

# Agile Expectations

Walter Bodwell  
Planigle



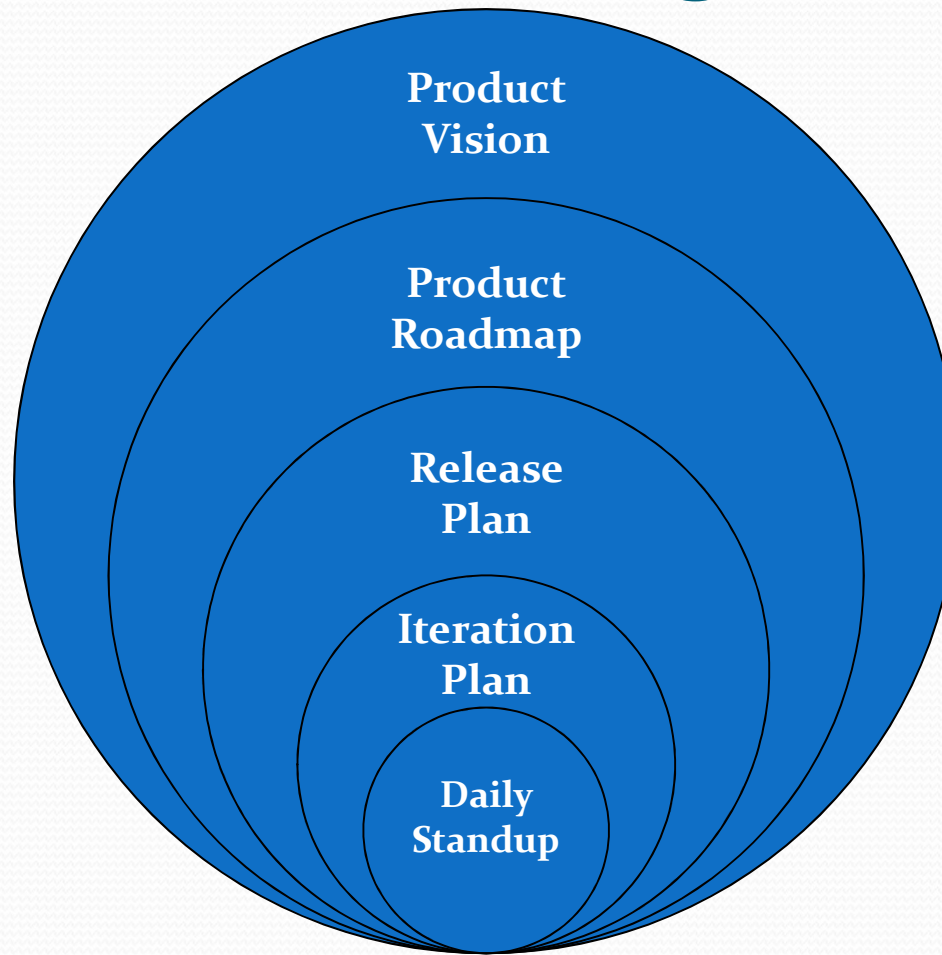
# An Introduction – Walter Bodwell

- First did agile at a startup in 1999
- Got acquired by BMC in 2000 and spent the next 8 years doing agile at scale
- Now providing consulting, tools and training to help teams get the most out of agile at Planigle



# 5 Levels of Planning

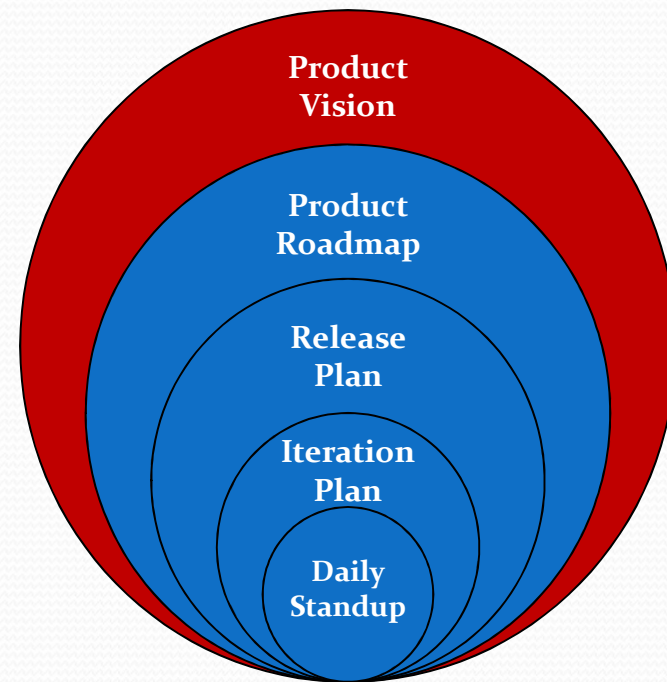
Adapted from “5 Levels of Agile Planning” by Hubert Smits





