# Agile Planning

Dealing with Reality

# Reality



WHY DOES REALITY KEEP MESSING WITH MY GANTT CHART !?!
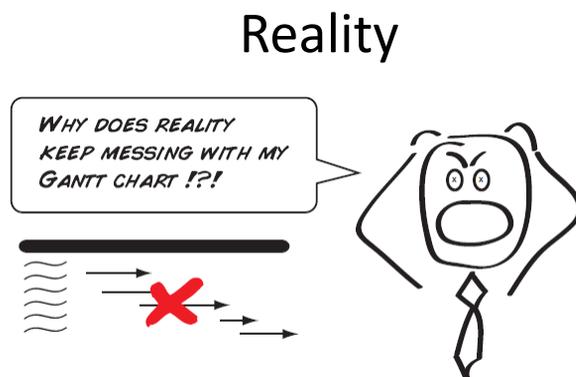
- Basic agile principle – don't expect static plans to hold, be flexible and expect changes

# Examples of the Unexpected

- Everything seems to be going well until…
- Crucial teammate drops the class
- Realization that you aren't going as fast as you thought
- You learn some new features that must be added halfway through
- You run out of time

# Agile Planning

- These problems are addressed by measuring the speed the team is working (the project velocity) and estimating when production-ready code will be delivered

# Iterations

How much we have to do

Master story list
Add user
Print itinerary
Cancel trip
Book permit
Update permit
Search
Create device
Add swap trade
Add option
Cancel plan

How fast we are going

Team velocity

When we expect to be done
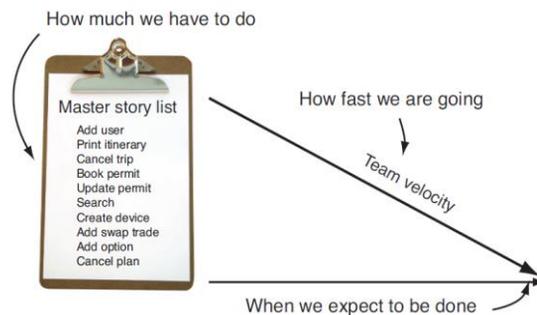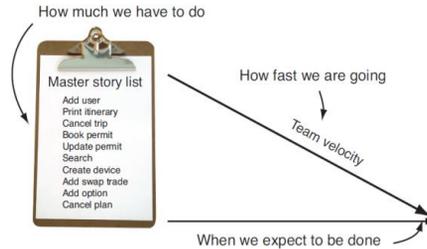
Number of iterations =  Number Total Units / Estimated Units Per Iteration
e.g.   100 / 10  = 10 iterations to completion

Pick iterations to deliver "MMF" – Minimal Marketable Features

Can deliver by core feature set or by date

We are approaching this one iteration at a time partly due to our fixed date at the end of the semester

# Iterations

- Important to convey our estimates are guesses not hard delivery dates

Effort remaining

Slower than expected

Faster than expected

Time
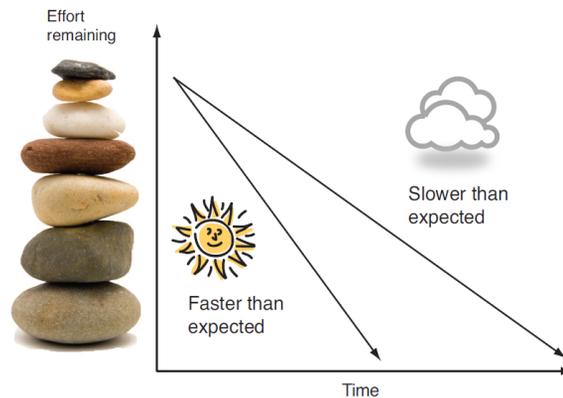
# Agile is flexible around scope

- To keep deadlines agile development tends toward tradeoffs around scope
- What do you do when the customer wants to add a new user story to your iteration?
- What is your team's reaction when the customer wants to add new features late in the development process?

