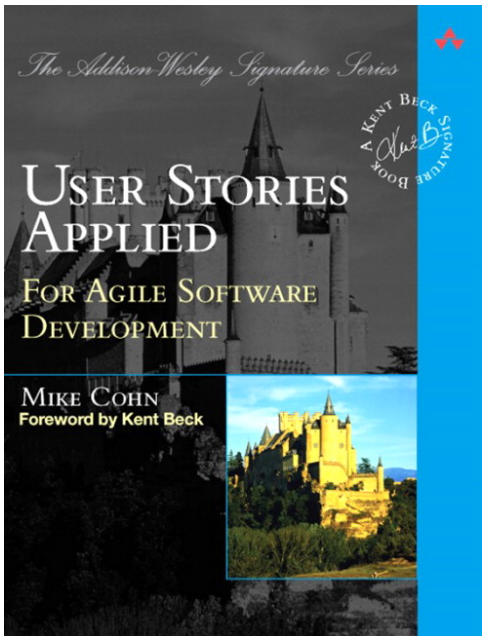
An Overview of Agile Estimating & Planning

Mike Cohn

March 17, 2004

SD WEST 2004 ADVANCE YOURSELF

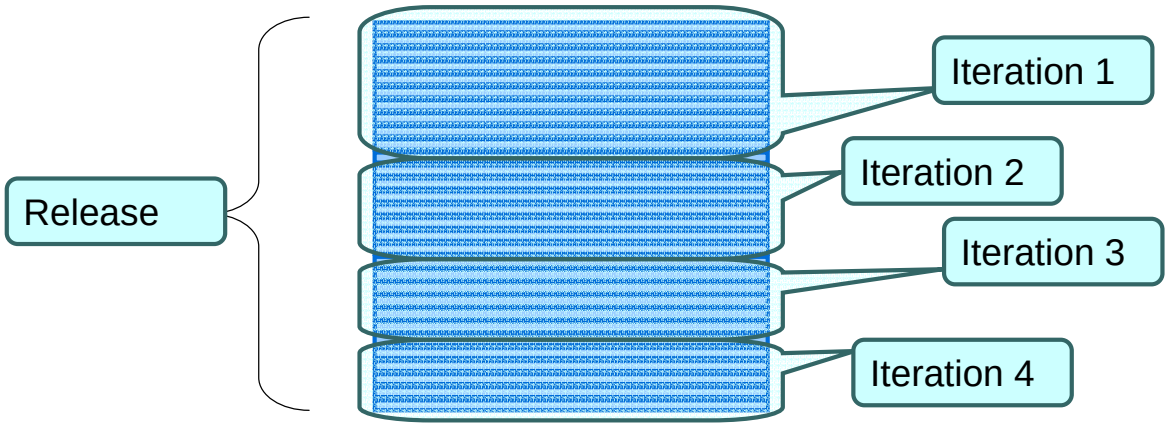
My books and background



- Programming for 20 years
  - Author of four programming books
- Past consulting to Viacom, Fidelity Investments, Procter & Gamble, NBC, United Nations, Citibank, other smaller companies
- Founding member and director of the Agile Alliance
- Currently VP, Engineering with Fast401k in Denver

MARCH 15-19, 2004, SANTA CLARA, CA

Releases and iterations



Today's agenda

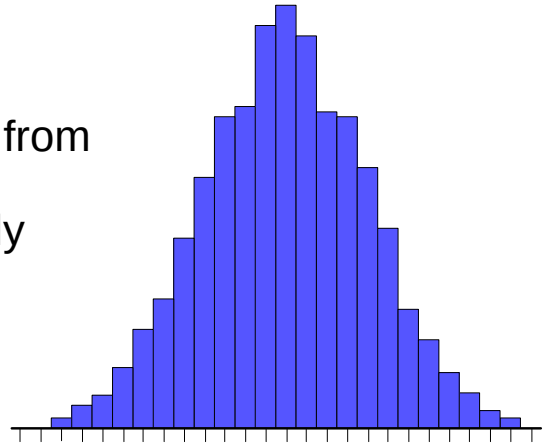
- Why traditional approaches fail
- What to estimate
  - Ideal time
  - Magnitude / complexity
- How to estimate
- Planning
  - Releases
  - Adding a critical chain buffer
- Why agile planning works

Why plans go wrong

1. Tasks are assumed to be independent
2. Lateness is passed down the schedule; earliness is not
3. The Student Syndrome

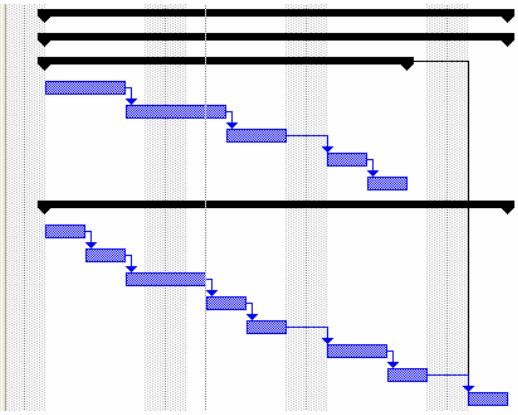
1. Task independence

- Sum of five dice
- Central Limit Theorem
  - Sum of a number of independent samples from any distribution is approximately normally distributed
- This means that
  - some are bigger
  - some are small
  - but overall things average out



Does CLT apply to software?

