Agile Software Development

Lecture 5: Project Planning

Mahmoud El-Gayyar

elgayyar@ci.suez.edu.eg

Slides are a modified version of the slides by Prof. Kenneth M. Anderson

Outline (Project Planning)

- *Milestones*
- *The dangers of adding more people*
  - The Mythical Man-Month
- *Velocity*
- *Burn down charts*
- *Big Board*

Project Planning I

- ***what to do if your estimate is too big?***
- *In the example in the book, the answer was clear*
- *Our estimate: 489 days (~1.85 years of development time!!)*
- *Customer's Ideal Deadline?*
  - 90 days

Project Planning II

- ***What to do?***
- *Scope the problem*
  - Focus on most critical functionality and see if customer is willing to focus on that subset
  - In general, “scope the problems” means drop features until the remaining features can be completed by the original due date
- *Who does the scoping?*
  - The customer

Milestone 1.0

- *In particular, we are attempting to define what features will go into "**milestone 1.0**"*

- *Milestone 1.0 == first **major** release to the customer*
  - In iterations, you show software for feedback but do not generally deploy the software for production use
  - With milestones, you are delivering software that will go into production use

Milestone 1.0 Do's and Don'ts

- *Do balance functionality with customer impatience*
  - Help customer understand what can be done before the deadline
  - Help them understand that features are being delayed not dropped
- *Don't get caught planning nice-to-haves*
  - You need to focus on what's needed: mission critical fun.
- *Don't worry about length (yet)*
  - You're trying to understand your customer's priorities

Soundness Check

- *Once you have identified the features that need to go into Milestone 1.0, recheck your estimate*
  - In the book, since you have less features, the new estimate comes to ***273 days (3/4 of a year)***
  - You still have ***90 days*** to complete the work
  - And we are assuming a ***team size of 1 person***
- *In this situation, we would be forced to reprioritize with the customer and cut functionality to the bone*
- *OR…*

Add More People I

- *… we could add more people!*
- *Lets increase the team size to 3 people*
- *273 / 3 = 91 days of work and we have 90 days left*
  - That should do the trick assuming a few sleepless nights as the deadline approaches, right?
- *WRONG!*
  - First, we have 90 calendar days, not 90 work days!
  - Recall that we get roughly 20 works days per month
  - So a team of 3 can accomplish roughly 180 days worth of work over 90 calendar days ASSUMING ALL GOES WELL

Add More People II

- *Second, you can't assume that the customer won't change things on you as you move forward*
- *Third, you can't assume that the two new developers will be up to speed on the project and ready to put full productive work days into the project on day one*
- *With three people, we now have*
  - two people to train and get ready to work on the project
  - three communication paths to manage (previously zero)

Mythical Man-Month (I)

- *Essays on Software Engineering is a book on software engineering and project management by **Fred Brooks**, whose central theme is that "**adding manpower to a late software project makes it later**". It looks at the unit of the man-month*
  - sometimes used by management to schedule large projects

But First: The Tar Pit

- *Developing large systems is "sticky"*
  - Projects emerge from the tar pit with running systems
  - But most missed goals, schedules, and budgets
  - "No one thing seems to cause the difficulty. But the accumulation of simultaneous and interacting factors brings slower and slower motion."
- *The analogy is meant to convey that*
  - It is hard to detect the nature of the problem(s) facing software development
- *Brooks begins by examining the basis of software development*
  - e.g. system programming

What makes programming fun?

- *Sheer joy of creation*
- *Pleasure of creating something useful to other people*
- *Creating (and solving) puzzles*
- *Life-Long Learning*

What's not so fun about programming?

- *You have to be perfect!*
- *You are rarely in complete control of the project*
- *Design is fun; debugging is just work*
- *Testing takes too long!*
- *The program may be obsolete when finished!*

Why are software project's late?

- *Estimating techniques are poorly developed*
- *Our techniques confuse effort with progress*
  - The Mythical Man-Month
- *Since we are uncertain of our estimates, we don't stick to them!*
- *Progress is poorly monitored!*
- *When slippage is recognized, we add people*
  - "Like adding gasoline to a fire!"
- *Or may be Optimism*