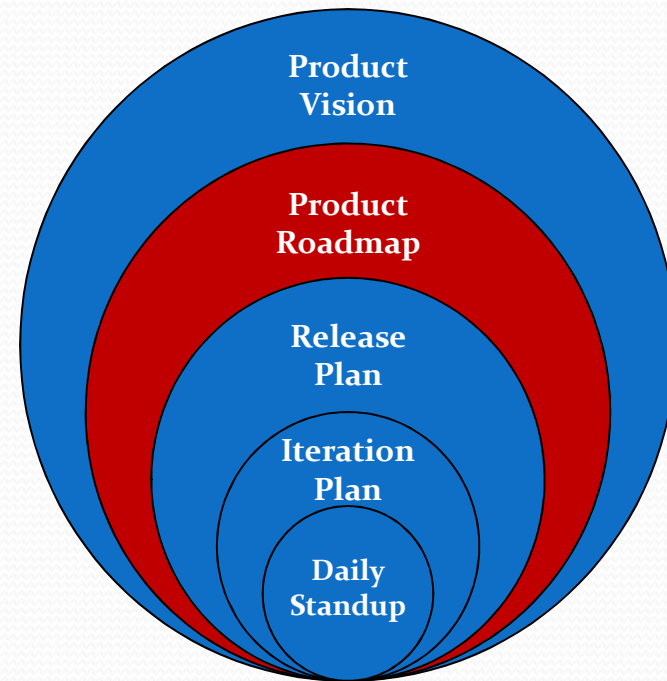
# Product Vision

- What are you trying to accomplish?
- How is that going to benefit the business?



# Product Roadmap

- High level themes for the next few releases
- Shows progress towards strategy
- Lots of “wiggle room”

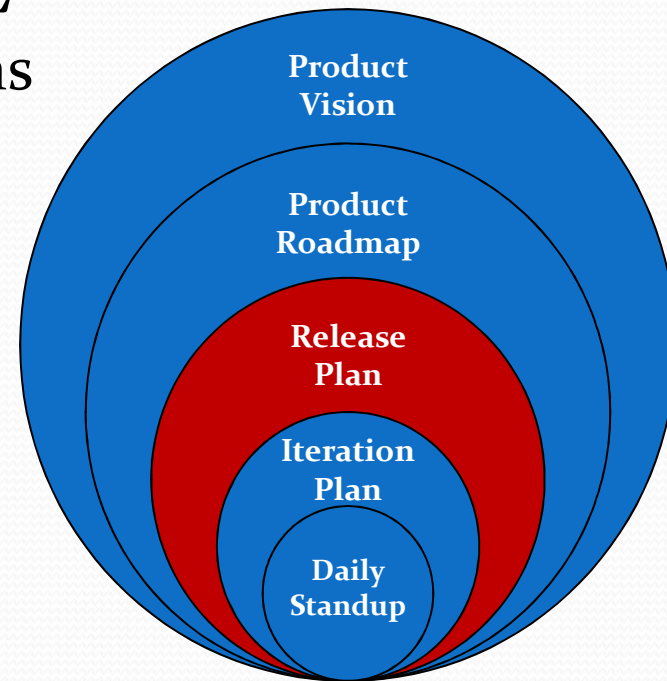




# Release Plan

- Go into next level of detail towards themes
- Get everyone on the same page
- Understand what you will likely achieve
- Balance load between the teams
- A projection,

**not a commitment**



# Timing

- Release planning should occur just before the teams start on the release
- If teams are just forming, delay until after the teams have done 2 or 3 iterations (so that they understand their velocity)
- If the release is long (over 4-6 months), release planning might occur multiple times to resynch
  - High fidelity for close items
  - Low fidelity for items further out





# Preparing for Release Planning

- Set themes
- Prepare the backlog
- Identify teams
- Divvy stories up
- Understand the issues
- Size the stories
- Define what done is
- Identify key dates





# Set Themes / Investment Areas

- What are the areas we're going to invest in for this release?
- Themes give you room to be flexible
  - We know we're going to do something
  - We'll decide as we go how much
- Themes are a great way to communicate the focus of the release without prematurely committing you to details