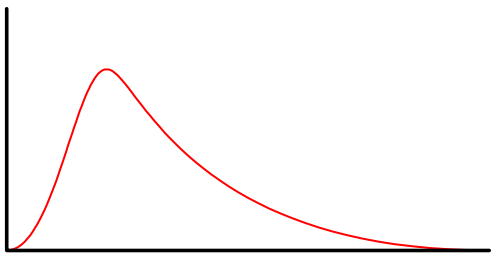
Sprint 4	17 days	Mon 3/4/02	Tue 3/26/02	
Analysis Manager	17 days	Mon 3/4/02	Tue 3/26/02	
Database	14 days	Mon 3/4/02	Thu 3/21/02	
Linkage	4 days	Mon 3/4/02	Thu 3/7/02	
SIMWALK	3 days	Fri 3/8/02	Tue 3/12/02	4
CRI-Map	3 days	Wed 3/13/02	Fri 3/15/02	5
Genehunter	2 days	Mon 3/18/02	Tue 3/19/02	6
Inheritance Checking	2 days	Wed 3/20/02	Thu 3/21/02	7
Client	17 days	Mon 3/4/02	Tue 3/26/02	
Create, rename, delete analysis	2 days	Mon 3/4/02	Tue 3/5/02	
Edit settings	2 days	Wed 3/6/02	Thu 3/7/02	10
Sample selection	2 days	Fri 3/8/02	Mon 3/11/02	11
Marker selection	2 days	Tue 3/12/02	Wed 3/13/02	12
Variable Selection	2 days	Thu 3/14/02	Fri 3/15/02	13
Integrate with Task Manager	3 days	Mon 3/18/02	Wed 3/20/02	14
Run in background	2 days	Thu 3/21/02	Fri 3/22/02	15
Invoke analysis	3 days	Mon 3/25/02	Tue 3/26/02	3,16



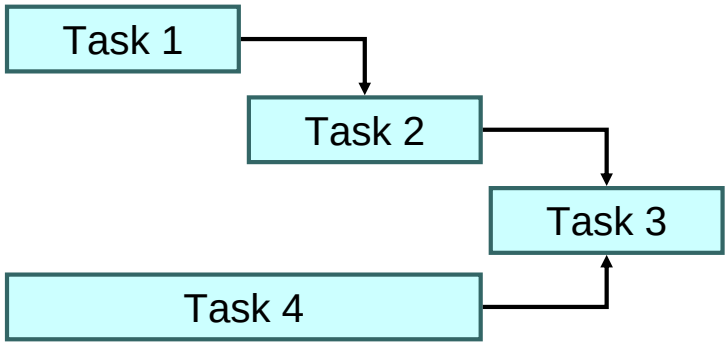
Highly correlated tasks

CLT and software

- The tasks on a software Gantt chart are not independent
  - Many tasks involve similar work; if one estimate is wrong the others tend to be wrong
  - There may be systematic error in the estimates
    - “Jay Days”
- Software estimates tend not to be normally distributed
  - When asked for a point estimate programmers respond with the mode



2. Lateness is passed along the schedule



- Task 3 starts:
  - LATE if 1, 2 *or* 4 is late
  - EARLY only if 2 *and* 4 are early, and resource is available

SD WEST 2004 ADVANCE YOURSELF

3. Student syndrome

Definition

- Starting a task at the last possible moment that does not preclude an on-time completion

Example

- Starting a term paper the night before it's due

MARCH 15-19, 2004, SANTA CLARA, CA

What happens with student syndrome

- Estimate is based on this

- But we behave like this

Today's agenda

- ☑ Why traditional approaches fail
- ❑ What to estimate
  - ❑ Ideal time
  - ❑ Magnitude / complexity
- ❑ How to estimate
- ❑ Planning
  - ❑ Releases
  - ❑ Adding a critical chain buffer

Ideal time

- An estimate of how long something would take if:
  - It's the only thing you work on
  - You have everything you need at hand when you start
  - There are no interruptions

SD WEST 2004 ADVANCE YOURSELF

Calendar time vs. ideal time

Calendar time

- Monday has 8 hours
- Each week has 40 hours

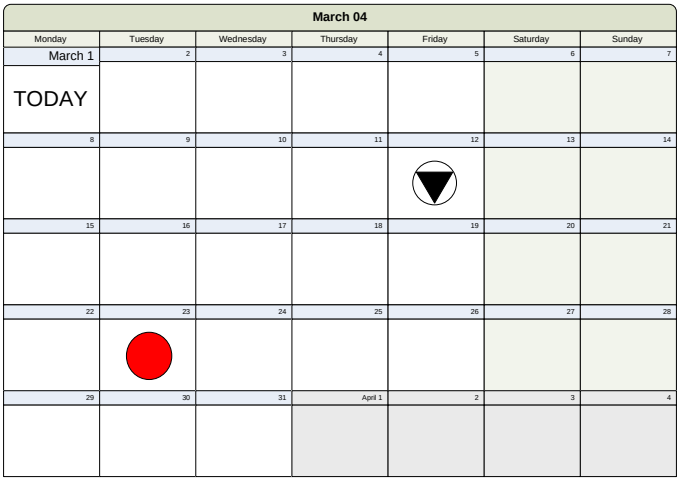
Ideal time

- Time on task
- Monday has
  - 3 hours of meetings
  - 1 hour of email
  - 4 hours of programming (time-on-task)

MARCH 15-19, 2004, SANTA CLARA, CA

"How long will this take?"

