# Agile Estimation: Beyond the Myths Part 1

**Andy Berner**

**Quantitative Software Management, Inc.**

QSM® The Intelligence behind Successful Software Projects

# Agenda

- 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

# Preview of Agenda for Part 2 Next Month

- **Further configuration for estimation:**
  - **Milestones**
  - **Productivity and Effort**
- **Project Control**
- **Data Collection**
- **Further Research**

# 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
- 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

QSM® The Intelligence behind Successful Software Projects

# 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

# 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*

# 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

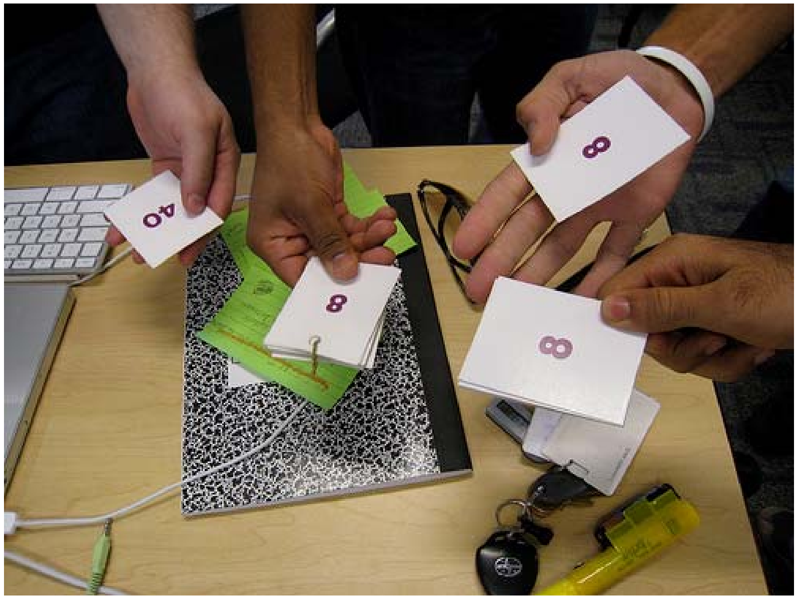
- At each sprint, we have to think about:
  - What detailed stories should we develop next?
  - Based on feedback, do we have a better understanding of the content or priorities?
  - What did we learn in the last couple of weeks, and how can we apply it?

… ***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

# “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**

QSM® The Intelligence behind Successful Software Projects

# “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

# 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

# People need help for longer term estimates

- People try to extrapolate from what they know they

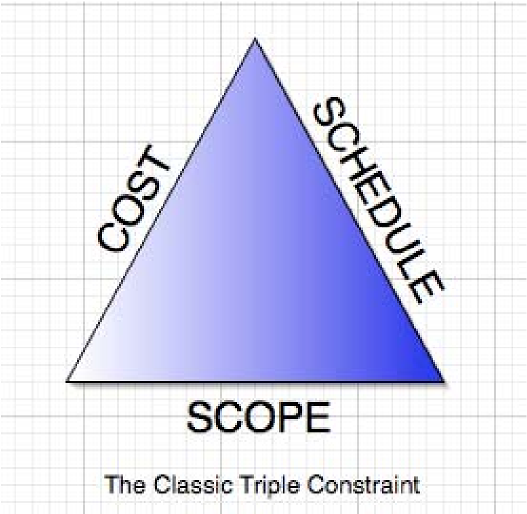
*The SLIM Suite of tools uses proven statistical techniques based on the nonlinearities of software development , using your organization’s historical data or industry-wide historical data to offer credible answers to the medium and long term questions.*

- nd is
- n history

The Intelligence behind Successful Software Projects

# 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



The Classic Triple Constraint

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

- **No more so in agile projects than any other!**

***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!***

QSM The Intelligence behind Successful Software Projects

# 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 enough features, users won't use it: must be feature rich enough to consume
  - "Value is in the eye of the beholder"

QSM® The Intelligence behind Successful Software Projects

# 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”

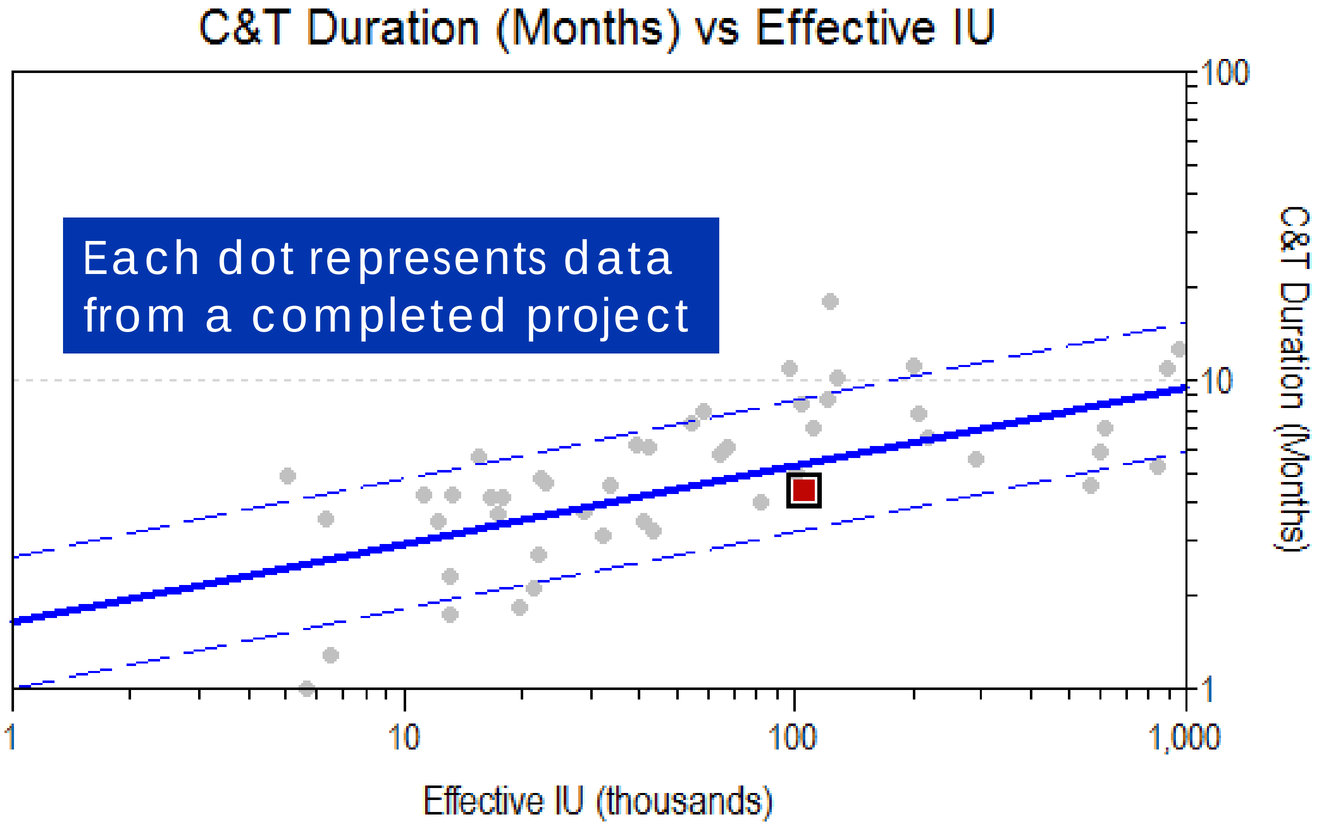
*Ladies and gentlemen, the new release of our navigation software worked flawlessly. We will be able to land in a few months when the next release comes out.*