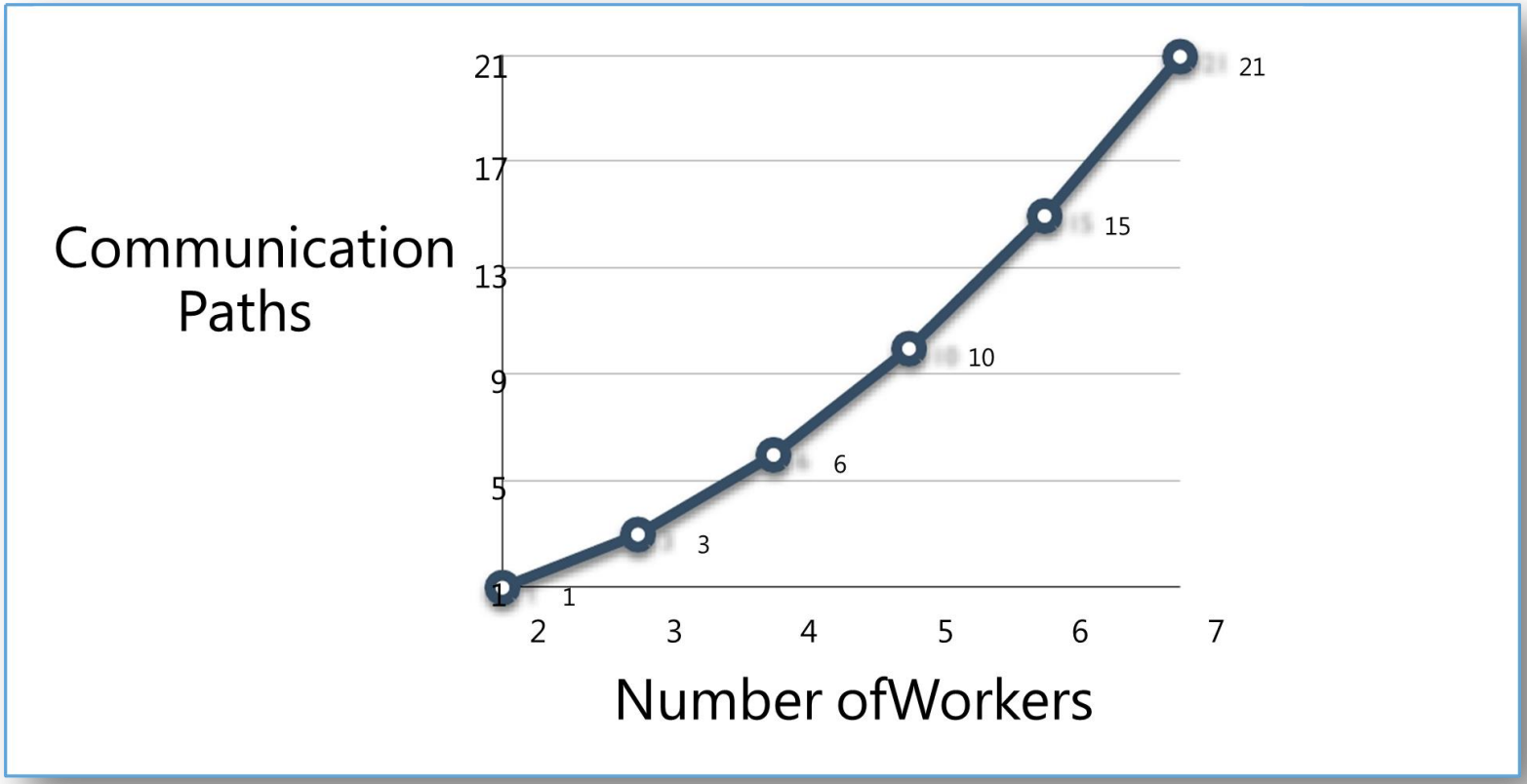
Mythical Man-Month (II)

- *The unit of the person-month implies that workers and months are interchangeable.*
  - However, this is only true when a task can be partitioned among many workers with no communication among them!
- *Brooks points out that cost does indeed vary as the product of the number of workers and the number of months. Progress does not!*
  - When a task is sequential, more effort has no effect on the schedule
  - And, unfortunately, many tasks in software engineering have sequential constraints. Especially debugging and system testing