- "Two weeks."
- Two *calendar* weeks or two weeks worth of *time on task*?



Factors affecting ideal time

- Vacations
- Sick time
- All-company meetings
- Department meetings
- Demos

- Personnel issues
- Phone calls
- Special projects
- Training
- Email

- Reviews & walk-throughs
- Interviewing candidates
- Spikes
- Leaves of absence
- Sabbaticals

Ideal time vs. calendar time

- It's easier to estimate in ideal time
- It's too hard to estimate directly in calendar time
  - Need to consider all the factors that affect calendar time at the same time you're estimating

But, there's a problem

- Whose ideal time? Yours? Mine?

How do we add
Your Ideal Time
to
My Ideal Time?

Experienced Senior Programmer Days

- How?
  - Define an archetypal programmer and estimate how long it will take her
  - I like an “Experienced Senior Programmer”
    - But it can vary and depends on the team
- Why?
  - Estimates can be more honest
    - If questioned, “Oh, it wouldn’t take *me* that long.”
  - Bias toward insufficient estimates goes away
  - Estimates can be added and compared

SD WEST 2004 ADVANCE YOURSELF

Disadvantages of ideal time

- Can’t add your ideal time to my ideal time
  - Without estimating in something like “Experienced Senior Programmer” days
  - But it can be hard to estimate someone else’s ideal time
- Need to re-estimate whenever we get better or when we know something new about a task
- Developers may make an implicit conversion
  - “Two ideal days is about a week. I think I could do this in a week. I’ll say it’s two ideal days.”

MARCH 15-19, 2004, SANTA CLARA, CA

Advantages of ideal time

- Very tangible and understandable
  - Easy to get started with
- Straightforward to convert from ideal time to calendar time

Magnitude

- The “bigness” of a task
- Influenced by
  - Complexity
  - Our current knowledge
  - How much of it there is
- Relative values are what is important:
  - “A login screen is a 2.”
  - “A search feature is an 8.”
  - “A login screen is small.”
  - “A search feature is large.”

What are the magnitudes of these?

Develop 100 screens, each with 1-2 fields

Code 1 screen with 200 fields on it

Remove the recursion from the ABC class and make it thread safe

Write a “Hello, World” servlet

Problems with magnitude

- Values must be meaningful and distinguishable
  - How do you tell a “67” from a “68”?
- Eventually you need to convert an estimate of magnitude into an estimate of duration
  - “We’ll be done in 8 mediums, 3 smalls and 4 larges.”
  - “We’ll be done in 43 Gummi Bears.”
- Developers may make an implicit conversion
  - “Most 3s take a day, this seems like a day; I’ll say it’s a 3.”
- Can feel very uncomfortable at first
- Very hard to estimate initial velocity

Advantages to magnitude

- Some developers find it much easier to say “this is like that”
- The abstractness can help developers from feeling coerced into giving an estimate that meets an expected deadline
  - “My boss wants this in two weeks, I guess I’ll say ‘two weeks.’”
- Can be done very quickly, once it’s familiar
- Less need to re-estimate than ideal time
  - Something that used to take 1 ideal day might now take ½ ideal day (as the team improves)
  - Something that is “big” is still big; even though the team may be faster

SD WEST 2004 ADVANCE YOURSELF

Story points

- A story point is either:
  - 1 ideal day
  - 1 unit of measure for magnitude

MARCH 15-19, 2004, SANTA CLARA, CA

What I do

- Start with ideal time
  - It gives a team a nice foundation for the initial stories
  - Helps them get started
  - I define “1 Story Point = 1 Ideal Day”
- Gradually convert team to thinking more about magnitude
  - This story is like that story
  - Stop talking about how long it will take

Today’s agenda

- ☑ Why traditional approaches fail
- ☑ What to estimate
  - ☑ Ideal time
  - ☑ Magnitude / complexity
- ❑ How to estimate
- ❑ Planning
  - ❑ Releases
  - ❑ Adding a critical chain buffer
- ❑ Why agile planning works