# Changing Requirements

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

- Get customers away from the mindset that they have to add everything in the requirements phase
- Everyone can learn as you go instead of trying to get everything perfect up front

# Could extend the date
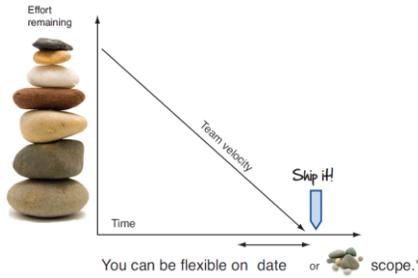


- But customer can't add something and not expect something to come off for the date to remain the same

What if your customer refuses to give up on scope and wants the same date?

# Why the client prioritizes stories



**The #1 Source of Waste on Projects**

Did you know that 64 percent of features are seldom or never used? It's true!*

- Often 13%
- Always 7%
- Never used 45%
- Sometimes 16%
- Rarely 19%

Breakdown of features typically used on software projects
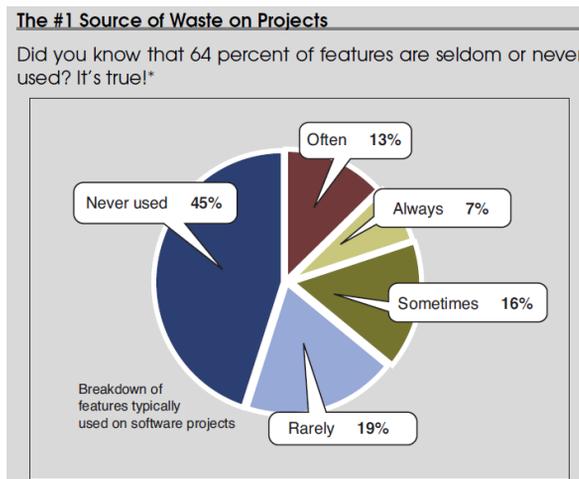
Reality ends up giving us requirements scrubbing

# Team Velocity
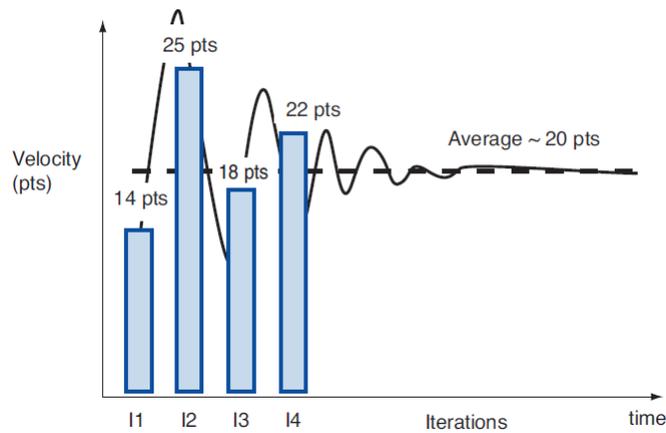


Start conservative; remember we are delivering a production-level story

# Burn-down chart

- We've been using a simple version to show our stories and delivery date
- Shows at a glance
  - How much work has been done
  - How much work remains
  - The team's velocity
  - Our expected completion date