QSM® The Intelligence behind Successful Software Projects

# Estimating a project to develop a release

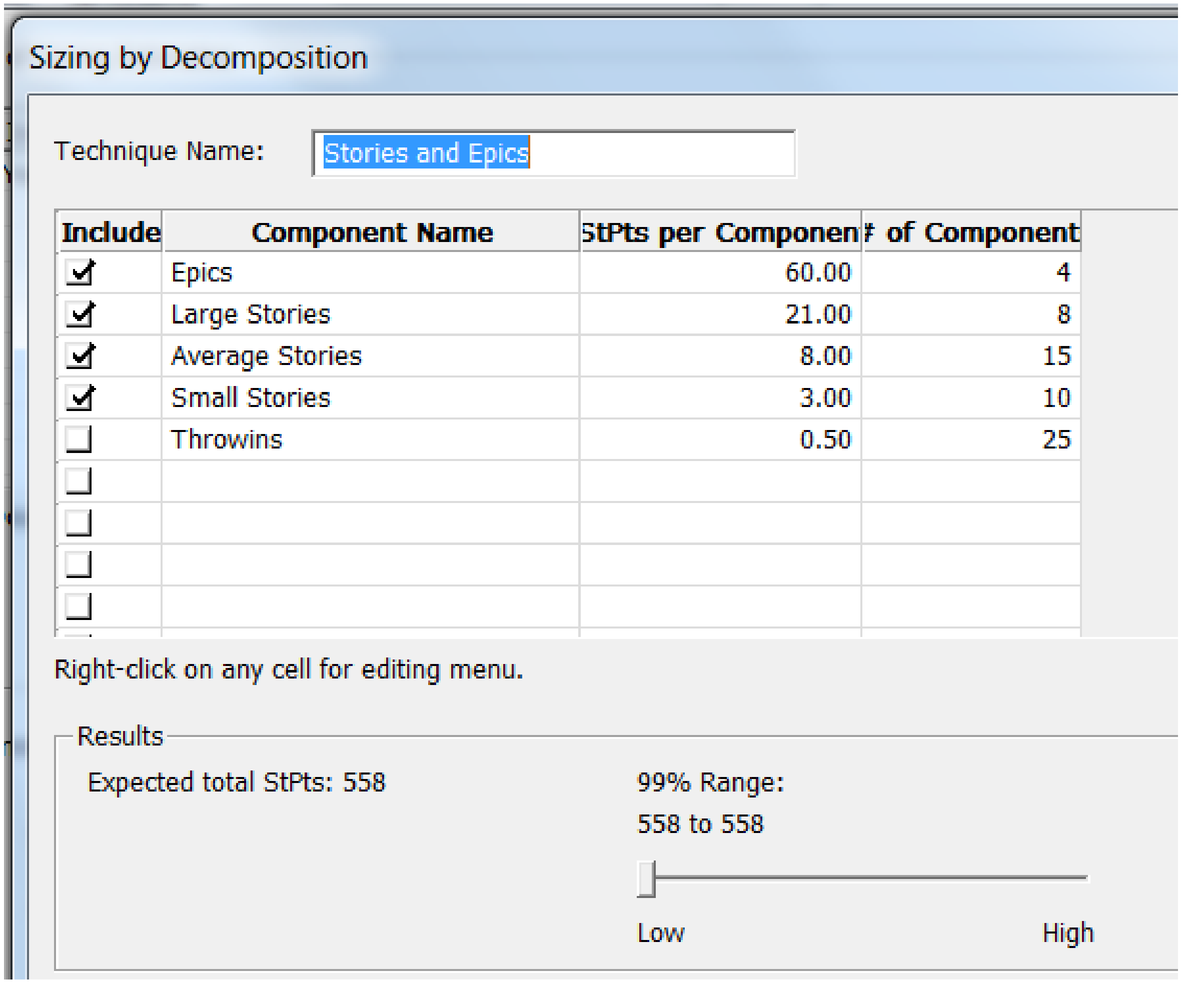
- Key inputs
  - Historical Data
  - Size of functionality needed for goals
  - Shape of the work/methodology
  - Productivity expectations
- Outputs
  - Tradeoffs between schedule and effort
  - Risk
  - Expected dates for key milestones

# Input: Historical Data



Each dot represents data from a completed project

# Input: Size of the functionality needed for goals



Sizing by Decomposition

Technique Name: Stories and Epics

| Include | Component Name | StPts per Componen | # of Component |
|---|---|---|---|
| ☑ | Epics | 60.00 | 4 |
| ☑ | Large Stories | 21.00 | 8 |
| ☑ | Average Stories | 8.00 | 15 |
| ☑ | Small Stories | 3.00 | 10 |
| ☐ | Throwins | 0.50 | 25 |
| ☐ | | | |
| ☐ | | | |
| ☐ | | | |
| ☐ | | | |

Right-click on any cell for editing menu.

Results

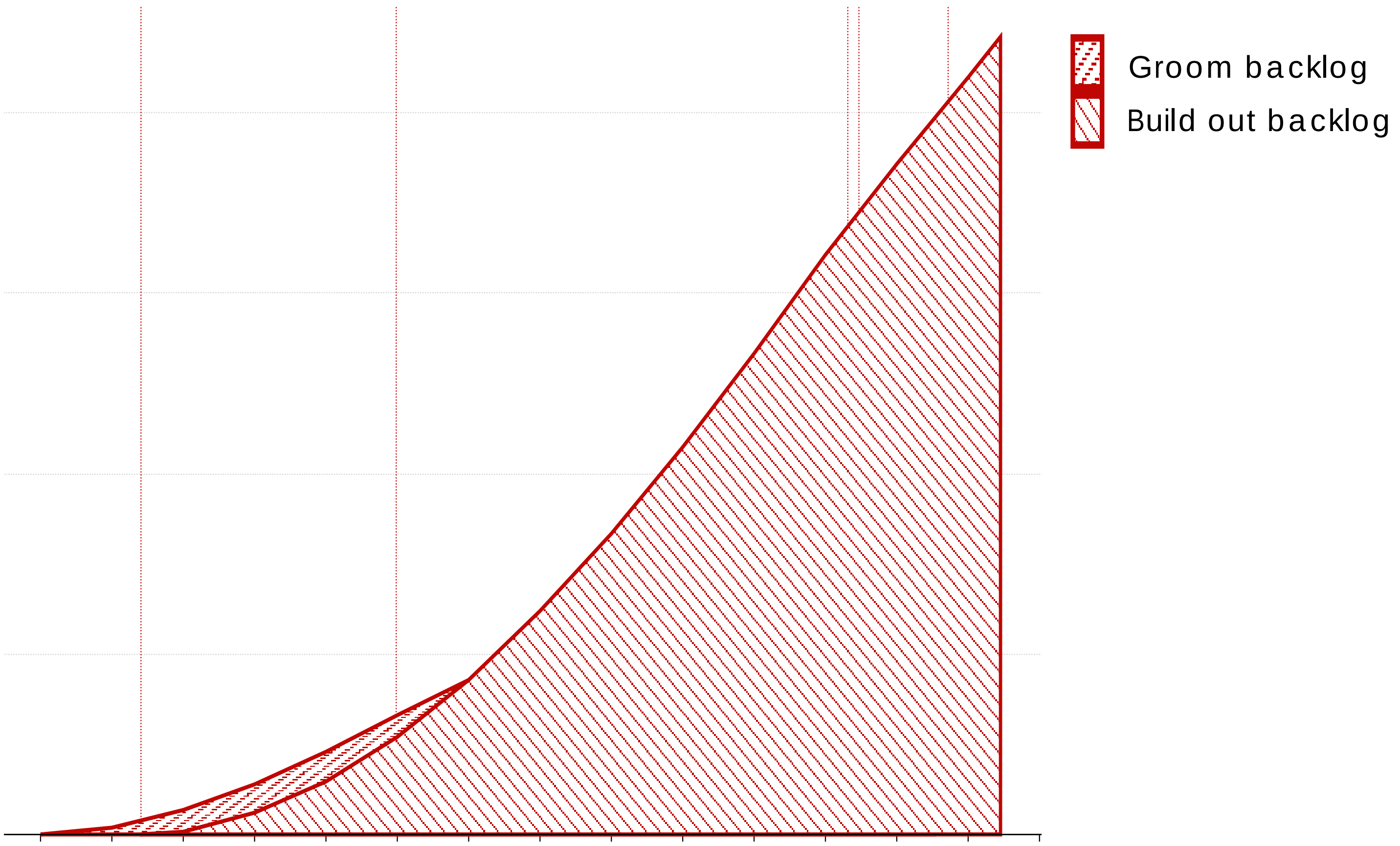
Expected total StPts: 558

99% Range:

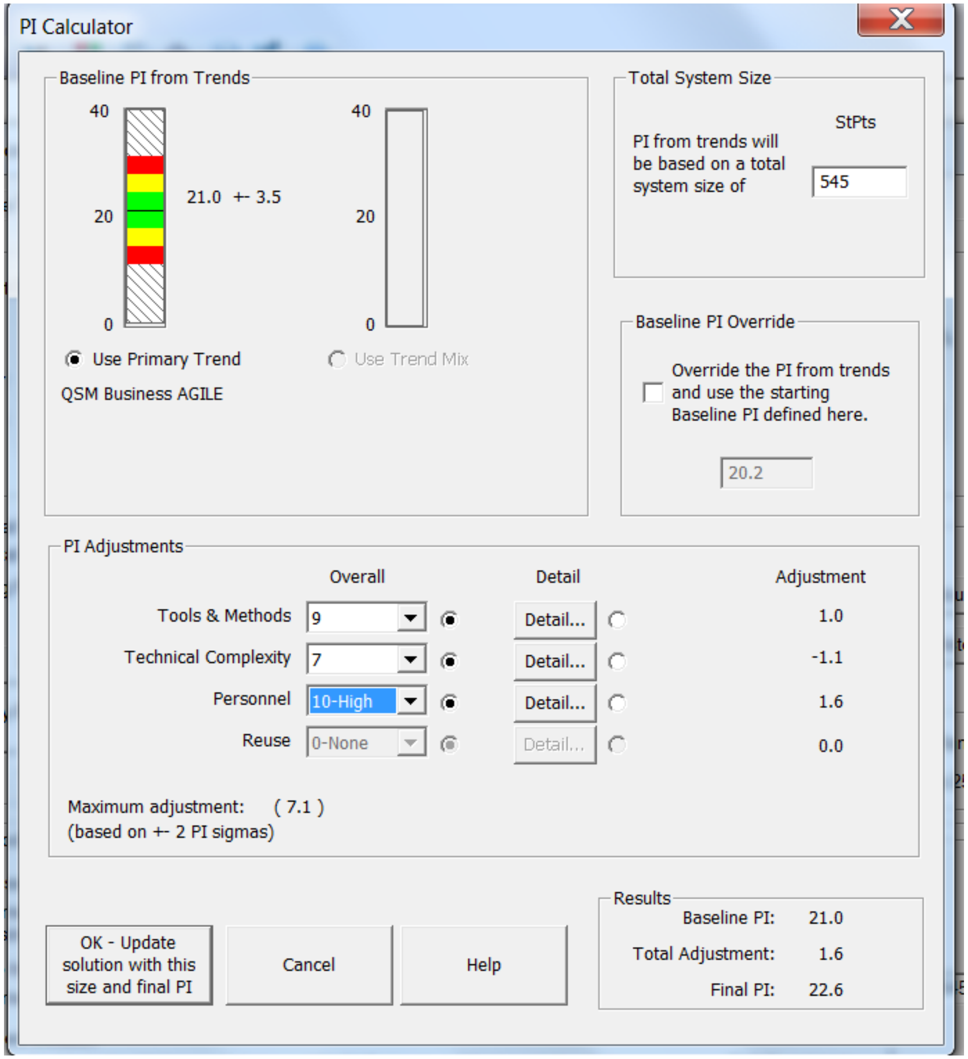
558 to 558

The Intelligence behind Successful Software Projects

# Input: Shape of the work/methodology



# Input: Productivity Expectations



PI Calculator

**Baseline PI from Trends**

21.0 +- 3.5

● Use Primary Trend ○ Use Trend Mix

QSM Business AGILE

**Total System Size**

PI from trends will be based on a total system size of

StPts: 545

**Baseline PI Override**

□ Override the PI from trends and use the starting Baseline PI defined here.

20.2

**PI Adjustments**

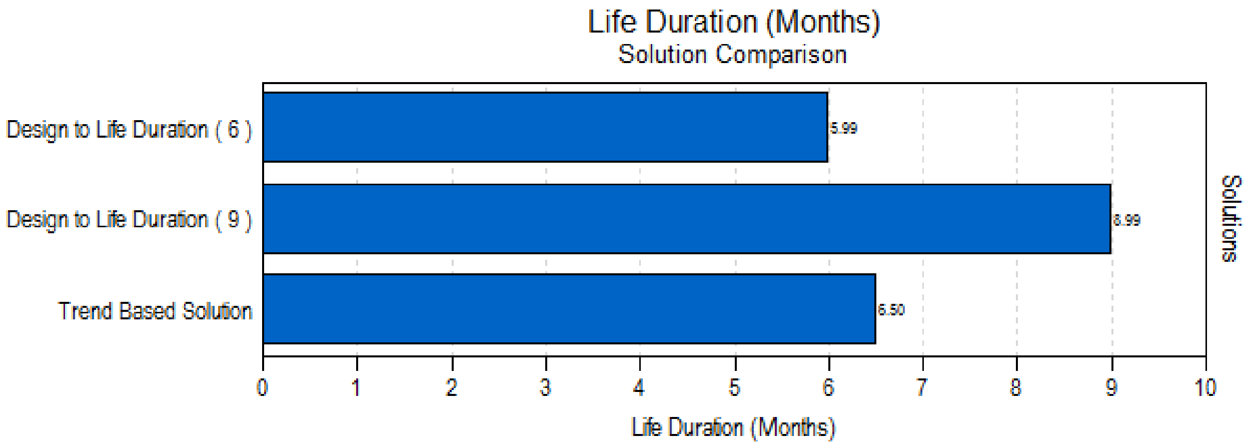
| | Overall | | Detail | | Adjustment |
|---|---|---|---|---|---|
| Tools & Methods | 9 | ● | Detail... | ○ | 1.0 |
| Technical Complexity | 7 | ● | Detail... | ○ | -1.1 |
| Personnel | 10-High | ● | Detail... | ○ | 1.6 |
| Reuse | 0-None | ● | Detail... | ○ | 0.0 |

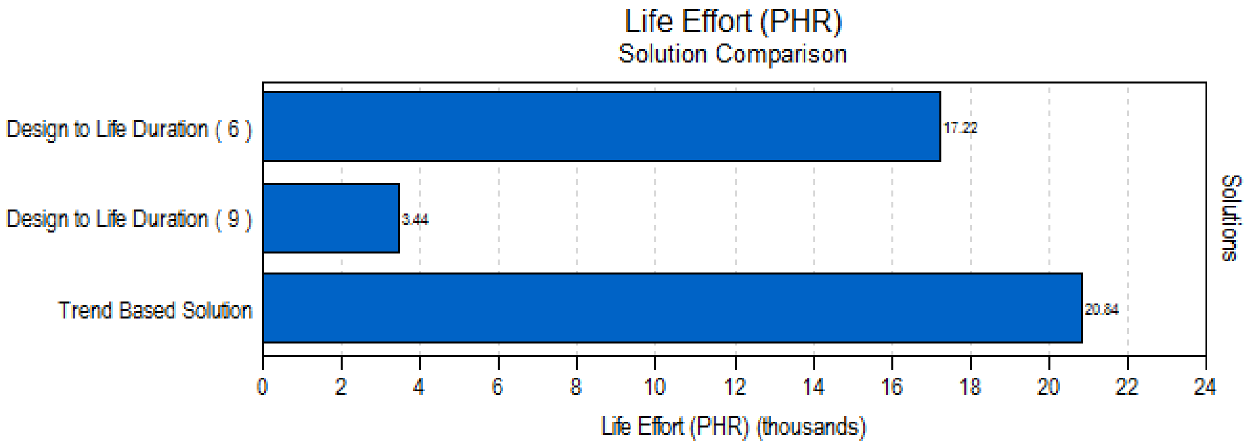
Maximum adjustment: ( 7.1 )
(based on +- 2 PI sigmas)

**Results**

| | |
|---|---|
| Baseline PI: | 21.0 |
| Total Adjustment: | 1.6 |
| Final PI: | 22.6 |