Use the right units

Ideal time

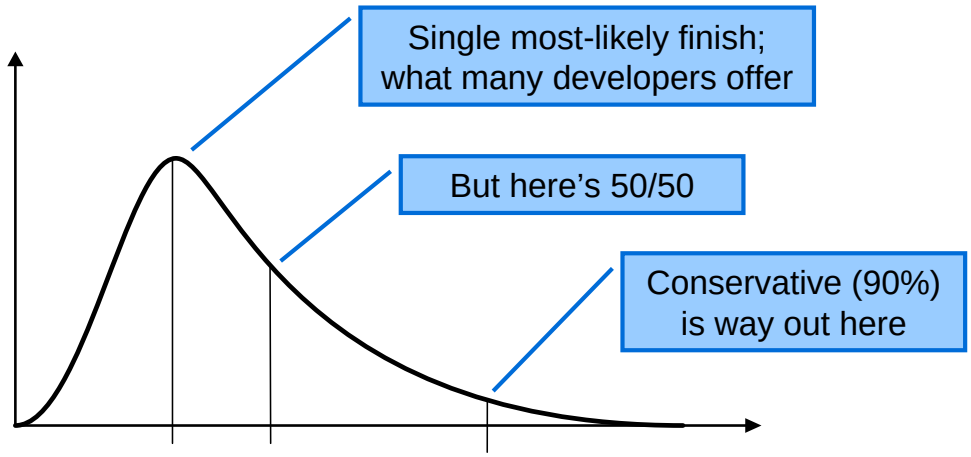
- Can you distinguish a 17-hour task from an 18-hour task?
- Can you distinguish a ½ day from a 1 day task?

Magnitude

- Can you distinguish a 17 from an 18?
- A ½ from a 1?

- Use units that make sense, such as:
  - 0, ½, 1, 2, 3, 5, 10, 20, 40
  - 0, ½, 1, 2, 3, 5, 8, 13, 21, 34

State your assumptions



pletion

Give both 50% and 90% estimates

- 50% estimates
  - Remove all *local safety*: no “padding”
  - An estimate you should / will miss half the time
- 90% estimates
  - Not really a worst case
    - No lightning strikes or busses running over people
  - Keep in mind that you’ll even exceed this estimate occasionally

SD WEST 2004 ADVANCE YOURSELF

Approaches to estimating

- Gut feel
- Analogy
- Disaggregation
- Wideband Delphi

MARCH 15-19, 2004, SANTA CLARA, CA

Gut feel

- Good as a reasonableness check

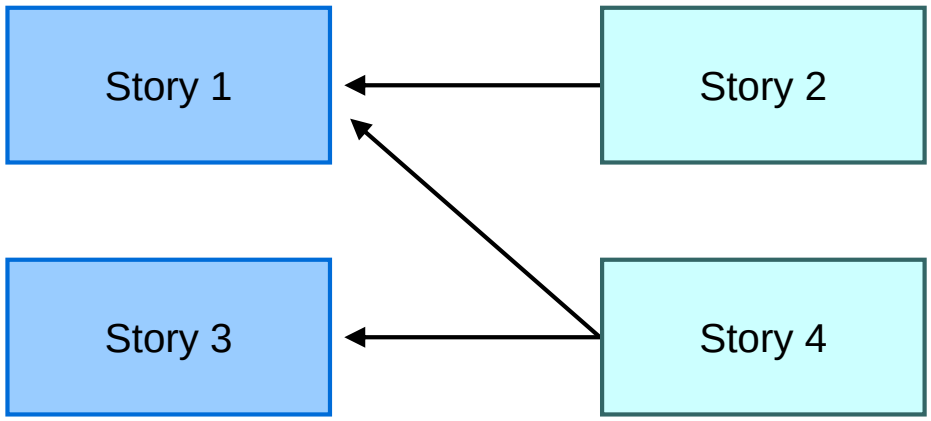
SD WEST 2004 ADVANCE YOURSELF

Analogy

- Analogy
  - "This story is like that story, so its estimate is what that story's estimate was."
  - Works especially well if baseline story has been coded
  - Triangulate
    - Estimate by analogy to two different stories

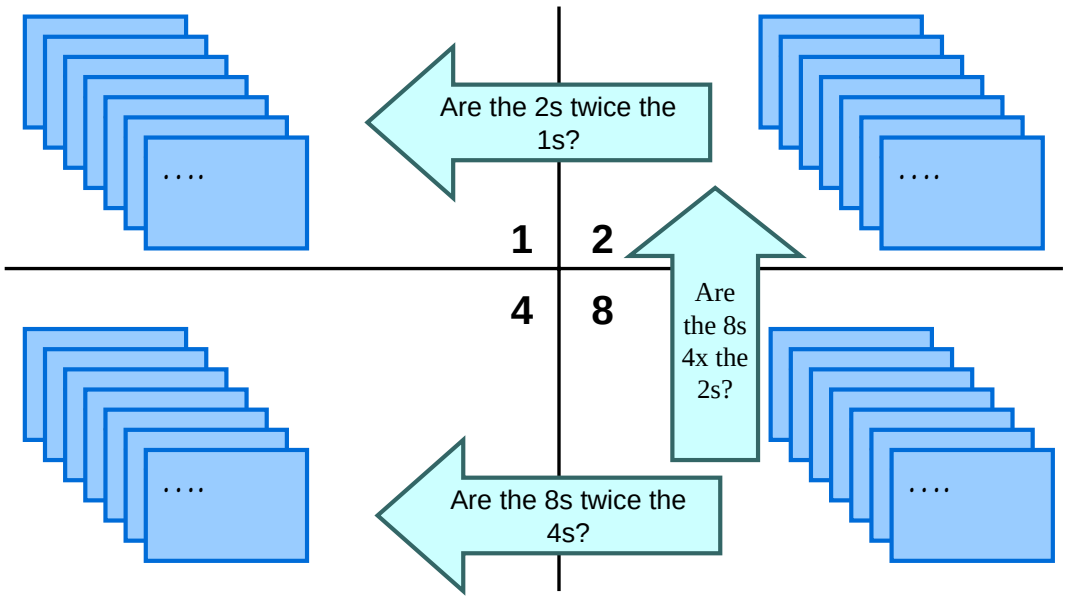
MARCH 15-19, 2004, SANTA CLARA, CA

Triangulation



- Confirm estimates by comparing the story to multiple other stories.
- Group like-sized stories on table or whiteboard

