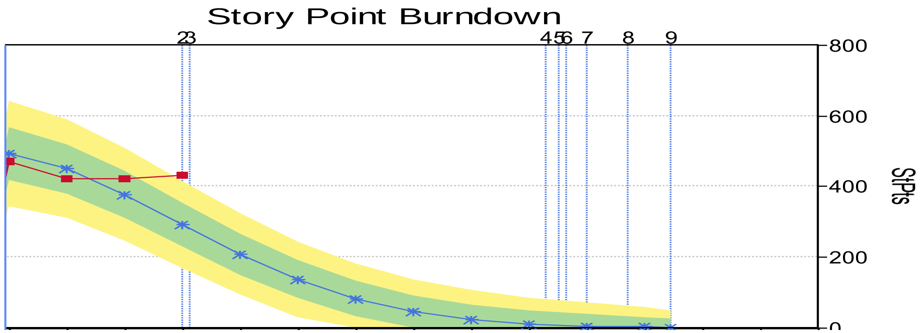
- Why isn’t the Burndown Chart just the Story Points Developed “upside down”?
  - Equivalently, is this the right equation?
    - Backlog remaining = Plan – Story Points Developed?
- It isn’t, because the Burndown Chart includes changes to the backlog
  - The equation is:
    - Backlog remaining = Plan – 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







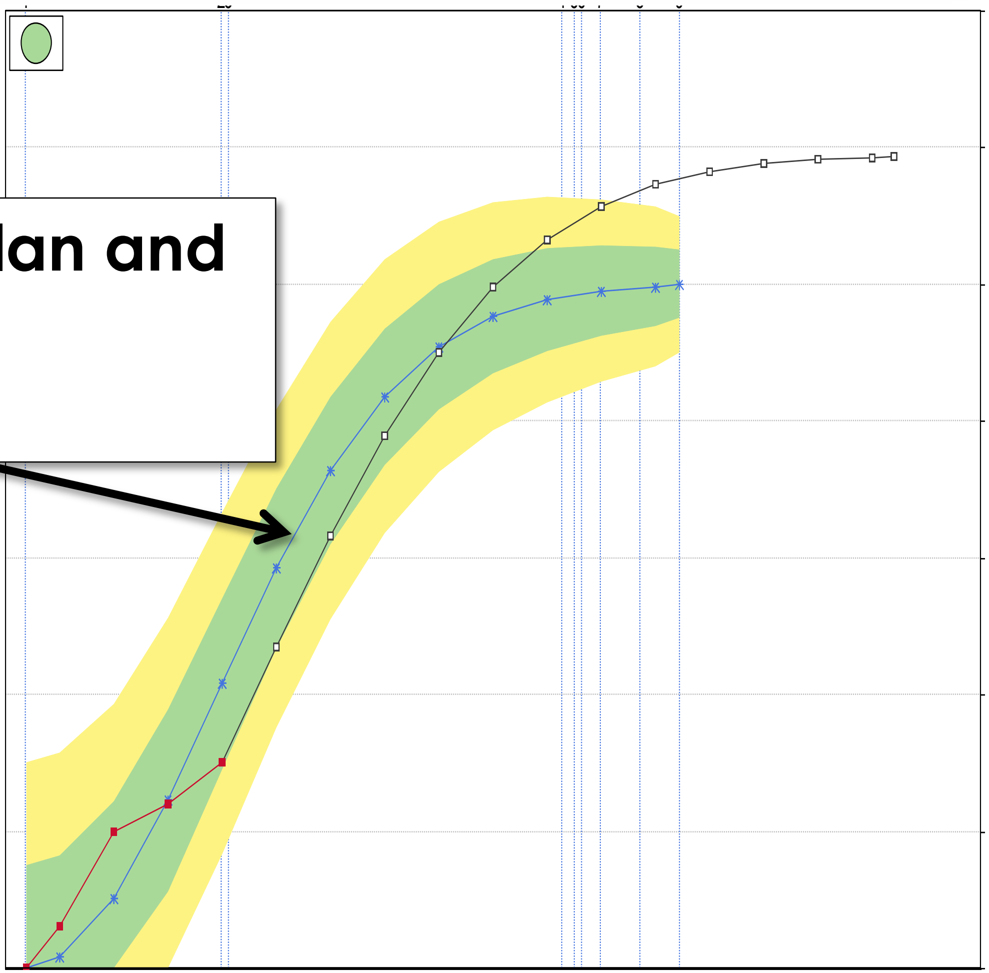
# When metrics are “out of control”