# Burn-down chart
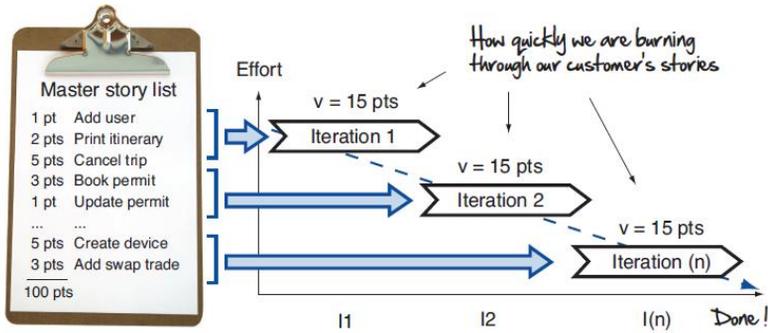


# Burn-down chart
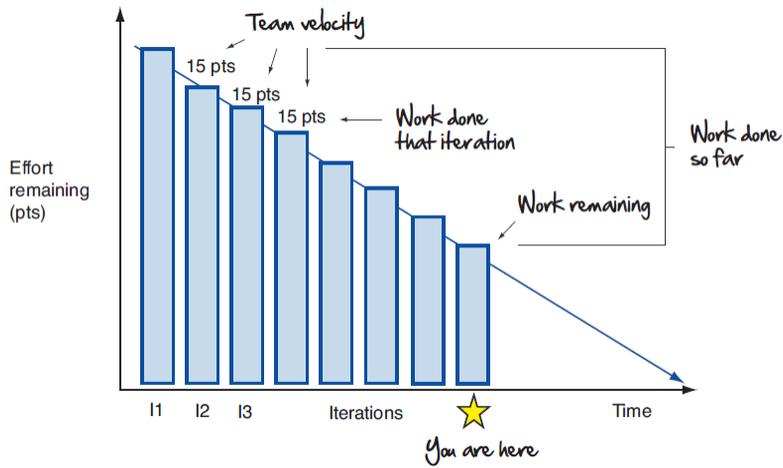
# More Realistic Burn-down chart
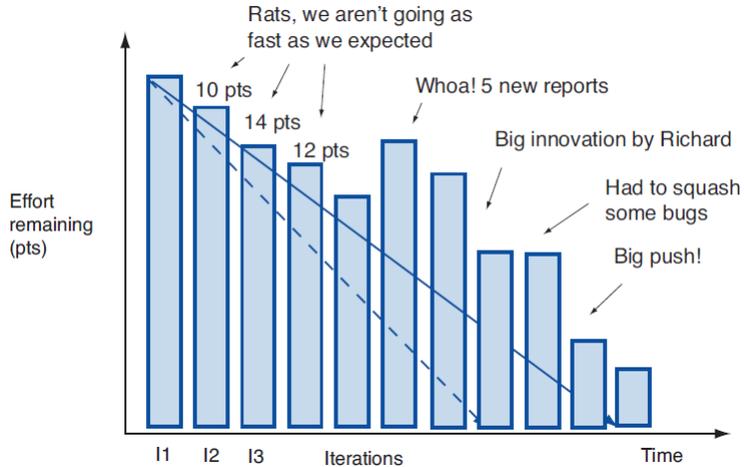


Tells story about what happened, makes events visible and see cause/effect
Honest and open with customers and developers

# Burn-up chart
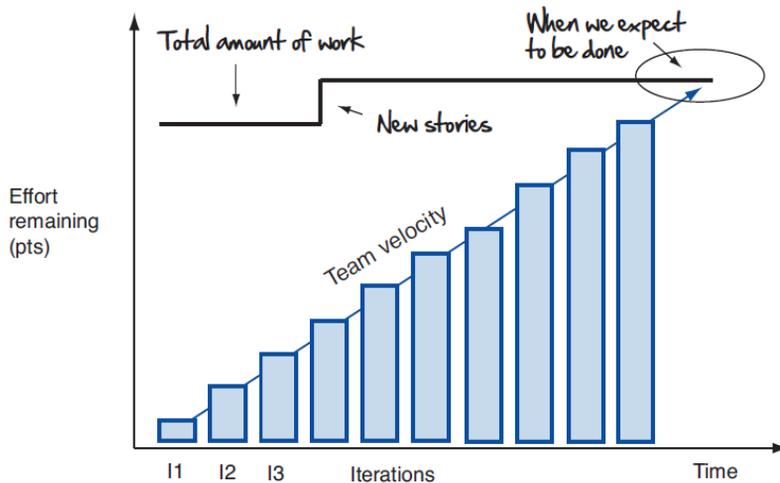


Shows added work of new user stories

# Some Scenarios

**Scenario #1: Your Customer Discovers Some New Requirements**

What do you do?