Check a few stories in each direction



Disaggregation

- Breaking a big story into littler stories or tasks
- You know how long the smaller tasks take
  - So, disaggregating to something you know lets you estimate something bigger you don't know
- Sometimes very useful
- But disaggregating *too far* causes problems
  - Forgotten tasks
  - Summing lots of small errors can be big number

Wideband Delphi

- An iterative approach to estimating
- Steps
  1. Identify small group of estimators and give them stories to read before the meeting
  2. Each estimator is given a deck of cards, each card has a valid estimate written on it
  3. A moderator reads a story and it's discussed briefly
  4. Each estimator selects a card that's his 50% estimate
  5. Cards are turned over so all can see them
  6. Discuss differences (especially outliers)
  7. Re-estimate until estimates converge
  8. Use the highest value or repeat for a 90% estimate

Wideband Delphi—an example

Estimator	Round 1	Round 2
Susan	4	4
Rafe	7	5
Ann	2	4
Sherri	4	4

Anonymity of estimates

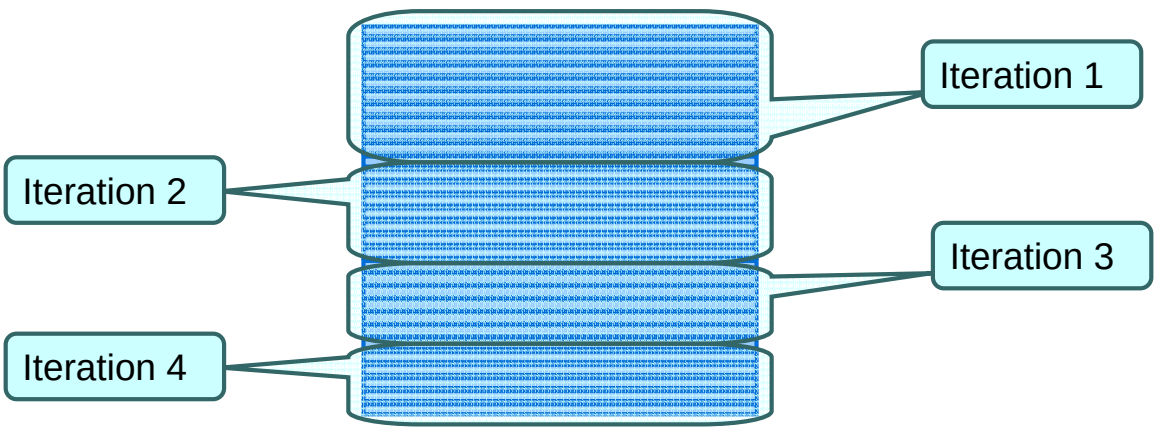
- Boehm recommends doing estimates anonymously
- Everyone submits estimates, gets written feedback
- There are advantages
  - People speak more freely
  - Discussion won't be dominated by a single strong personality or the lead developer
- But there are also disadvantages
  - Tends to take longer
  - Less communication
- Goes against XP's value of courage
- My recommendation
  - No anonymity, but use your judgment

Today's agenda

- ☑ Why traditional approaches fail
- ☑ What to estimate
  - ☑ Ideal time
  - ☑ Magnitude / complexity
- ☑ How to estimate
- ❑ Planning
  - ❑ Releases
  - ❑ Adding a critical chain buffer
- ❑ Why agile planning works

What we'd like to do

- Take a prioritized stack of user stories
- Figure out how much we can do per iteration
- And then know how many iterations it will take



Different dimensions to prioritization

Technical

- Risk that the story cannot be completed as desired
- Impact the story will have on other stories if deferred

Customers / Users

- Desirability of the story to a broad base of users
- Desirability of the story to a small number of important users
- Cohesiveness of the story to other stories.