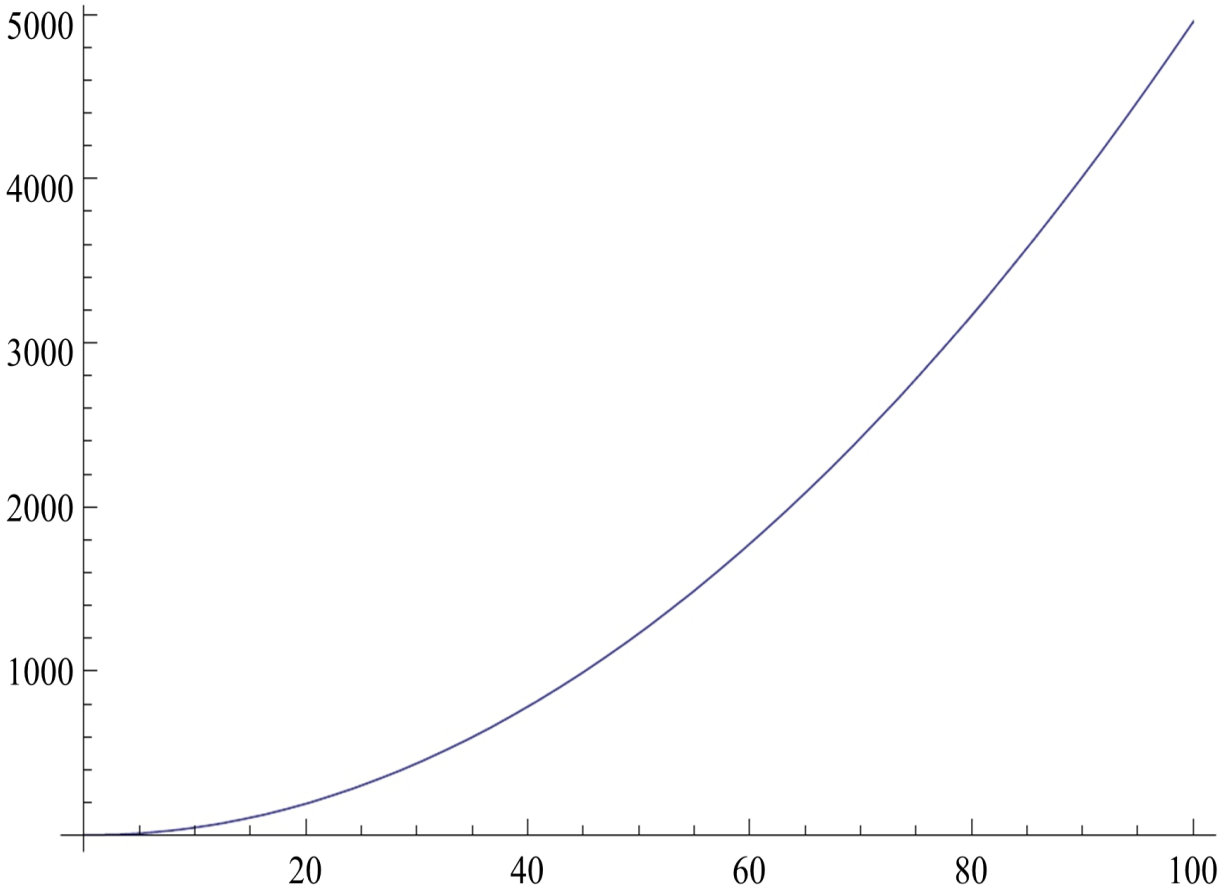
Mythical Man-Month (III)

- *In addition, most tasks require communication among workers*
- *In a software dev. project, communication consists of*
  - training, and
  - sharing information (intercommunication)
- *training will effect effort at worst linearly*
  - (i.e. if you have to train N people individually, it will take N * trainingTime minutes to train them)
  - intercommunication adds n(n-1)/2 to the effort
    - *if each worker has to communicate with every other worker*

Mythical Man-Month (IV)