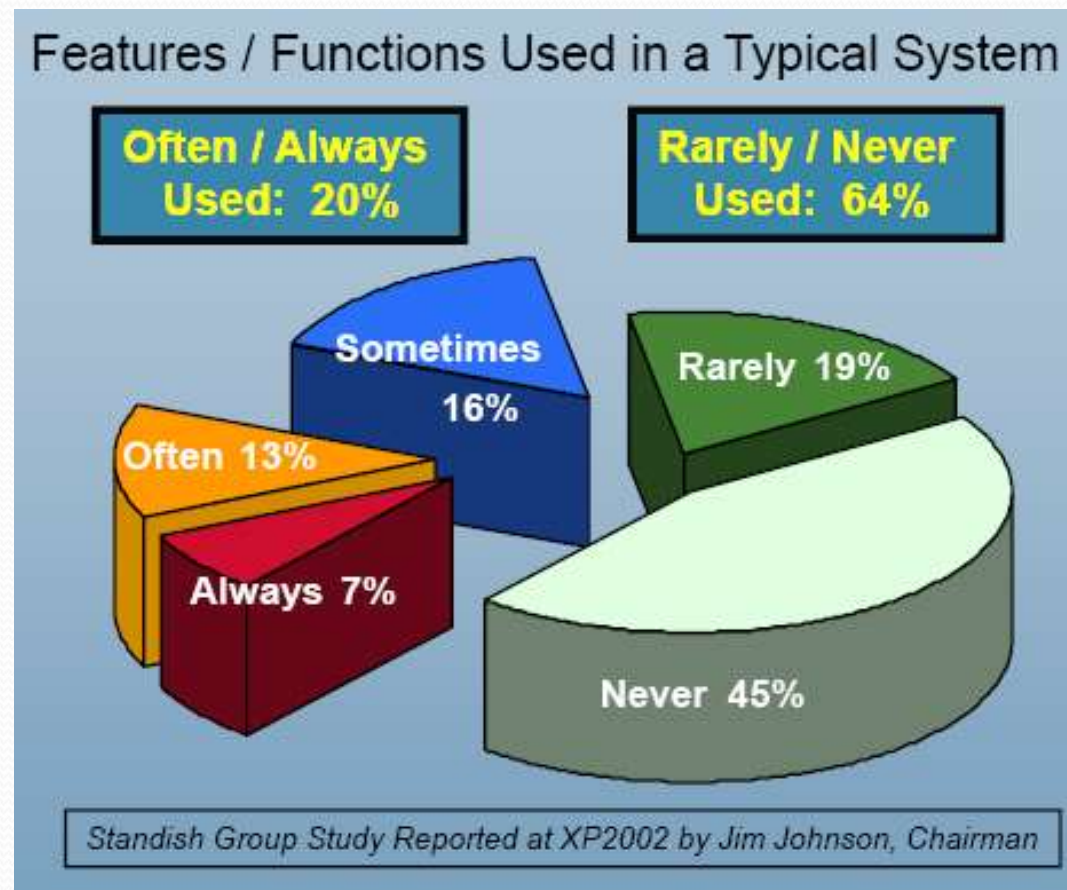
# The Backlog

- Rank order things
- Each should meet I.N.V.E.S.T. criteria
- Size will vary based on how far out





# Why Prioritize?



# Identifying Teams

- It is preferable to have each team have the ability to complete its work by itself
- In other words, instead of a team per component, have teams with members who have knowledge of each component that will need to change to deliver something
- Don't change the teams frequently





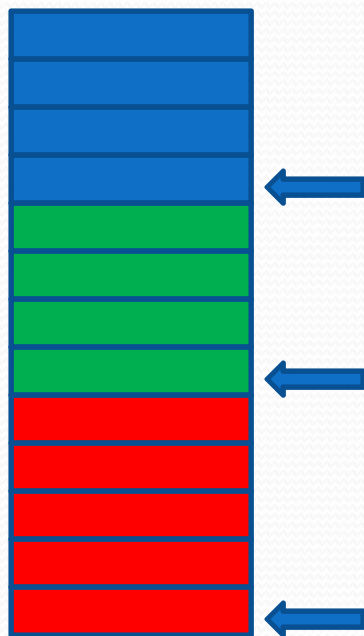
# Components or Features?