Who wins

- Customer wins—always
- But need developer input in order to prioritize

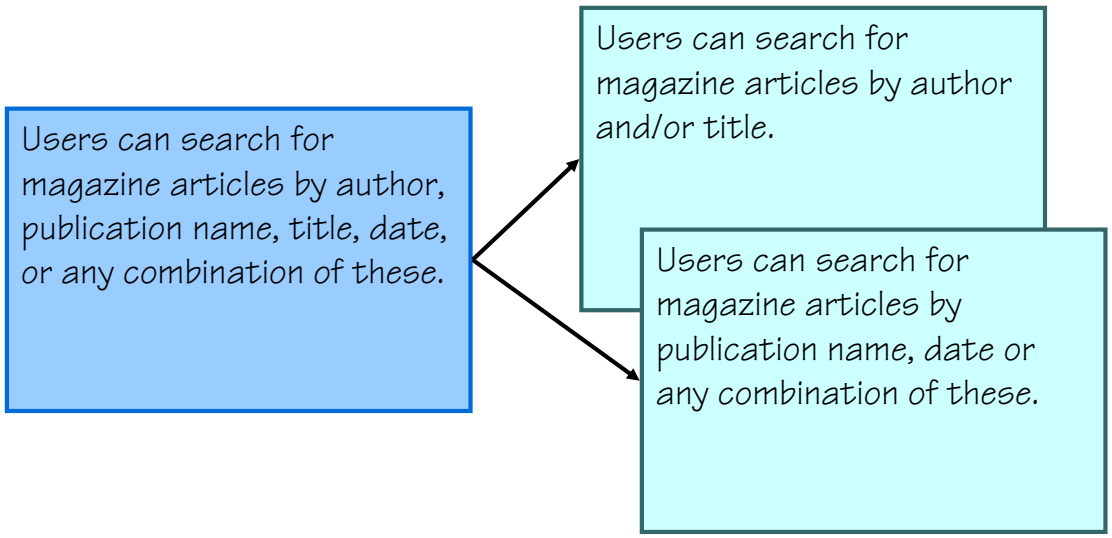
Customer cannot prioritize without knowing the cost of the stories

The user can book a new trip based on a previous trip.

3—5 days

Developers are best at identifying dependencies between stories

Split stories with mixed priorities



Risky stories vs. juicy stories

- Agile is firmly in the camp of doing the “juicy bits” first
- But cannot totally ignore risk
  - If some stories are very risky, the developers need to tell the customer

Infrastructural stories

- Infrastructural stories are usually best assessed by the risk of deferring them (but still doing them later)

Be able to generate 50 stock chart images per second.

Is this performance achievable on targeted hardware?

Can we still use Java or should we do this natively?

What type of caching do we need to achieve this?

How much can we do per iteration?

- Velocity
- Our best guess is that we can do next iteration what we did last iteration
  - “Yesterday’s Weather” (Beck & Fowler)
- But sometimes we don’t have a last iteration

Getting an initial velocity

Use historicals
- Great if you have them from a similar project by the same team

Run an iteration
- Great if you can do it
- Not always viable, e.g.,
  - No team in place yet
  - Boss wants early estimate

Forecast
- May not always be preferred approach
- But, you need it as a tool

SD WEST 2004 ADVANCE YOURSELF

Forecasting velocity from ideal time

- Estimate each developer's productivity relative to the archetypal Experienced Senior Programmer used in the estimates
- Considerations
  - Programming skill
  - Domain knowledge
  - Availability to actual code
  - Vacation

MARCH 15-19, 2004, SANTA CLARA, CA

Example: forecasting initial velocity

Developer	Iteration 1	Iteration 2	Iteration 3	Thereafter
Susan	.5	.6	.7	.7
Ann	.5	.5	.5	.5
Randy	.2	.3	.4	.4
Clark		.2	.3	.4
Vlade	.5	.6	.7	.7
Chris	.8	.9	1.0	1.0
Total	**2.5**	**3.1**	**3.6**	**3.7**

- This tells you how many ideal programmers you have working per calendar day

SD WEST 2004 ADVANCE YOURSELF

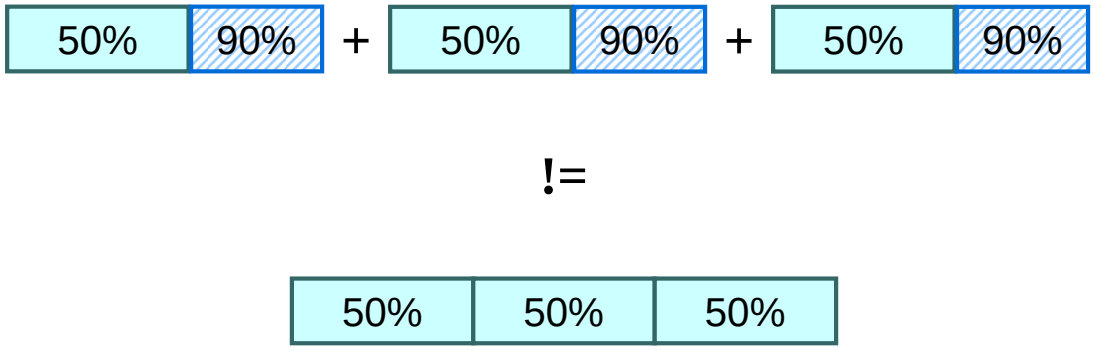
Forecasting velocity from magnitude

- Starting with the highest-priority story, select as many stories as you think will fit in the first iteration
- Break each story into smaller tasks (< 1 calendar day)
- When the iteration feels full, stop and see how many story points were brought in
- That's your guess at velocity