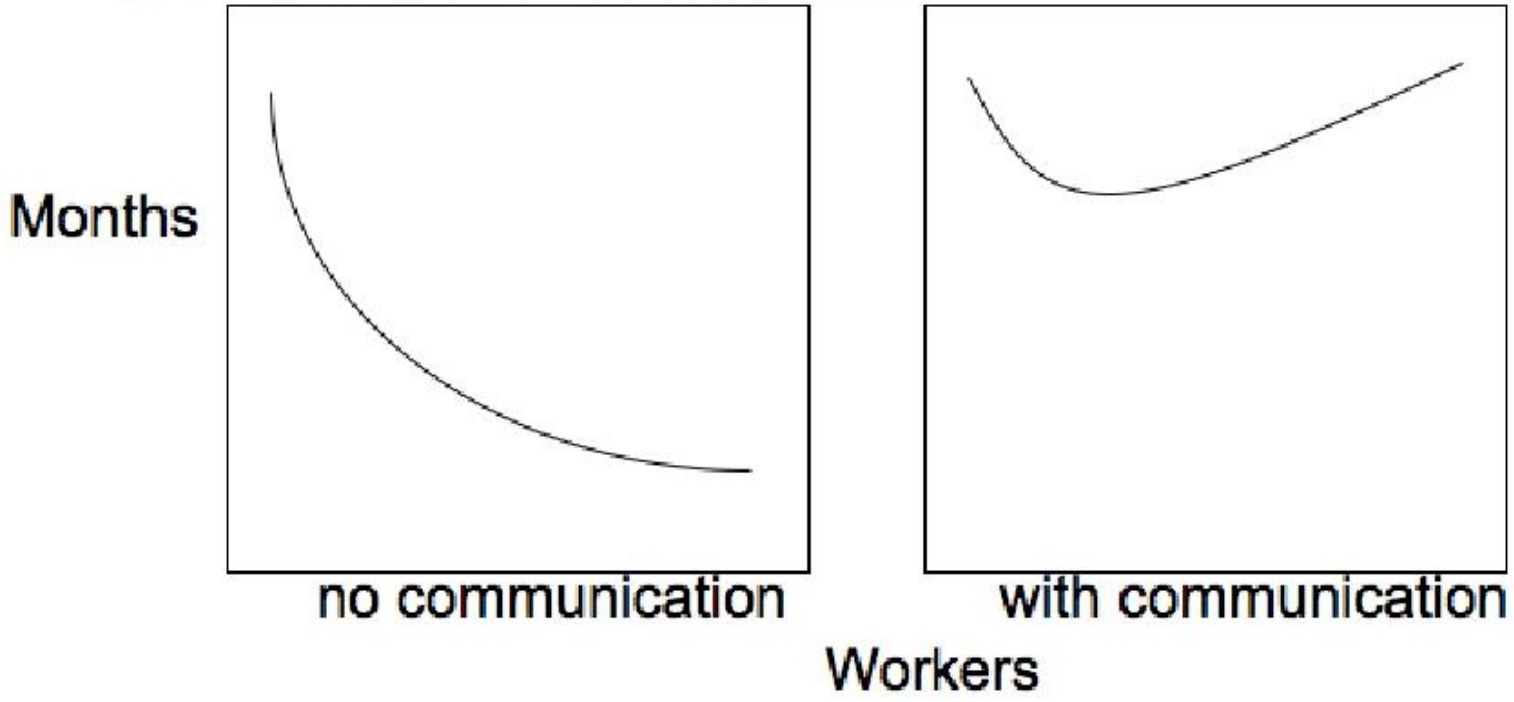
Mythical Man-Month (V)



A 100 person team has 4950 potential communication paths to manage!

Some Benefits

"Adding more people then lengthens, not shortens, the schedule!"
-- (A paraphrase of) Brooks' Law



Back to the Example

- *With 3 developers, we start by assuming that they can produce 180 days of development effort*
- *You then negotiate with the customer until the estimate of all the features in milestone 1.0 is less than 180 days*
  - You then create an iteration plan and get to work
  - Keep your iterations short (30 calendar days, 20 work days)
    - *It helps you deal with change and keep you focused*
  - Keep your iterations balanced (new features, fixing bugs, etc.)

And, now reality sets in

- *We can't necessarily assume 180 days of work from three developers over three calendar months*
- *During the day there are constant interruptions that prevent developers from remaining "in the flow"*
  - rather than 8 productive hours in a work day, you find that you only achieve 5 hours (or less)
- *To account for this, agile methods make use of a concept called "**team velocity**" or "**velocity**"*
  - Velocity is a percentage: given X number of days, how much of that time is productive? ***A default value is 0.7***

Realistic Estimate

$$\frac{\text{Project Estimate (in days)}}{\text{velocity}} == \text{realistic project estimate}$$

- *30 calendar days, 20 work days == 14 days of productive time !!!!*

Example, cont.

- *Now, that we know about velocity, we can use it to estimate how many days of productive work we can achieve during each iteration*

number of developers x working days in iteration x velocity

$$3 \text{ x } 20 \text{ x } 0.7 = 42$$

- *Since we have three iterations:*

$$3 \text{ x } 42 = 126$$

Example, cont.

- *Went from: 3 people could do 270 days of work in 90 days*
- *To: 3 people could do 180 days of work in 90 days*
- *To (finally): 3 people could do 126 days of work in 90 days*
- *Assuming an overhead of 0.7*

- ***Question:* what should we do with our velocity if we add MORE people to the project?**
  - How would you change velocity if we shifted to 4 people?
  - or to 10 people?

Your time !!

- *Exercise 4 : Be the customer*

Your time !!

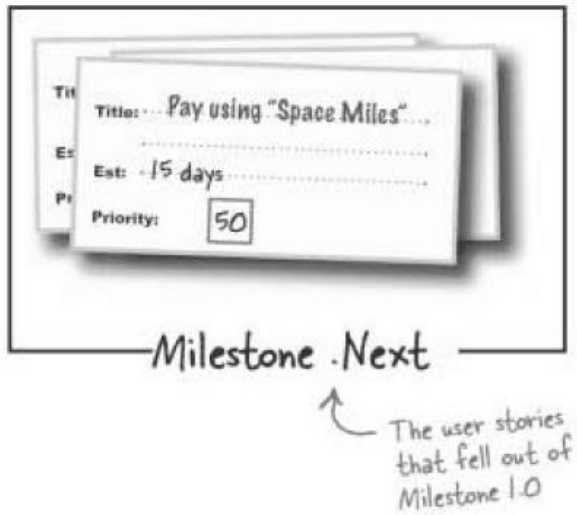
- *Exercise 5 : Add them into iterations (no velocity)*

Your time !!