Effort remaining

← New features discovered

Iterations

The impact
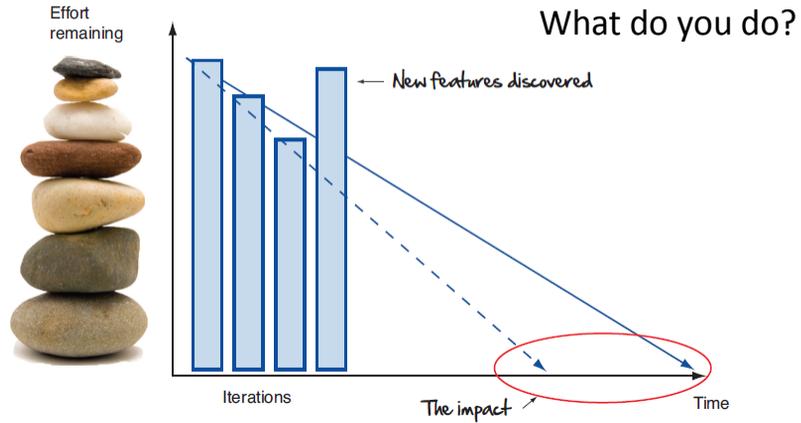
Time

**Scenario #2: You Aren't Going as Fast as You'd Hoped**



Effort remaining

Actual velocity

Estimated velocity
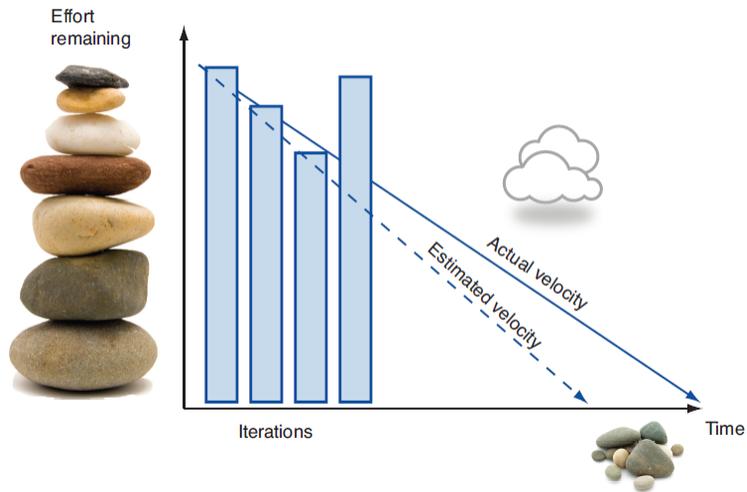
Iterations

Time

# Spartan Warrior strategy for checking reality of deliverable date
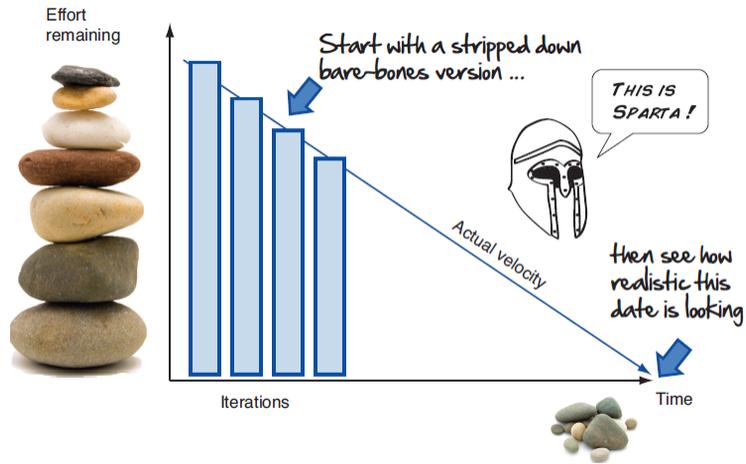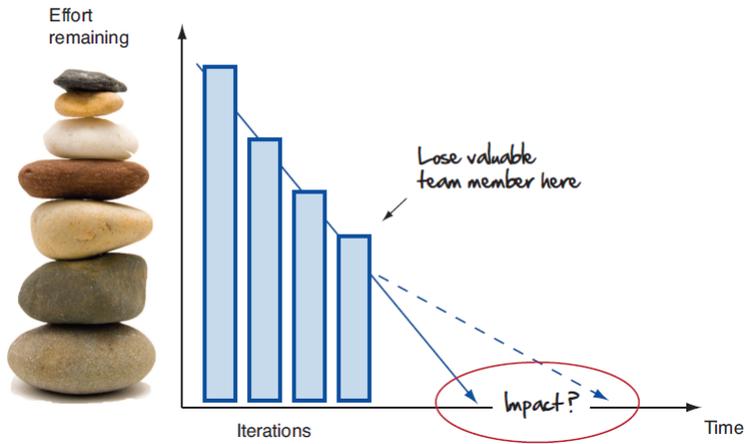
Add new team member?