- 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

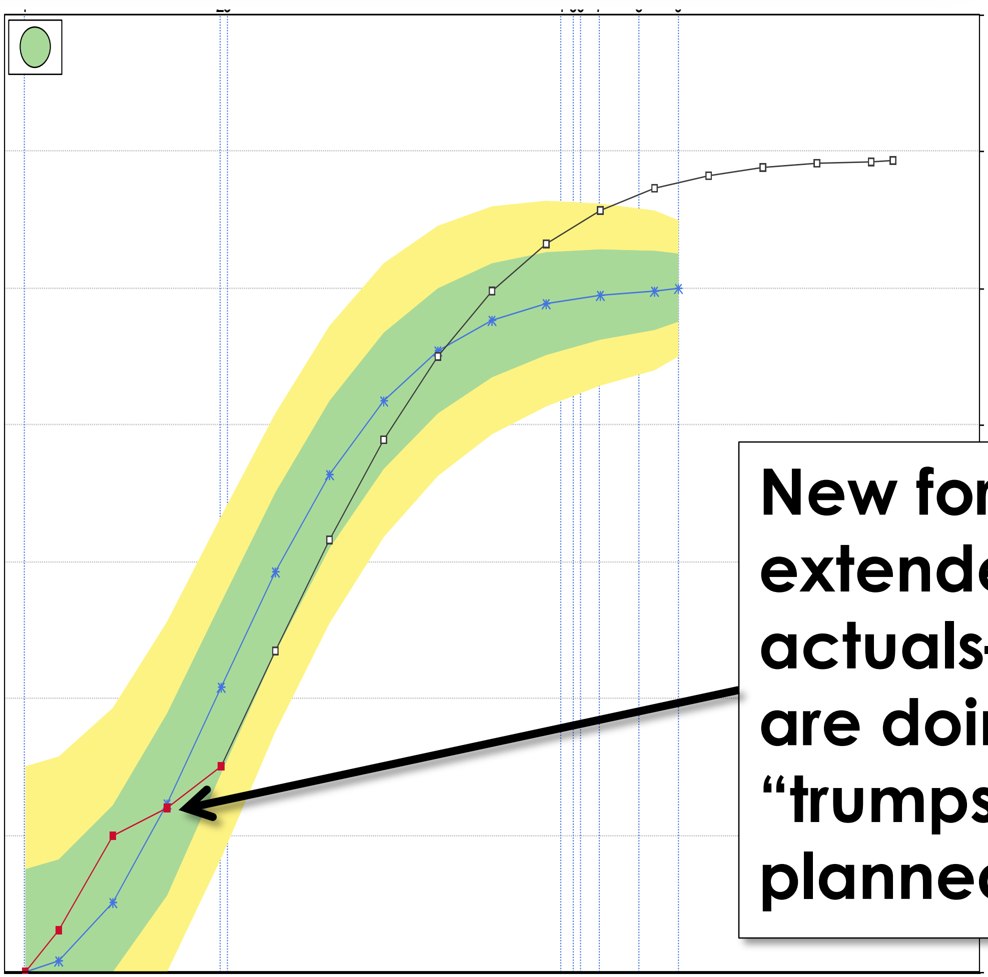
- When actuals are consistently out of the “green zone” in the same direction and milestones are late, it’s time to reforecast.
- Use actuals to forecast to completion
  - Different metrics may give different answers
  - Weigh multiple metrics
  - Include plan changes