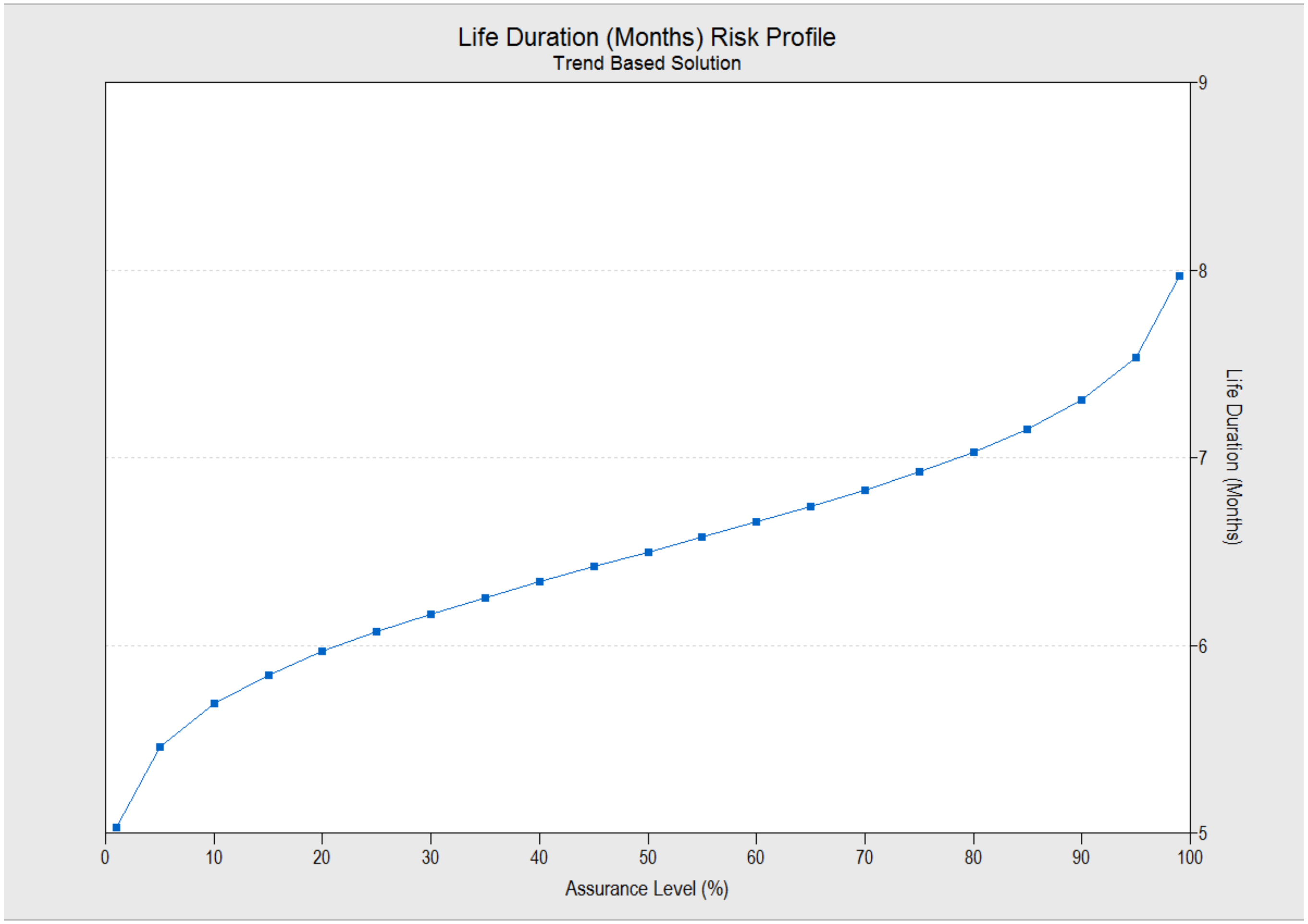
# Output: Tradeoffs between schedule and effort





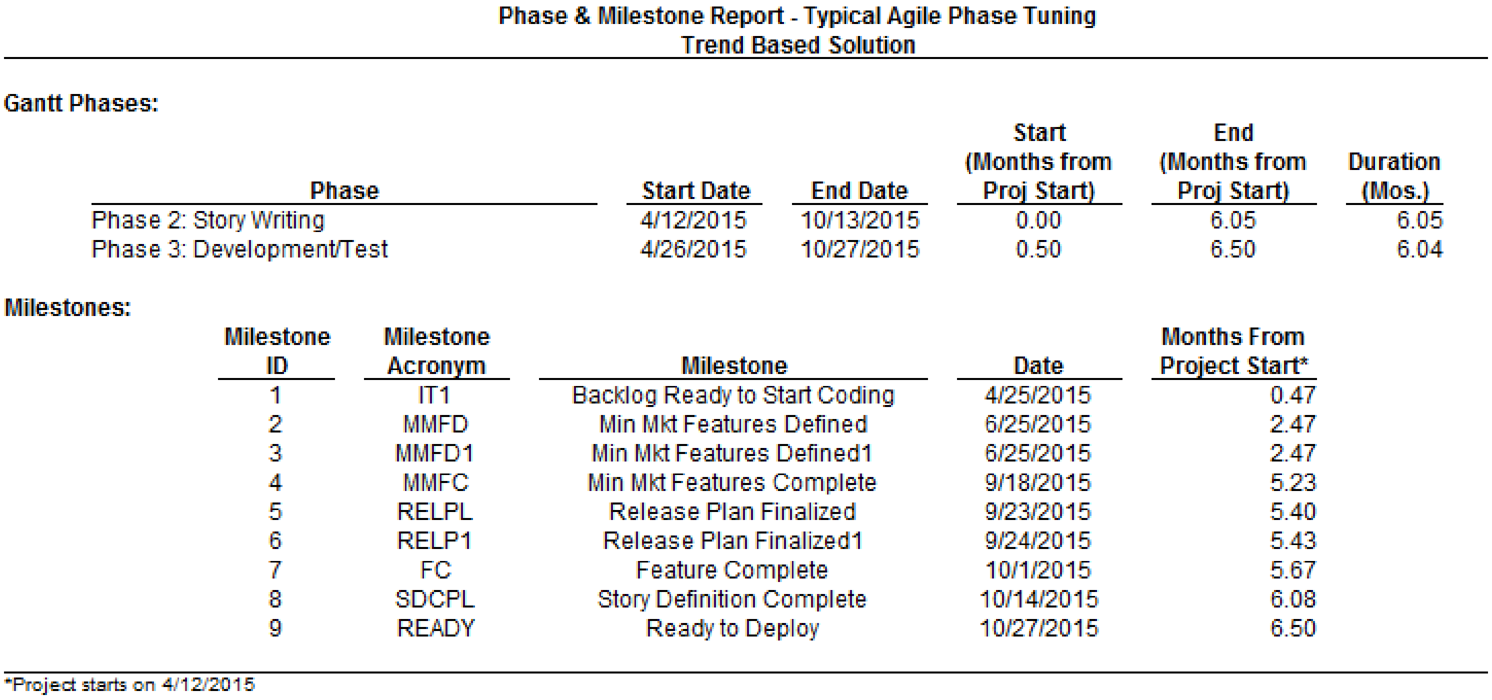
# Output: Risk

| Assurance Level (%) | Life Duration (Months) |
|---|---|
| 1 | 5.03 |
| 5 | 5.46 |
| 10 | 5.69 |
| 15 | 5.85 |
| 20 | 5.97 |
| 25 | 6.07 |
| 30 | 6.17 |
| 35 | 6.26 |
| 40 | 6.34 |
| 45 | 6.42 |
| 50 | 6.50 |
| 55 | 6.58 |
| 60 | 6.66 |
| 65 | 6.74 |
| 70 | 6.83 |
| 75 | 6.93 |
| 80 | 7.03 |
| 85 | 7.15 |
| 90 | 7.31 |
| 95 | 7.54 |
| 99 | 7.97 |



QSM® The Intelligence behind Successful Software Projects

# Output: Expected Dates for Key Milestones



Phase & Milestone Report - Typical Agile Phase Tuning
Trend Based Solution

**Gantt Phases:**

| Phase | Start Date | End Date | Start (Months from Proj Start) | End (Months from Proj Start) | Duration (Mos.) |
|---|---|---|---|---|---|
| Phase 2: Story Writing | 4/12/2015 | 10/13/2015 | 0.00 | 6.05 | 6.05 |
| Phase 3: Development/Test | 4/26/2015 | 10/27/2015 | 0.50 | 6.50 | 6.04 |

**Milestones:**

| Milestone ID | Milestone Acronym | Milestone | Date | Months From Project Start* |
|---|---|---|---|---|
| 1 | IT1 | Backlog Ready to Start Coding | 4/25/2015 | 0.47 |
| 2 | MMFD | Min Mkt Features Defined | 6/25/2015 | 2.47 |
| 3 | MMFD1 | Min Mkt Features Defined1 | 6/25/2015 | 2.47 |
| 4 | MMFC | Min Mkt Features Complete | 9/18/2015 | 5.23 |
| 5 | RELPL | Release Plan Finalized | 9/23/2015 | 5.40 |
| 6 | RELP1 | Release Plan Finalized1 | 9/24/2015 | 5.43 |
| 7 | FC | Feature Complete | 10/1/2015 | 5.67 |
| 8 | SDCPL | Story Definition Complete | 10/14/2015 | 6.08 |
| 9 | READY | Ready to Deploy | 10/27/2015 | 6.50 |

*Project starts on 4/12/2015

# 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
  - Statistical methods that understand the nonlinearities of software development
  - Organizational or industry-wide historical data