MARCH 15-19, 2004, SANTA CLARA, CA

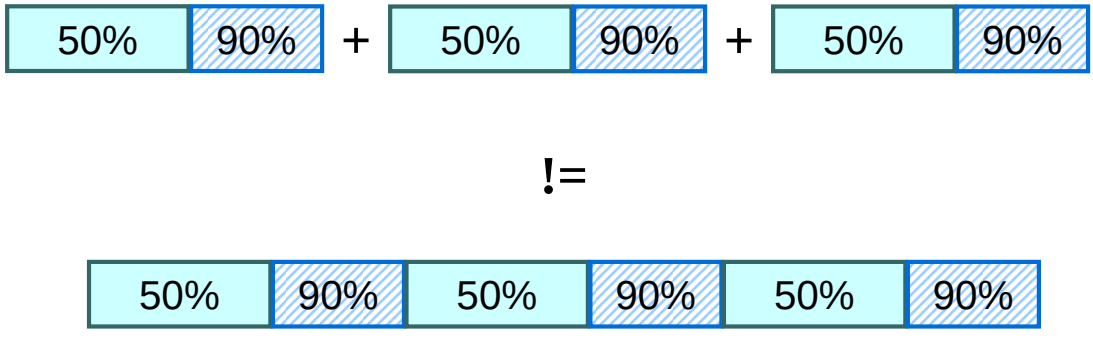
Release planning

- We can't add the 50% estimates together
  - That assumes everything goes smoothly
  - Overall schedule will be too short



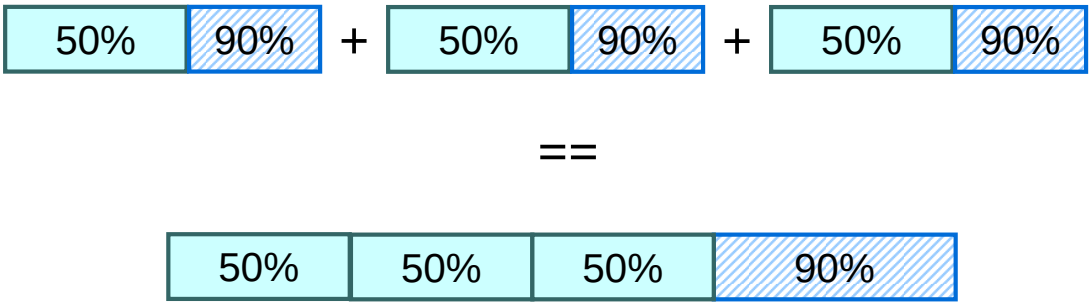
Release planning

- We can't add the 90% estimates together
  - That assumes that everything goes wrong
  - Overall schedule will be too long

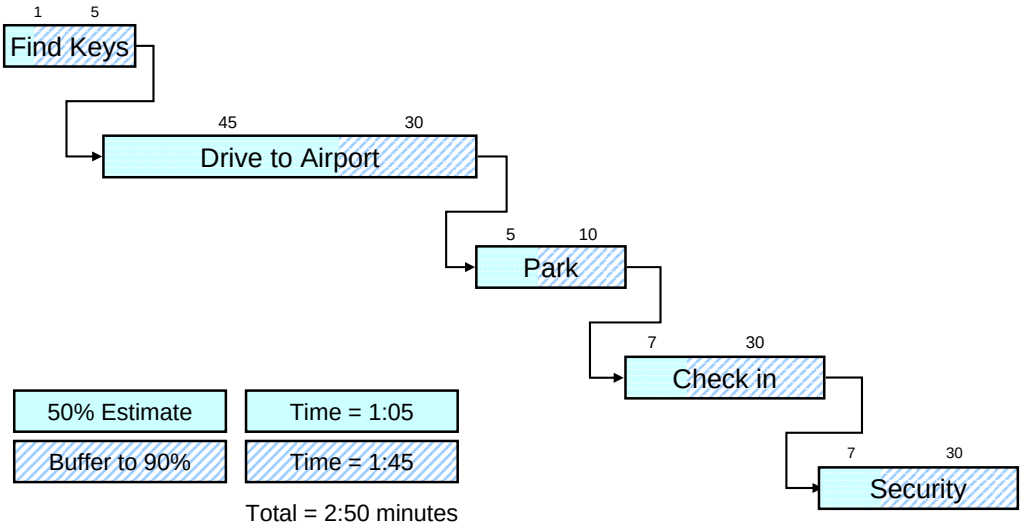


The solution

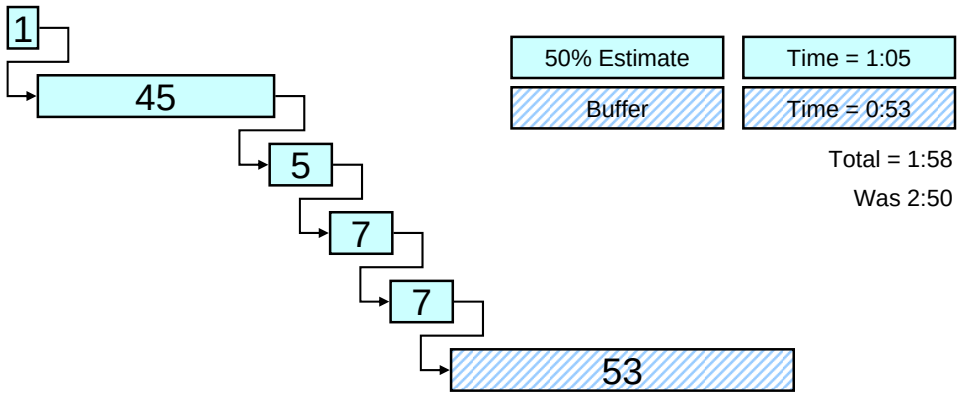
- We add the 50% estimates
- And buffer the overall project, rather than the tasks



My trip to the airport



My trip with a project buffer



A project buffer isn't padding

- Padding is extra time you don't think you'll need but add to be safe
- You will need the project buffer
  - Even with the project buffer you're not guaranteed to be done on time
- I had about a 3% chance of making it to my flight in 65 minutes

$$50\% \times 50\% \times 50\% \times 50\% \times 50\% = 3.125\%$$

- Would you call something that increases your odds of success from 3% "padding"?