# Anatomy of a forecast



Original plan and duration

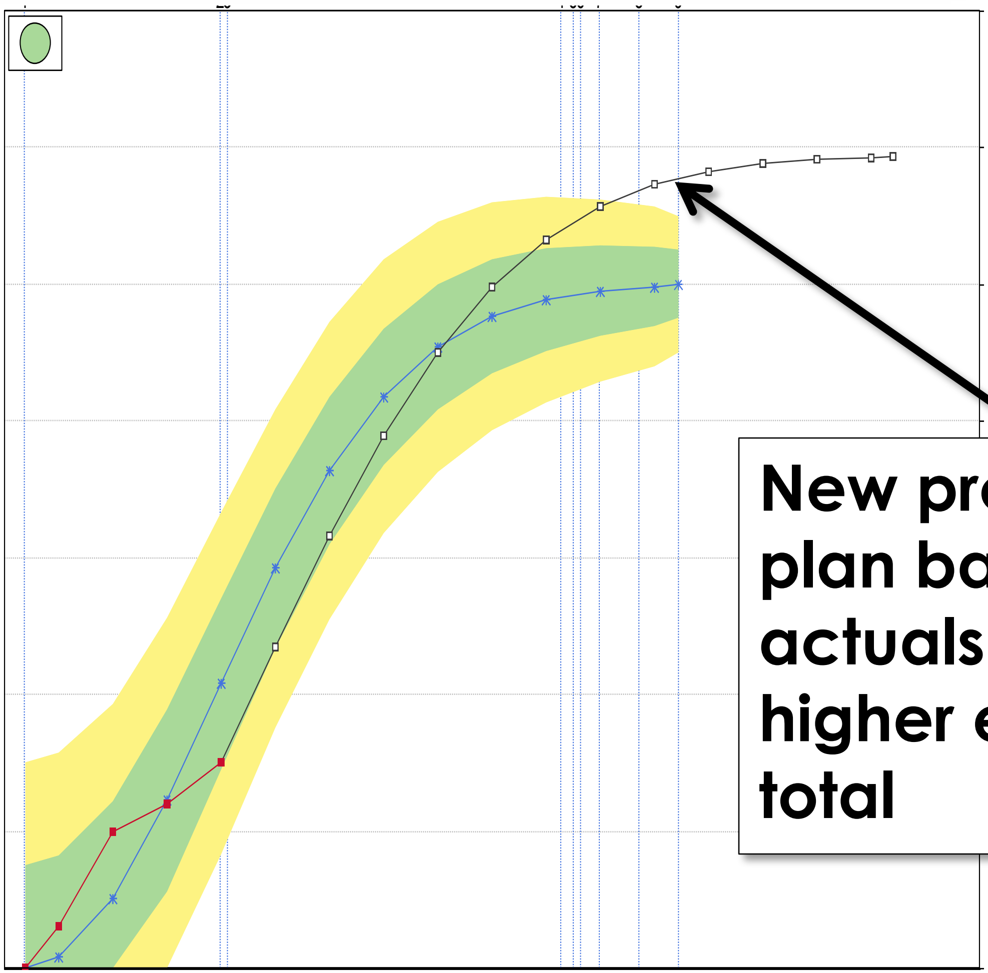
# Anatomy of a forecast



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

QSM® The Intelligence behind Successful Software Projects

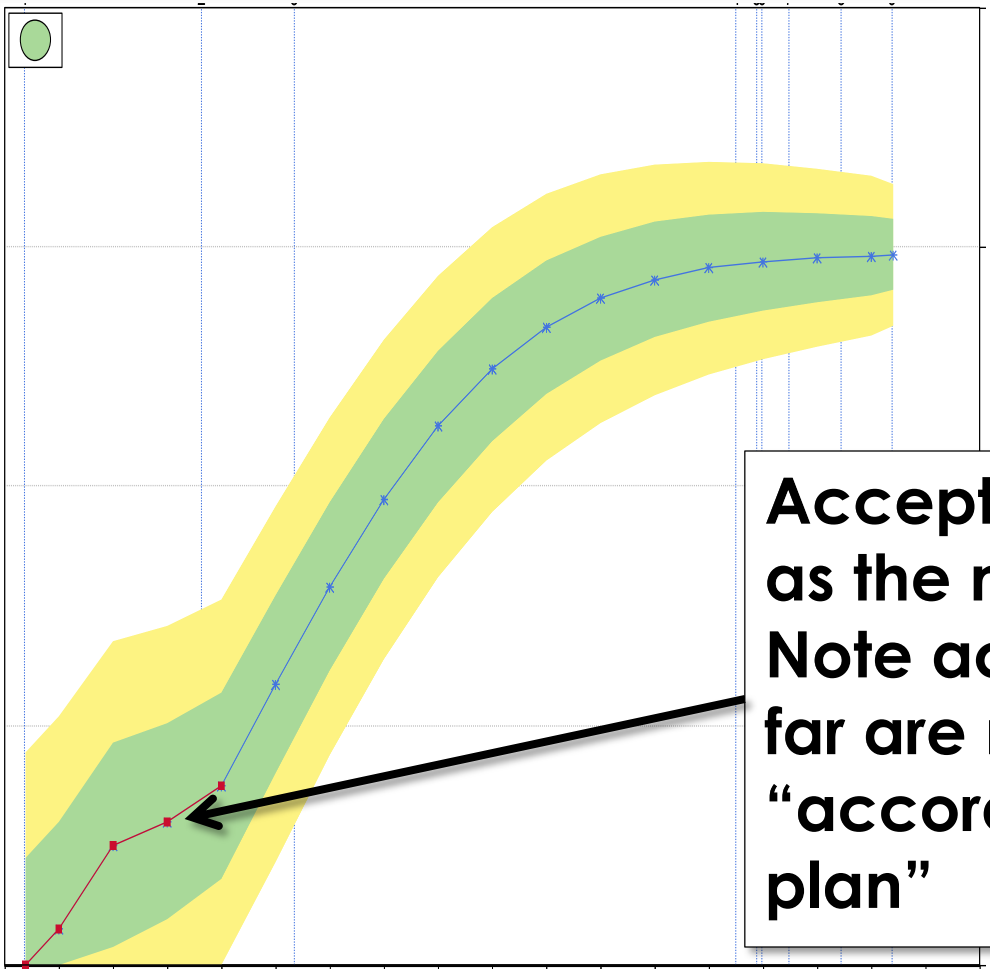
# Anatomy of a forecast



New projected plan based on actuals and higher expected total

QSM® The Intelligence behind Successful Software Projects

# Anatomy of a forecast



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

QSM® The Intelligence behind Successful Software Projects

# 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

# Questions?

> ***"I never learn anything talking.  I only learn when I ask questions."***
>
> *Lou Holtz*

**Contact us at info@qsm.com**

QSM® The Intelligence behind Successful Software Projects