QSM® The Intelligence behind Successful Software Projects

# Sizing

***"Is it bigger than a breadbox?"***

*Steve Allen, on "What's My Line"*

QSM® The Intelligence behind Successful Software Projects

# 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® The Intelligence behind Successful Software Projects

# 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® The Intelligence behind Successful Software Projects

# 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

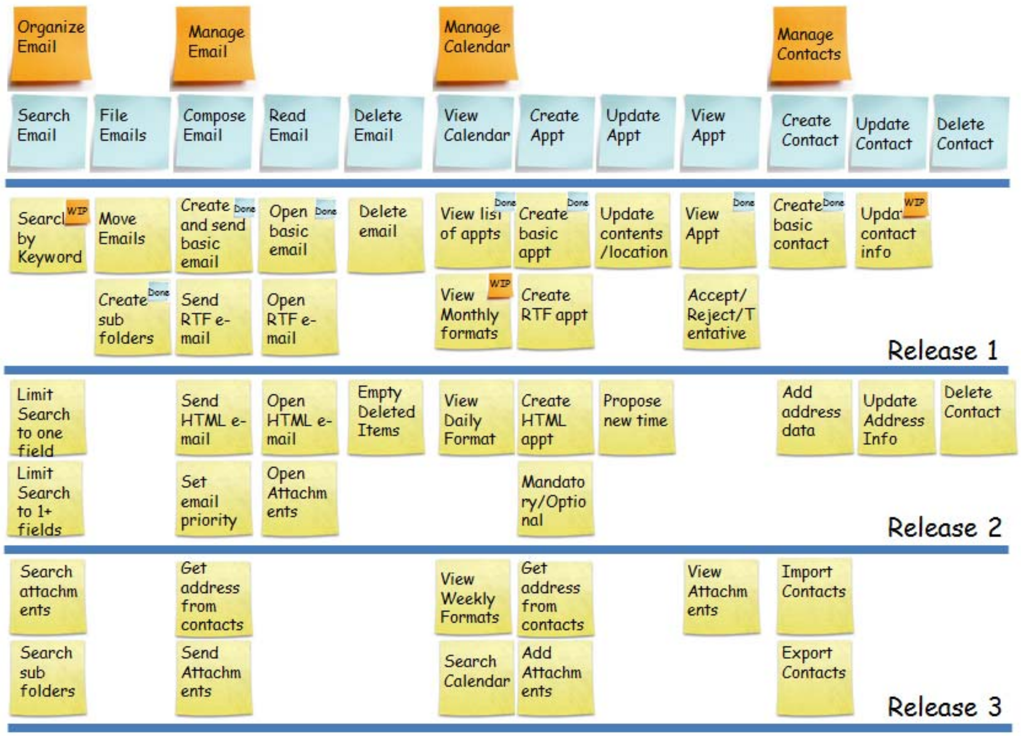
- Epics and stories, like any other way to express scope, is hierarchical! Each level … it.
- … t's

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.

  – User size bites may be at any level

# Ways of showing the hierarchy

- **User Story Maps (Jeff Patton)**



| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | | | | Release 1 |
| 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 | | | | | | Release 2 |
| 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 | | | Release 3 |

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

QSM® The Intelligence behind Successful Software Projects

# Ways of Showing the Hierarchy

- **Minimal Marketable Features**
  - **Mark Denne and Jane Cleland-Huang**

**nd**

- **ce**

***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.***

The Intelligence behind Successful Software Projects

# 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

- Duration depends on software size, effort expended, and productivity of the team.
- ne
- e

S 3 )

*We will discuss the issues of effort and productivity in the next webinar in this series.*

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

## 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?

# 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!**

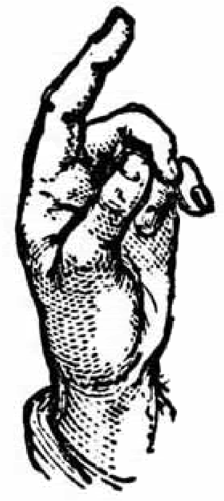
# Size and duration– a nonlinear relationship!

- 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?”

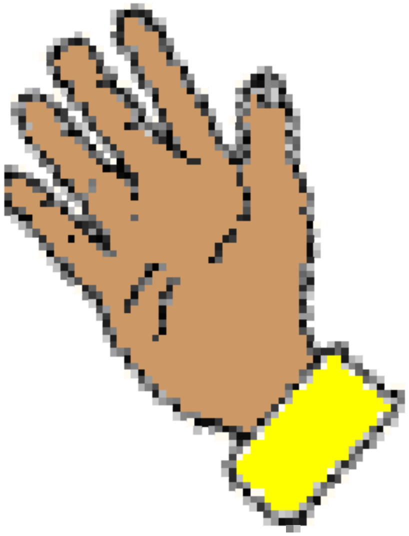


- The second kid says, “I don’t know”
- The first kid then replies….

QSM® The Intelligence behind Successful Software Projects

# An Old Kid’s Joke

- “I don’t know either, but here come five of them.



QSM® The Intelligence behind Successful Software Projects

# 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!***

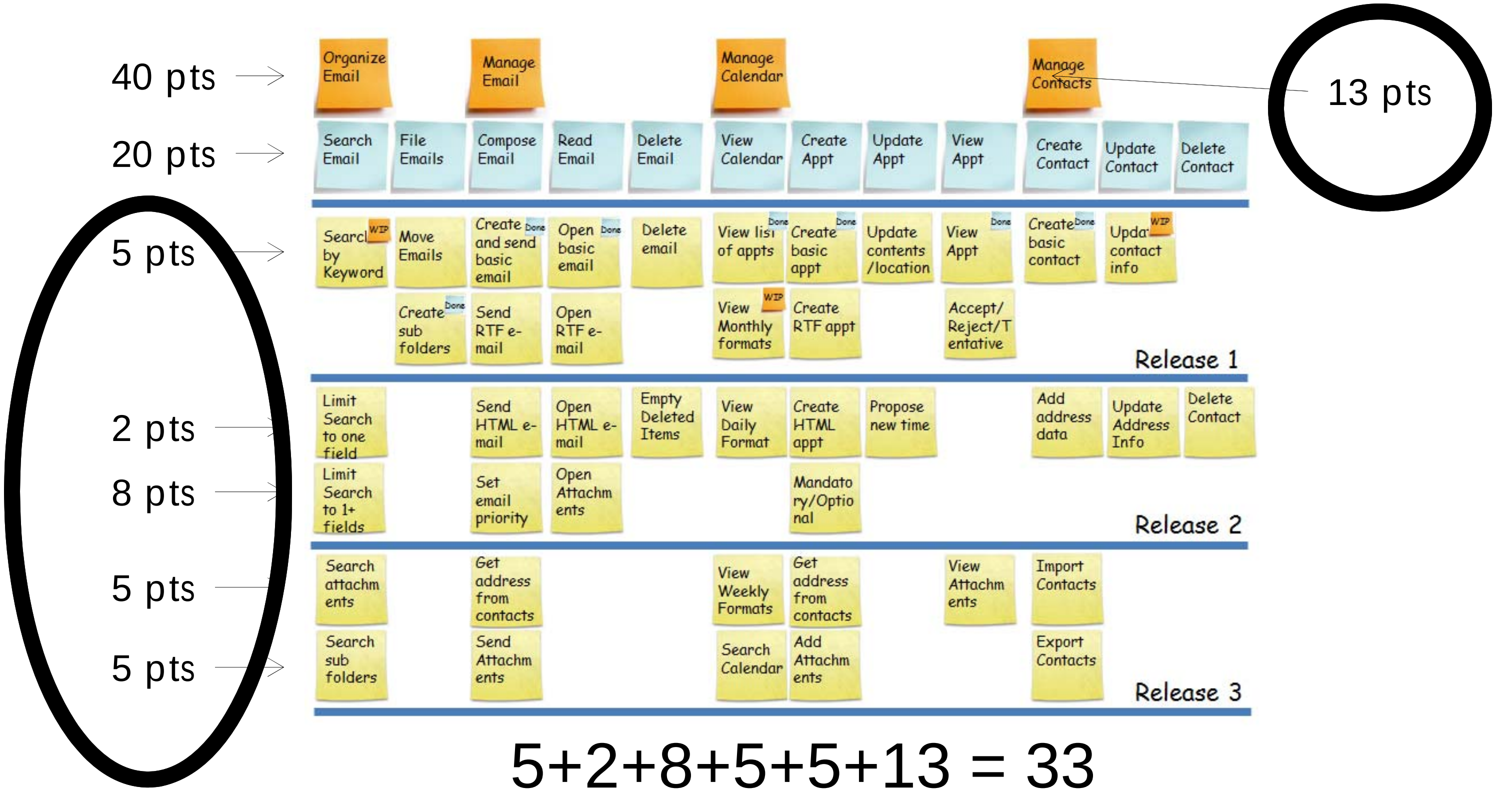
# Function Points and Story Points

- 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

QSM® The Intelligence behind Successful Software Projects

# 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**



40 pts

20 pts

5 pts

2 pts

8 pts

5 pts

5 pts

13 pts

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

Release 1

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

Release 2

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

Release 3

5+2+8+5+5+13 = 33

QSM® The Intelligence behind Successful Software Projects

# 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"

QSM® The Intelligence behind Successful Software Projects

# 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.**

QSM® The Intelligence behind Successful Software Projects

# Four Levels of Normalization

- **Within a company**
  - **All teams within the company rate stories consistently. Then**