How long should the buffer be?

- Simple rule
  - Use 50% of the unbuffered (50%) schedule
- More sophisticated, usually better

$$\sqrt{(w_1-a_1)^2+(w_2-a_2)^2+\cdots+(w_n-a_n)^2}$$

  - w = worst case
  - a = average case

Sample buffer calculation

Story	50%	90%	(90%—50%)²
Story 1	2	5	9
Story 2	3	5	4
Story 3	1	1	0
Story 4	1	3	4
Story 5	5	8	9
Story 6	5	6	1
Total	17	28	27

$$Schedule = 17 + \sqrt{27} = 17 + 5.2 = 22$$

Full example of planning a release

Story	50%	90%	(90%—50%)²
Story 1	2	5	9
Story 2	3	5	4
…	…	…	0
Total	**117**	**200**	**1089**

$$117 + \sqrt{1089} = 117 + 33 = 150$$

Developer	Iteration 1	Iteration 2	Iteration 3	Thereafter
Susan	.5	.6	.7	.7
Ann	.5	.5	.5	.5
Randy	.2	.3	.4	.4
Clark		.2	.3	.4
Vlade	.5	.6	.7	.7
Chris	.8	.9	1.0	1.0
Total	**2.5**	**3.1**	**3.6**	**3.7**

Example, continued

Velocity estimates from previous slide

Iteration	Duration (Days)	Daily Velocity	Story Points in iteration	Cumulative Story Points
Iteration 1	10	2.5	25	25
Iteration 2	10	3.1	31	56
Iteration 3	9	3.6	32	88
Iteration 4	10	3.7	37	125
Iteration 5	10	3.7	37	162

Company holiday

Accumulate 150 Story Points sometime during Iteration 5

Communicating the estimate

- When communicating the estimate to management
  - Don't talk about the project buffer
    - Don't necessarily hide it
    - I include it in a document on the estimation approach, rather than in the estimate itself
  - Clearly state your assumptions
  - Stress that it will be refined
    - Then refine it!
    - Not fair to "refine" it only with a big slip at the end

Why agile planning works

- Why plans go wrong

1. Tasks are assumed to be independent

2. Lateness is passed down the schedule; earliness is not

3. The Student Syndrome

Why agile planning works

1. Tasks are assumed to be independent

- ✓ Stories (the main unit of estimation) are largely independent.

SD WEST 2004 ADVANCE YOURSELF

Why agile planning works

2. Lateness is passed down the schedule; earliness is not

- ✓ No overall Gantt or PERT chart
- ✓ Each day, each person picks what she'll do
  - ✓ Lateness doesn't pass down an agile plan
  - ✓ Earliness does pass down
- ✓ Naturally, there are some dependencies
  - ✓ But these are limited with an agile plan

MARCH 15-19, 2004, SANTA CLARA, CA

Why agile planning works

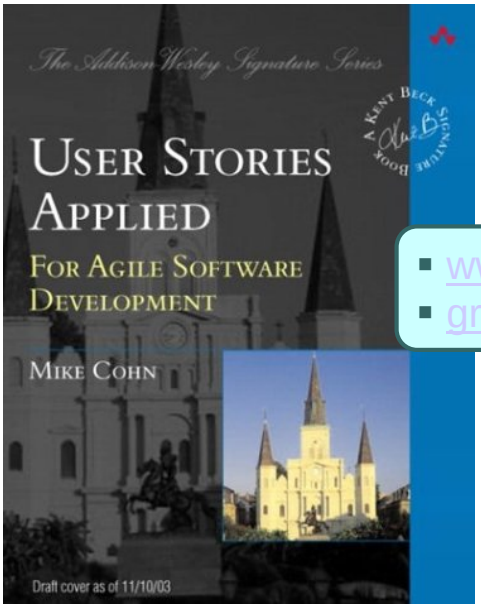
3. The Student Syndrome

- ✓ No Gantt chart saying what to do today and how long to take
- ✓ Increased visibility through daily standup meetings and pair programming

Additionally

- ✓ Agile planning encourages and enforces continuous re-estimation and recalibration

For more on user stories



- www.userstories.com
- groups.yahoo.com/group/userstories

Where to go next?

Agile Alliance

Agile Planning	groups.yahoo.com/agileplanning
Agile in General	www.agilealliance.com
Scrum	www.mountaingoatsoftware.com/scrum

My contact information

Email

- mike@userstories.com
- mike.cohn@computer.org

- www.userstories.com

Website