- *Exercise 6 : Add them into iterations (consider velocity)*

Managing Customers – I

- *The customer will ~~probably~~ definitely not like the change from 273 days of work possible to 126*
  - Since it means a big reduction in what can be accomplished



Managing Customers – II

- *The customer will ~~probably~~ definitely not like the change from 273 days of work possible to 126*
  - Since it means a big reduction in what can be accomplished
- *What to do?*
  - Add an iteration (if they will let you)
  - Explain that overflow work is not lost, just postponed
  - Be transparent about how you came up with your figures
    - *You now have an estimate that you can be confident in*

Your time !!

- *Exercise 7 : Who does what?*

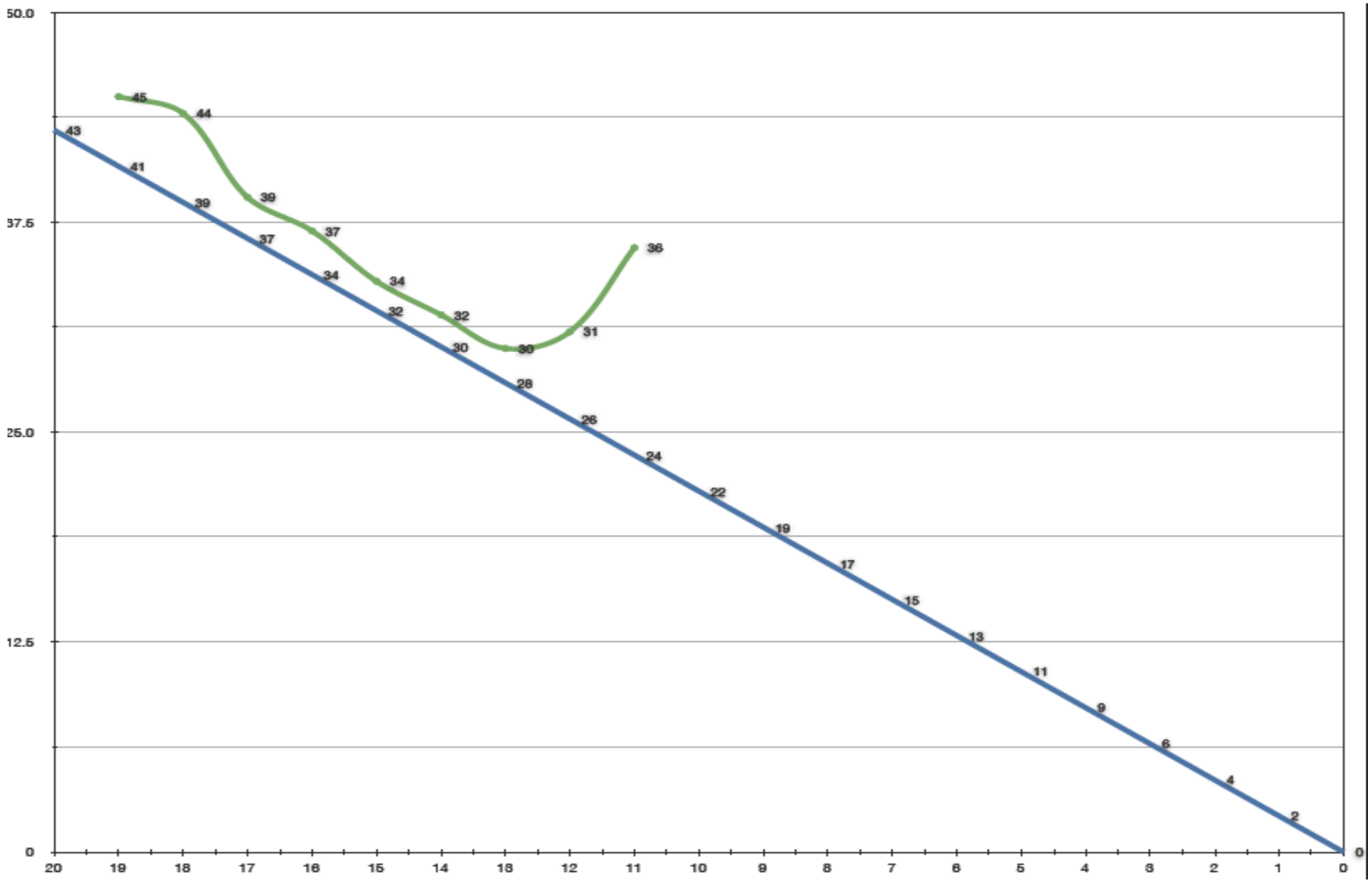
User Stories and Tasks

- *Each user story needs to be split into tasks*
  - Each task then needs an estimate associated with it
  - The entire team should participate in breaking a user story into tasks; planning poker should be used to assign estimates