**size**

-

**try**

**ht**

***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.***

QSM® The Intelligence behind Successful Software Projects

# 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

- **Industry standard**
  - **Can compare to other projects in your company using your own history**

***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.***

- **nd**
- 
- **on points completed at each iteration**

QSM® The Intelligence behind Successful Software Projects

# 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

# 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.

QSM® The Intelligence behind Successful Software Projects

# SLIM-Estimate Agile Template

- Trend based on industry wide agile projects
- Size unit and size calculator for estimating size
- Specific agile settings we will describe in this webinar series:
  - Milestones for project control
  - Settings for emergent requirements
- Customizable to your specific ways of working

# SLIM Agile Adjuster

- Specific for sprint based methods
- Extension to SLIM-Estimate, available right from the menu
- Choose sprint length
  - Computes estimated duration in terms of the chosen sprint length
- Adjusts key parameters based on sprint length and estimated number of sprints

QSM® The Intelligence behind Successful Software Projects

# Shape of the Work

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

*From "Company" by Stephen Sondheim*

QSM® The Intelligence behind Successful Software Projects

# When is a phase not a phase?

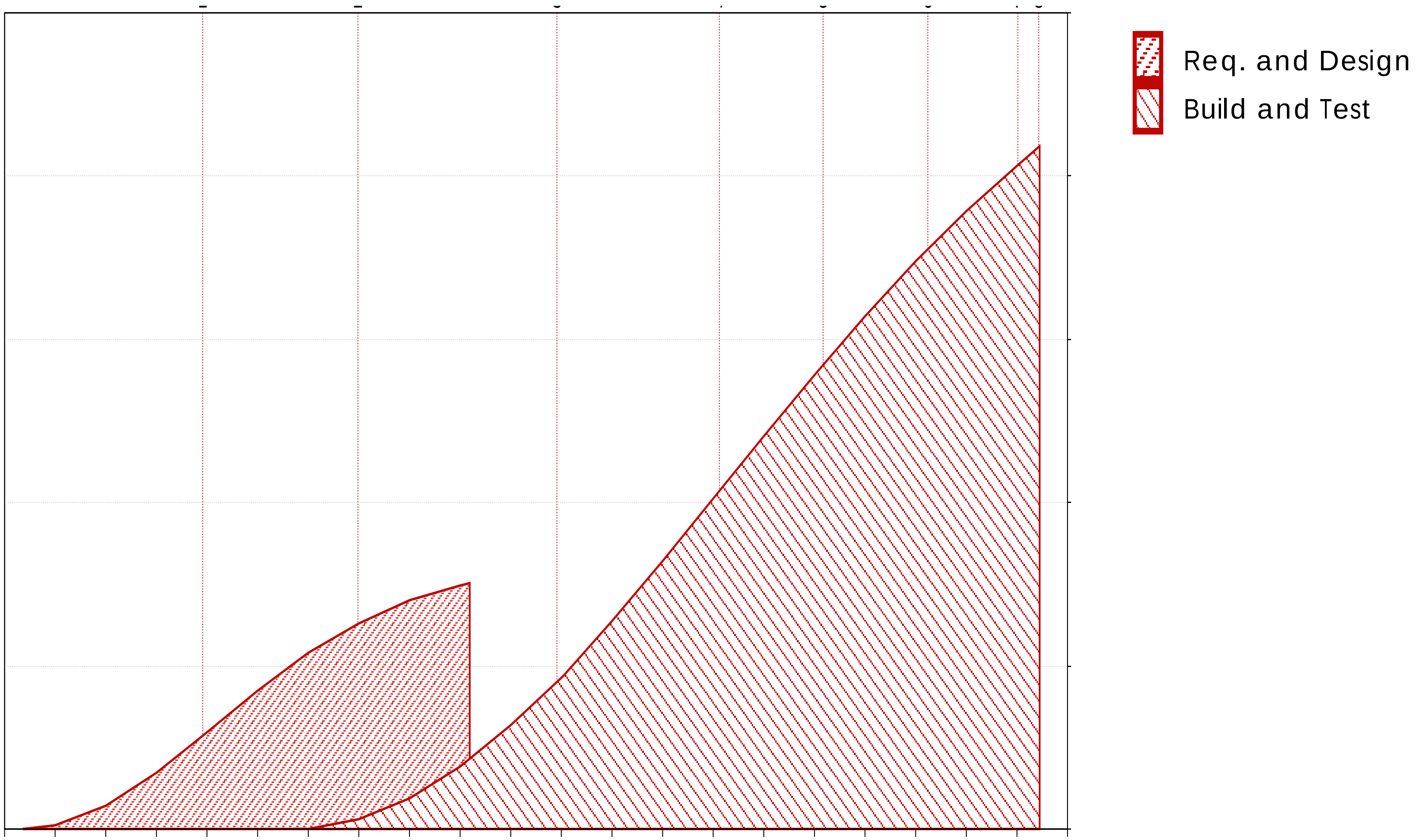
- Biggest difference in scheduling agile and waterfall projects:
  - In waterfall projects, different types of work (e.g. requirements definition, coding, testing) happen largely at different times
  - In agile projects, these different types of work happen much more concurrently
- In SLIM-Estimate, the term “phase” refers to different types of work
- “Phase tuning” adjusts the schedule and effort for these different types of work