**Scenario #4: You Run Out of Time**   (due date pushed earlier)
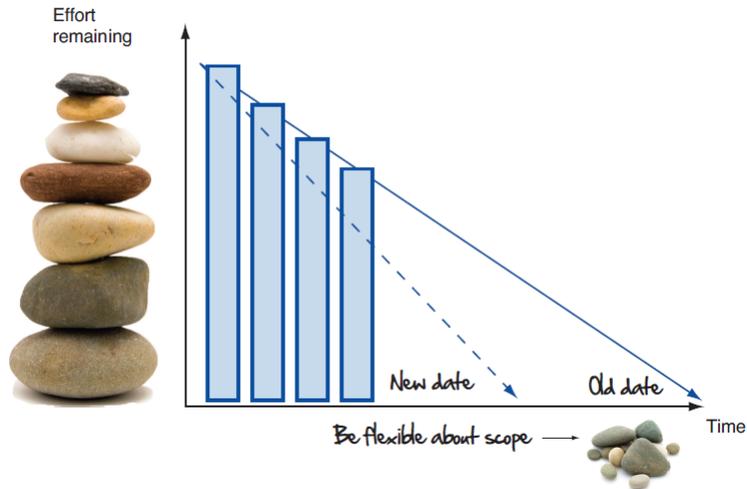


# Focusing on an Agile Iteration

- Goal is to produce production software for user stories within a short iteration but we have more to do than just coding
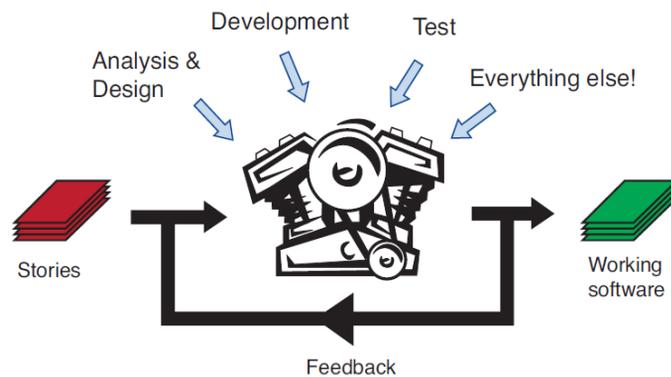
# Analysis

*Do just enough analysis for what you need*



Index card & conversation

One pager

Something heavier

Start light ...

and add weight only when necessary.
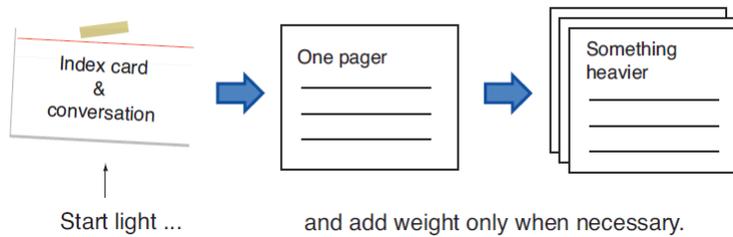
# Sample One Pager

**Story name: Create work permit**

### Description

Before contractors can legally work on the construction site, they need a work permit. This permit is what they will take to the job site when they are ready to begin construction.