  - Example User Story: Create a Date in the System (Estimate: 11 days)
  - Tasks
    - *Create a date class that contains events: 3 days*
    - *Create user interface to create, view and edit a date: 5 days*
    - *Create the schema for storing dates in a database: 3 days*
    - *Create SQL scripts for adding/finding/updating dates: 2 days*
  - Total Task Time: 13 days! (Recommend, do estimation on tasks)

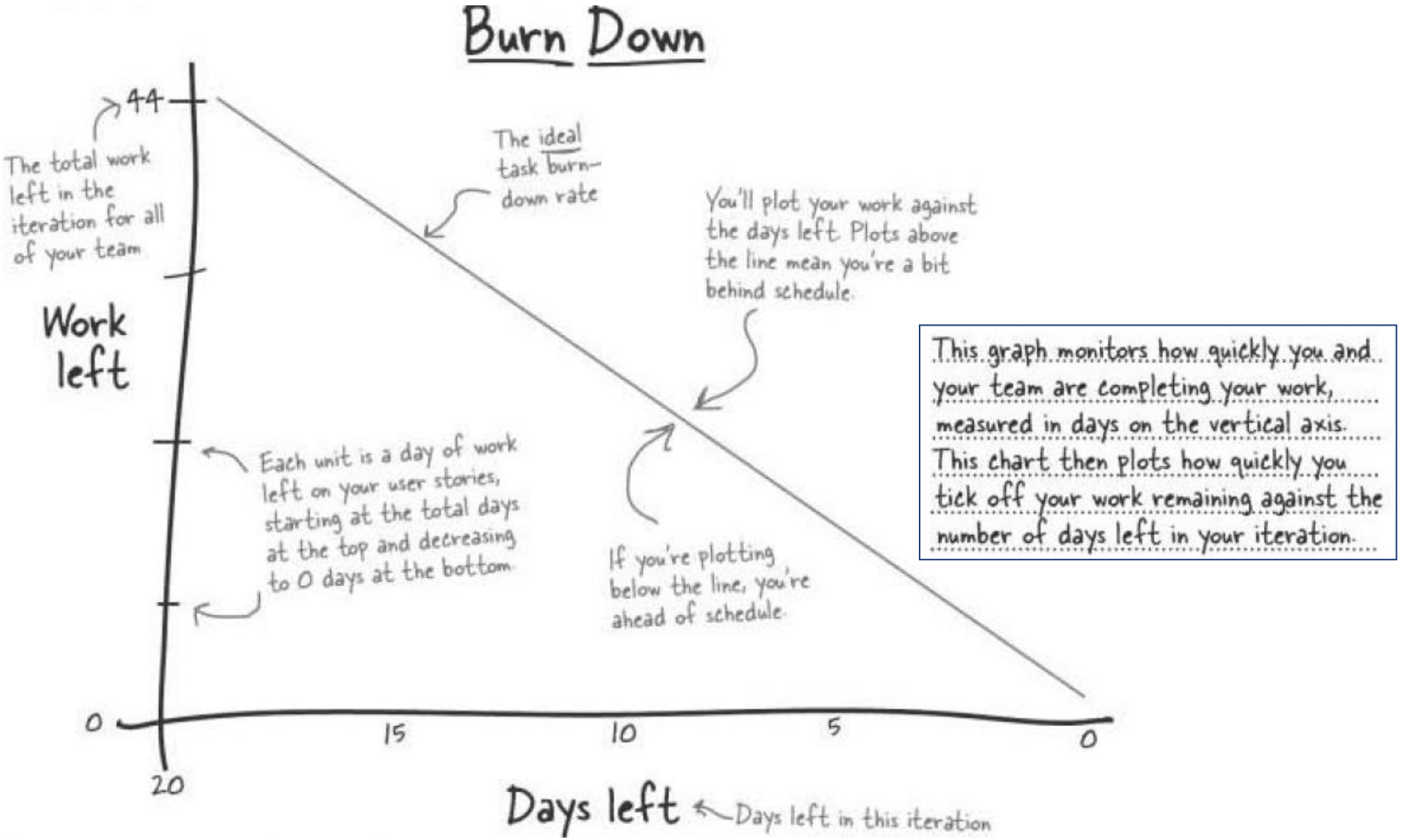
Burn Down Chart - I

- *Fortunately, the burn-down chart gives us a specific action item whenever an estimate changes or work gets done*
  - Update the burn-down chart
- *In the case of an estimate changing, calculate its impact on the work remaining and plot your status*
  - In the book, the original estimate for the iteration was 43 days of productive work; a 2 day increase in the first story pushes the amount of work left to 45 days
  - and they spent a day working on task decomposition
  - The following chart contains this info. plus more