QSM® The Intelligence behind Successful Software Projects

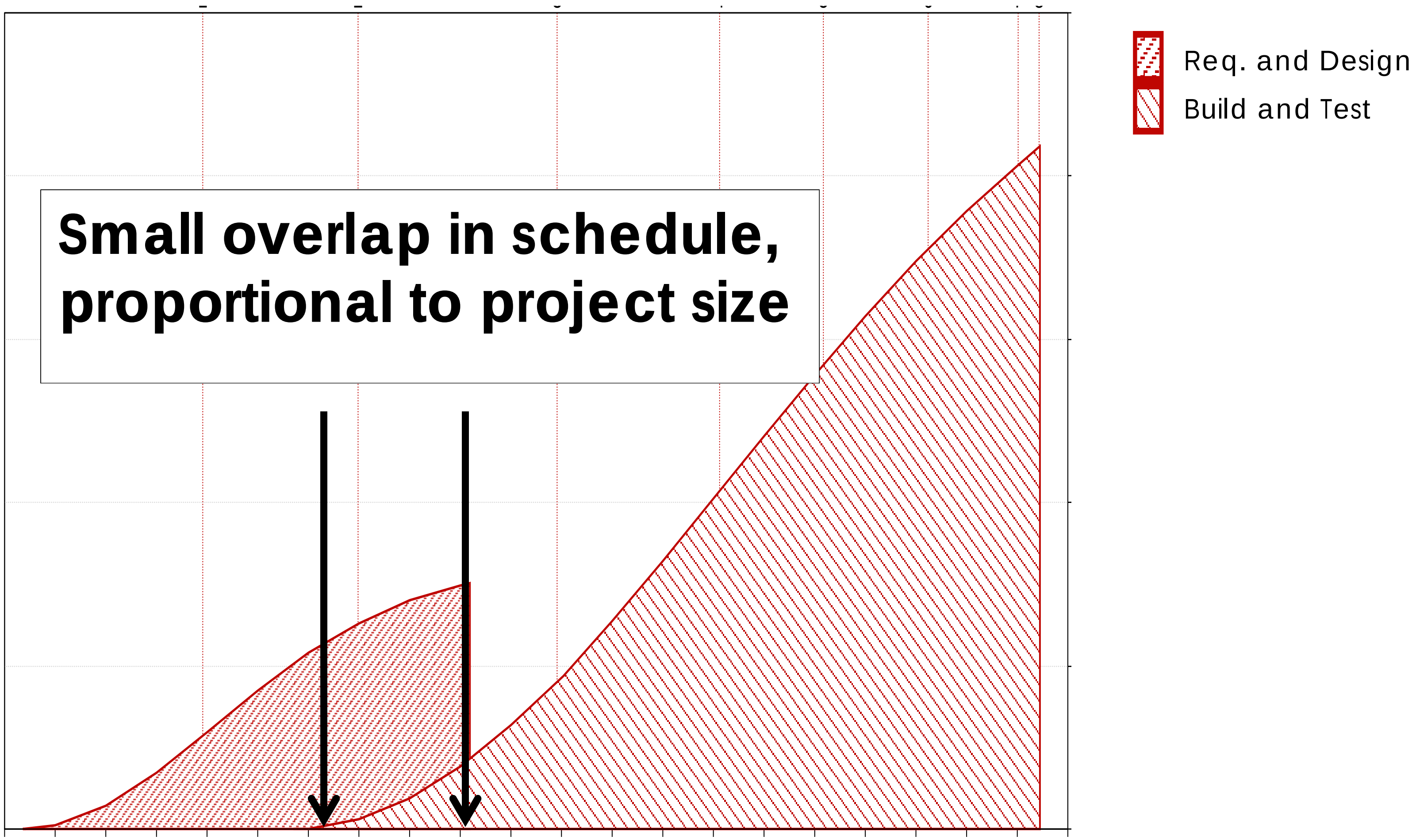
# Two key types of work

- 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 Phase Tuning

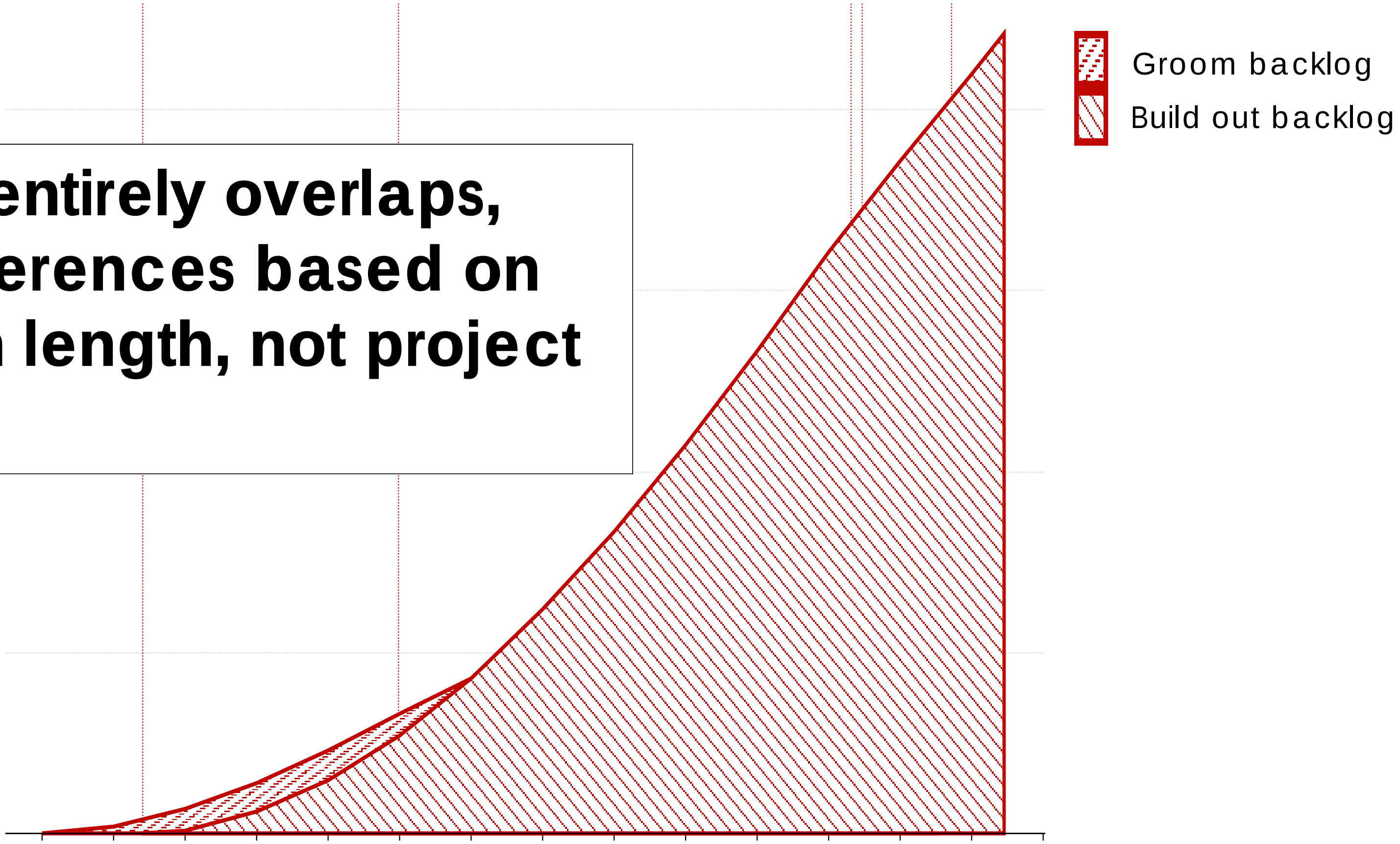


# "Big Upfront Requirements Phase"



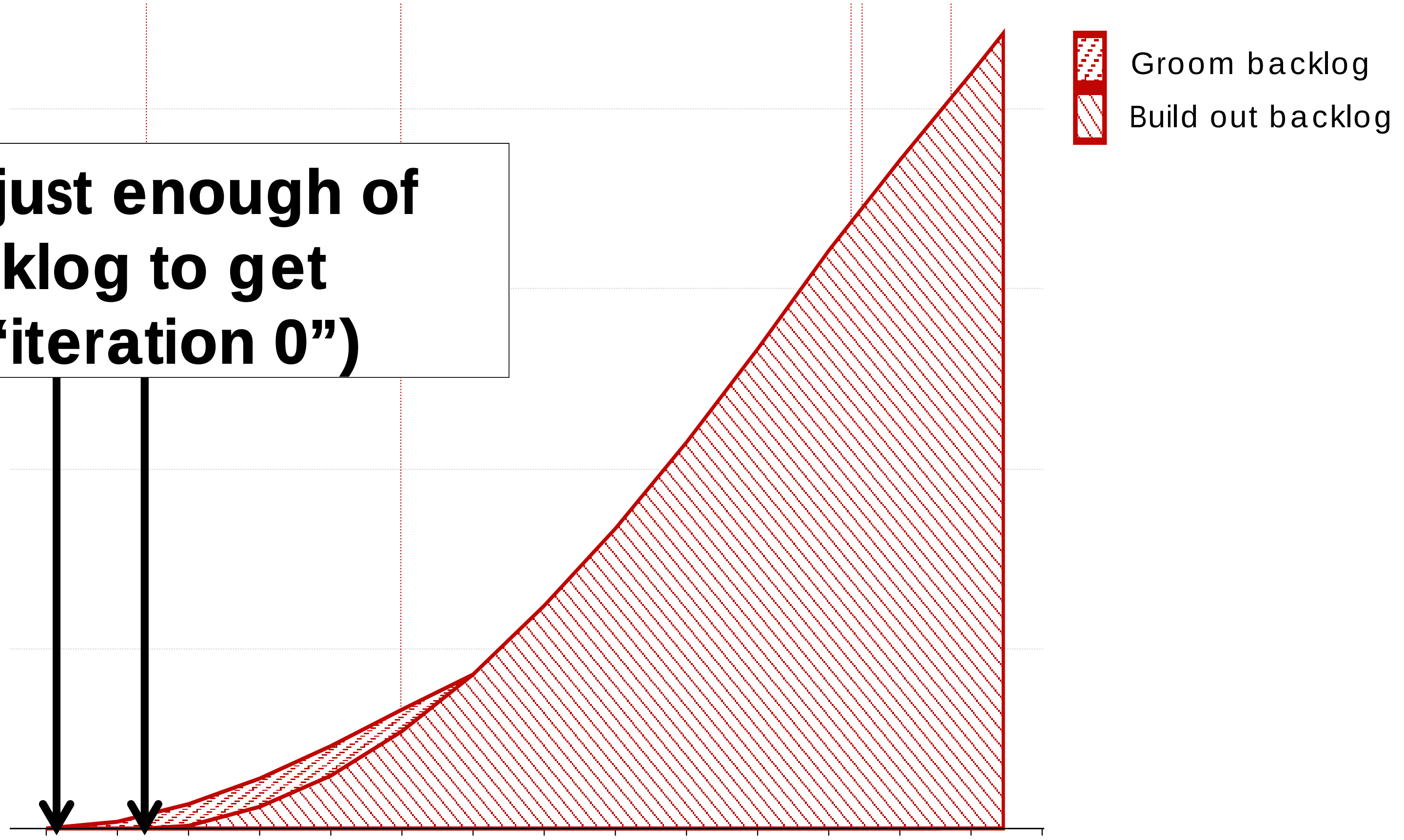
**Small overlap in schedule, proportional to project size**