### Tasks

1. Create permit page.
2. Save permit to database.
3. Add basic validation.
4. Ignore security (for now).

### Test criteria

1. Requestor can save basic permit.
2. Permit gets saved to the database.
3. Invalid permits are rejected.
4. Permit defaults to next week's start date.

# Just In Time Analysis

Do your analysis here ...

Iteration (n)

for stories you want
to develop here

Iteration (n+1)
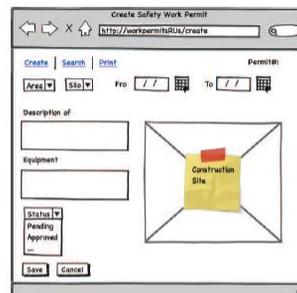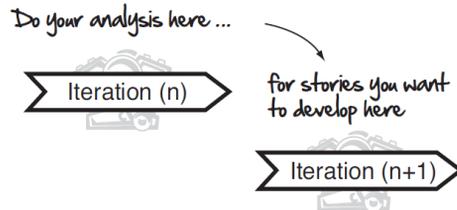
- Analysis done with latest information
- Everyone can learn as you go
- Avoid rework by analysis too early

# Sample Analysis Artifacts - Flowchart

Create/update permit
24 hrs in advance

Approve
?

Y

Worker comes to trailer
to pick up permit

N

Reject permit
Record reason
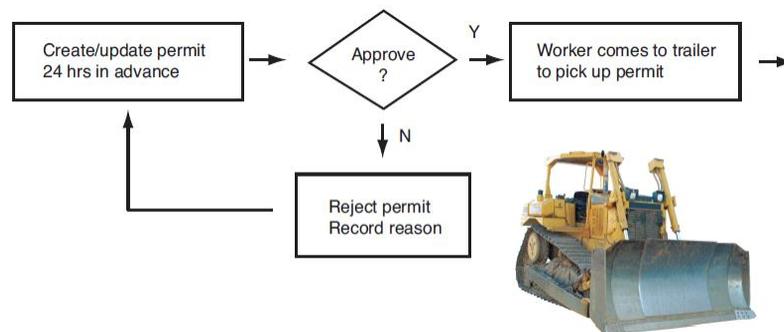
# Sample Analysis Artifacts - Personas

**Administrator**

*"Amanda"*

Needs to be able to add and remove users to the system.

Is comfortable with computers.

Runs the office (all permits are distributed through her for new construction workers).

**Requestor**

*"Robert"*

Construction manager or engineer who will request permits on behalf of his/her employees.

WIll know details about the work.

Responsible for ensuring permits are requested on time.

**Approver**

*"Mr. Kelly"*

Safety and loss management officer responsible for overall safety at construction site.

Must approve any permits before being issued.

Final word on validity of permit.

# Analysis Artifacts – Paper Prototypes

Try different designs fast using paper prototypes

Option #1

Option #2

Option #3
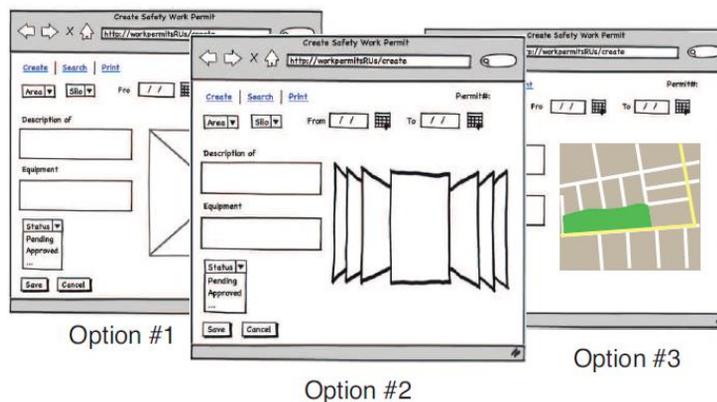
# Analysis Artifacts – Acceptance Tests

How do we know if this thing is working?