Burn Down Chart - II

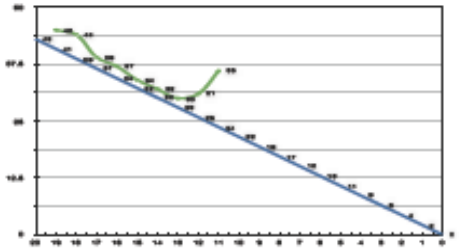


Burn Down Chart - III



The Big Board

- *Once you have a realistic estimate and an iteration plan based on that estimate, you are ready to get started*
- *You will track your progress with a **software development dashboard***
  - A large whiteboard that is partitioned to help your team focus on what is happening during the current iteration
  - It is updated at least once per day during the stand up meeting
  - But could be useful to update it more often than that
  - It is a one-stop shop for getting a “big picture” view of the current iteration

User Stories	In Progress	Complete	Burn Down
			
			Next
	Start with this…		
			Completed

Title: Show Current Deals

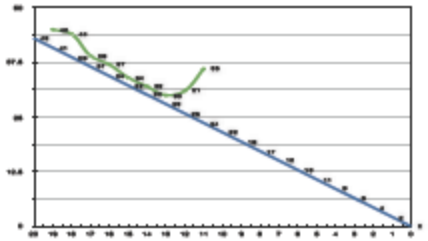
Description: The website will show current deals to Orion's Orbits users.

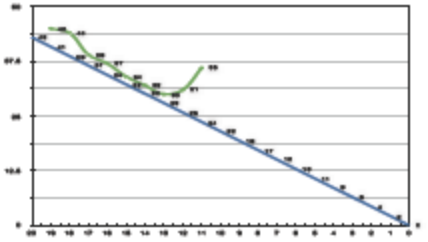
Title: Book package

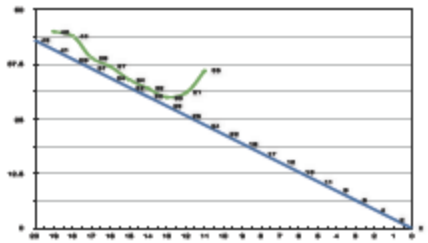
Description: An Orion's Orbits user will be able to book a special package with extras online.

Title: Pay online

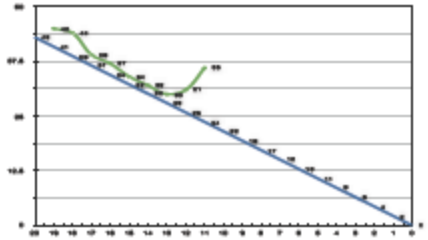
Description: An Orion's Orbits user will be able to pay for their bookings online.

User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a special package with extras online.			
Title: Pay online Description: An Orion's Orbits user will be able to pay for their bookings online.	And add user stories		Next
Title: Show Current Deals Description: The website will show current deals to Orion's Orbits users.	one per swim lane		
			Completed

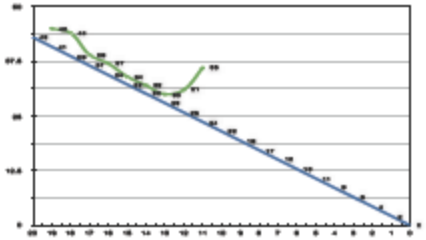
User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to			
Title: Pay online Description: An Orion's			Next
Title: Show Current Deals Description: The website			
	Now add tasks each with a description and estimate		Completed

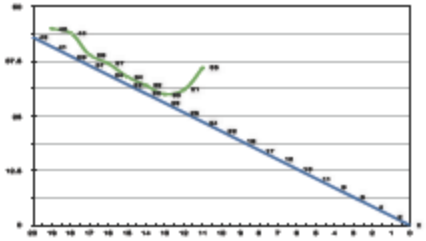
User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a spe[redacted] extras onlin[redacted]			
Title: Pay online Description: An Orion's C[redacted] bookings or[redacted]			Next
Title: Show Current Deals Description: The website w[redacted] deals to C[redacted] Orb[redacted]rs.			
	Tasks that a developer is working on move to "In Progress"		Completed