# Typical Agile Phase Tuning



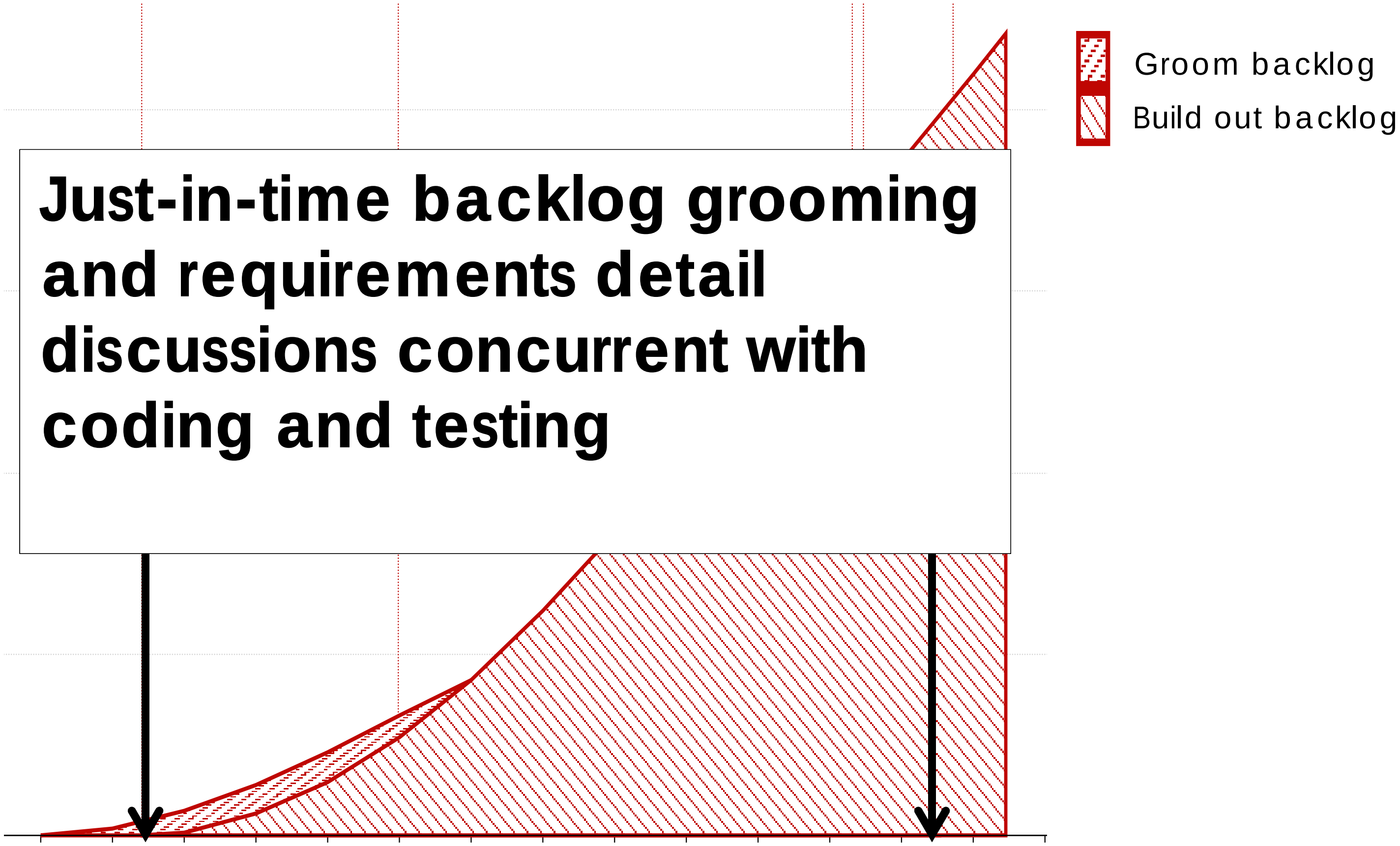
**Almost entirely overlaps, with differences based on iteration length, not project size**

# Emergent Requirements



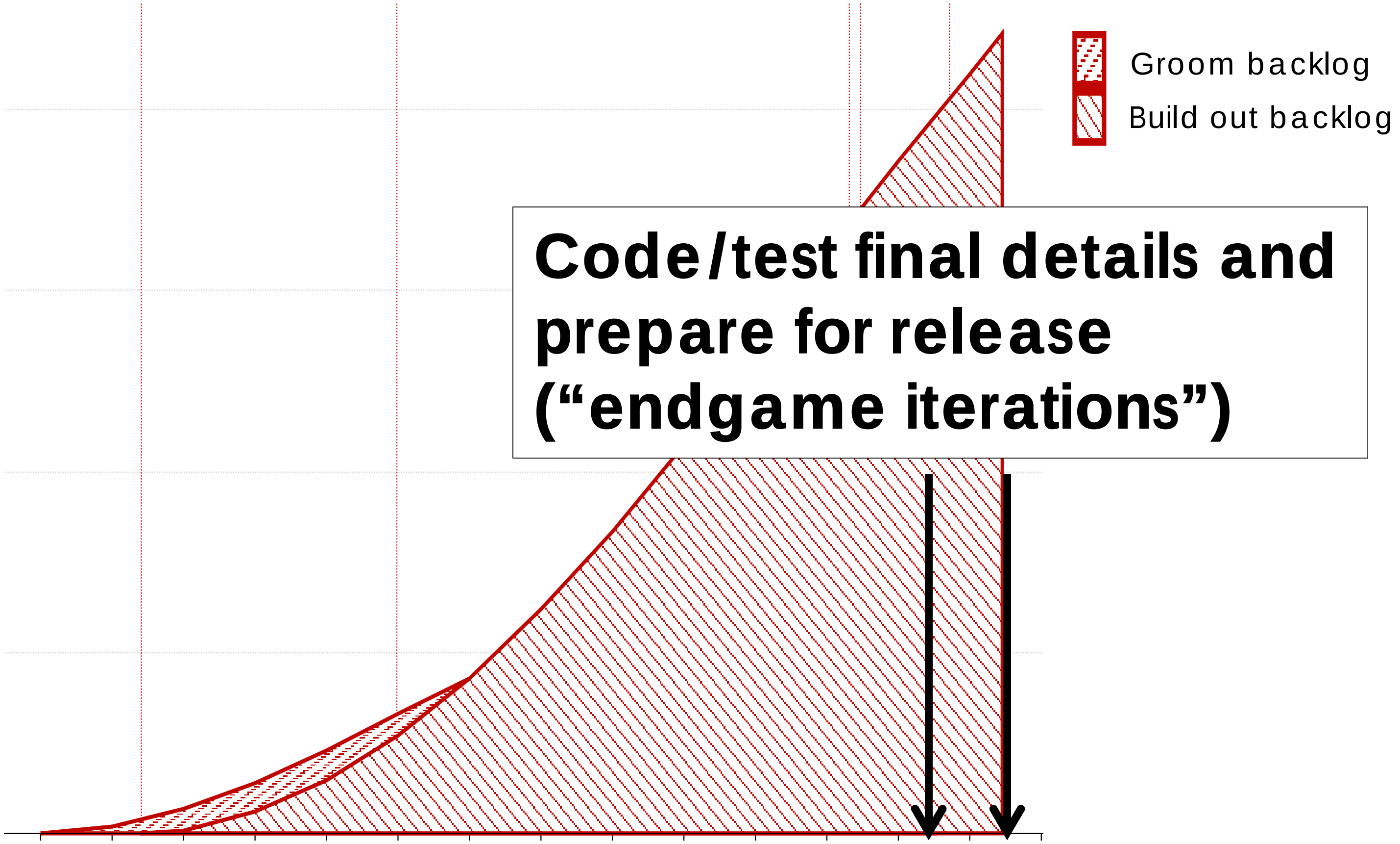
Groom just enough of the backlog to get going ("iteration 0")

# Emergent Requirements



Just-in-time backlog grooming and requirements detail discussions concurrent with coding and testing

# Emergent Requirements



Code/test final details and prepare for release (“endgame iterations”)

# Agile phase tuning in SLIM-Estimate

- Agile Adjuster adjusts phase tuning based on estimated number of iterations
- Based on your methodology choices, you pick:
  - Number of iterations spent grooming the backlog before coding starts (“iteration 0”)
  - Number of extra development iterations spent in final changes and other activities to get ready to release (“endgame”)

# To Earn PDUs for this Webinar:

- Title: Agile Estimation: Beyond the Myths Part 1
- ID: WEB042215

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

QSM® The Intelligence behind Successful Software Projects

# 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