Create work permit from Requestor interface, should be visible from Admin and Approver interfaces

Create work permit, save, identical permit should be retrievable after logout and login

….

# Development and Testing

- We'll have more to say about this later
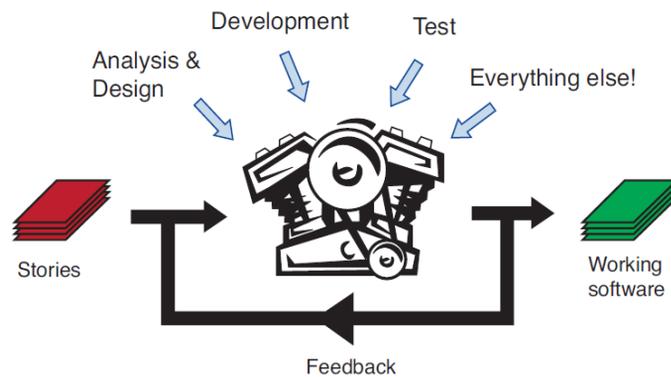
# Other Iteration Activities

- Make sure the work for the next iteration is ready (user story planning meeting)
- Get feedback on the last iteration's stories (showcase with customer)
- Plan for actually doing the work for next iteration (iteration planning)
- Continuously look for areas of improvement (mini retrospective)

# Story Planning Meeting

- Is the next batch of user stories ready to go?
- Have we done our homework?

# Showcase

- Demo project and features completed
- Make it fun
- Get feedback

# Iteration Planning Meeting

Set next week's expectations!

IPM

Showcase

**Plan next iteration**

Mini-retrospective

Iteration (n)

Check the health of the project
Review the team's velocity
Sign up for stories and commit

Use burn-down chart
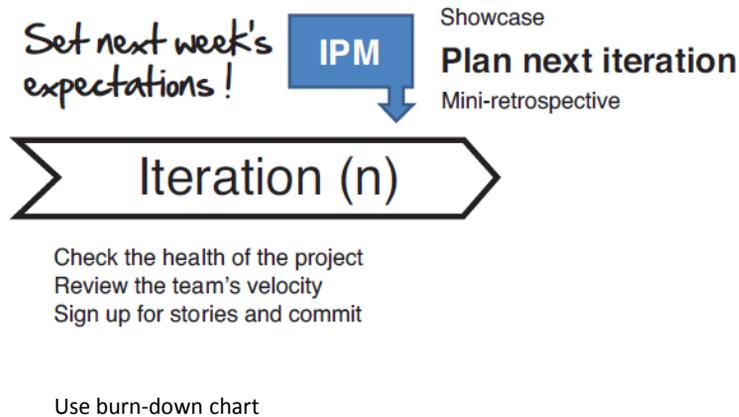
# Mini Retrospective

In 10-15 minutes, where are you lagging and where are you kicking butt?

Agile principle

At regular intervals, the team reflects on how to become more effective and then tunes and adjusts its behavior accordingly.

**The retrospective prime directive**

Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.

In other words it's not a witch hunt.

# Constructive Feedback

- Avoid pure criticism and offer a way to improve performance
- Cold
  - "Suzy, I noticed you did some great work on the print module last iteration, but your unit tests were really lacking."
- A little sweetness
  - "Suzy, awesome work on the print module. Apply that same level of detail to your unit tests, and you are soon going to be world-class."
- Objective
  - "Suzy, the print module runs great and the code is nicely organized. The unit tests didn't cover all the functionality though and should check for error conditions as well."

# Do whatever works for you

- All in one meeting or separate meetings?

- Some activities conducted online?

- Story planning meeting not needed because it happens naturally through regular interactions?

- Just make sure that each iteration you show your customer working software, set expectations for the next iteration, and look for ways to improve