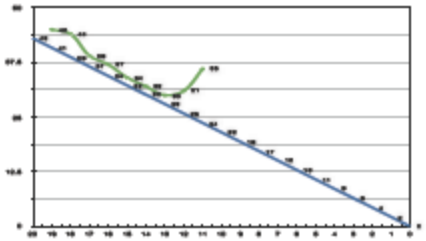
Don't forget to add developer name on the task

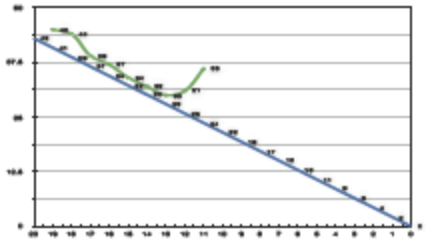
User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a spe… extras onlin…			
Title: Pay online Description: An Orion's C… …ay fo…			Next
Title: Show Current Deals Description: The website w… …deals to C… Orb… …rs.			
			Completed

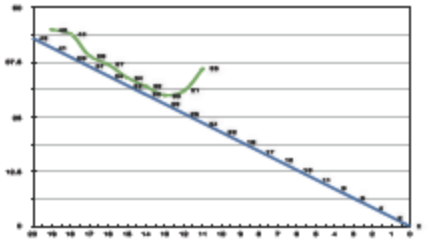
If you stop working on a task, it goes back to its user story

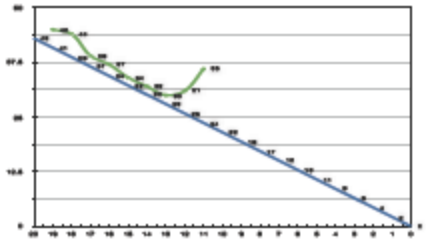
User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a spe[redacted]ckage with extras onlin[redacted]			
Title: Pay online Description: An Orion's O[redacted]er will be al[redacted]ay fo[redacted]bookings or[redacted]			Next
Title: Show Current Deals Description: The website w[redacted]r co[redacted]te C[redacted]Orb[redacted]rs.			
	Tasks that get finished move to the "Complete" section of the swim lane and more tasks get started		Completed

User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a special package with extras online.			
Title: Pay online Description: An Orion's [illegible] user will be [illegible] bookings or [illegible]			Next
Title: Show Current Deals Description: The website [illegible]			
	Be sure to update the Burn Down chart as you make (or don't make) progress		Completed

User Stories	In Progress	Complete	Burn Down
Title: Book package Description: An Orion's Orbits user will be able to book a special package with extras online.			
Title: Pay online Description: An Orion's Orbits user will be able to pay for their bookings online.			Next
Title: Show Current Deals Description: The website			
	Its okay to be working on more than one task for a single user story		Completed

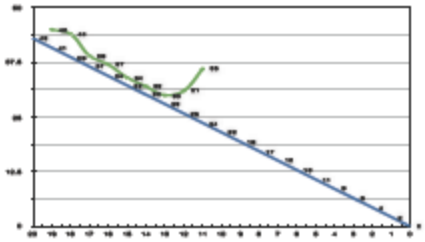
User Stories	In Progress	Complete	Burn Down
			
Title: Pay online Description: An Orion's Orbits user will be able to pay for their bookings online.			Next
Title: Show Current Deals Description: The website			
	Eventually all tasks for a user story are complete; the whole story moves to the Completed section		Completed Title: Book package Description: An Orion's

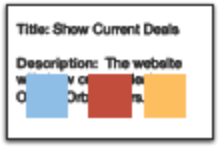
User Stories	In Progress	Complete	Burn Down
Title: Pay online Description: An Orion's Orbits user will be able to pay for their bookings online.			
			Next Title: Show Current Deals Description: The website
	If a user story gets bumped (for whatever reason); move it to the Next section		Completed Title: Book package Description: An Orion's

User Stories	In Progress	Complete	Burn Down
			
		Title: Pay online Description: An Orion's	Next Title: Show Current Deals Description: The website
	Keep working until all stories are complete or have been pushed to the next iteration		Completed Title: Book package Description: An Orion's

User Stories	In Progress	Complete	Burn Down
			Next
			Completed

Don't forget to update the Burn Down chart with your final status





Your time !!

- *Exercise 8 : Correct your big board*

Standup Meeting

- *A daily meeting used to*
  - keep the team motivated and aware of progress (or not)
  - keep your board up-to-date
  - highlight problems early
- *It should*
  - Track progress, update burn-down rate, update tasks,
  - discuss what happened yesterday and plan today's activities,
  - bring up issues, and last between 5 and 15 minutes
- *"Its so short, no one has time to sit down"*

Key Points

- *Factors that weigh into making an initial project estimate*
  - Number of team members
  - Team Velocity
- *Mythical Man-Month*
- *Burn down charts*
- *The Big board*
- *Stand up meeting*