- Component teams provide expertise
- But limit the ability to attack the highest priorities
- And require more coordination (complexity)



# Divvying Things Up

- Your goal is to divvy things up so that teams are working on items of around the same priority



Bad

If each team able to get 3 blocks in the release, the highlighted stories won't make it



Good

By better distributing stories amongst the teams, look which stories won't make it



# Dependencies

- Approaches (go with the first one you can):
  - Structure the teams so that a single team can solve the problem end to end
  - Do the work within the same iteration
  - Implement the service and then use it
  - Stub out the service and implement it later
- Make sure the dependent teams (including external ones) are represented at release planning



# Understanding the Issues

- Keep to the minimal responsible amount of doc
- No more than you need at any point in time
  - Just enough to understand dependencies and mitigate risks
  - The right size for the problem
- Do you need this now?
  - If not, try to reduce or eliminate it





# Agile Architecture

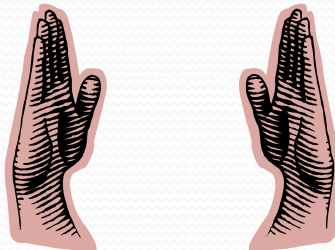
- Just enough
- Refactor as you go





# Estimating

- Identify a medium sized story that is well understood; call it a 5
- Now estimate other stories relative to that
- Is it about the same,  $\frac{1}{2}$  as difficult, twice as difficult?
- Use Fibonacci numbers: 1, 2, 3, 5, 8, 13, 21
- If bigger than that or if too hard to estimate, split the story





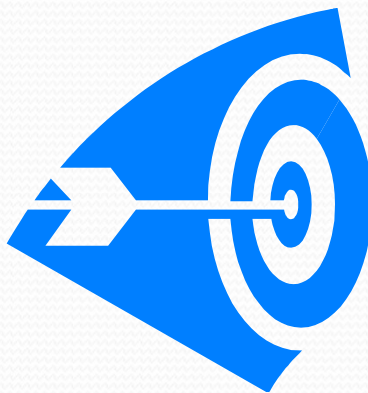
# Why Story Points?

- Time estimates
  - Vary by person
  - Encourage padding
  - Tend to grow stale
- Story points
  - More consistent from person to person
  - Not a commitment to time frame
  - Don't change as much
  - Easier to estimate relative size



# Velocity

- Now that stories have sizes, you can track how many points you typically get done in an iteration
- You can now use this to predict future completion rates





# Story Points Across Teams

- To get teams in the same ballpark, pick a baseline story
  - Each team should understand the complexity
  - Choose a medium size story
  - Call it a '5'
  - All other stories are relative to it
- Don't compare velocity
  - Used by a team to evaluate itself
  - If others use it for evaluation, it will be gamed and become useless



# Definition of Done

- Define what “Done” means for your team
- Make “Done” more stringent over time
- Definition of Done evolves as you do





# Key Dates

- How many iterations?
- When do they start / stop?
- Which iterations (if any) are hardening?
- Any other dates to be mindful of?



# Release Planning

- Kick off / Overview
- Break Out Sessions
- Review Results





# Attendees

- Product Owners
- ScrumMasters
- Architects / Leads
- QA
- Writers
- Other Stakeholders



**This is the best time to travel!**

# Release Planning Deliverables

- Plan for each Iteration
- Assumptions
- Dependencies
- Risks





# Philosophies



- Simple Is Better
  - Allows for better coverage across features
  - Prevent unnecessary complexity
    - If you go too simple, it will remain unused until you follow up with the market demanded features next release
    - If you go too complex, it is very difficult to identify and remove the unnecessary complexity
- Leave Room To Be Agile
  - No more than 70% of our time should be allocated to committed features

# Release Planning Wrap Up






- Go through each iteration for each team
- Are things synched up across teams?
- Are you attacking the most important stories?
- Does the team believe in the results?





# Is This Reasonable?



- Everyone agrees the release is doable
- Use disagreement and uneasiness in team members to drive out hidden risks, tasks, and issues
- Drive agreement with a fist to five
  -  • Absolutely, no question
  -  • I think this is good and will make it happen
  -  • I can support this
  -  • I'm uneasy about this and think we need to talk it out some more
  -  • Let's continue discussing this idea in the parking lot

# After The Meeting

- Capture the results in your tool of choice
- Update after each iteration
- Notify of major changes to the plan



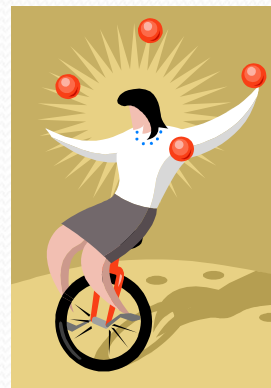


# Communicating the Future

- Themes give you room to be flexible
- If a customer is asking about a particular feature, you can get into a discussion of priorities
- Demos are a potential opportunity to get a customer involved
- Smaller, incremental releases generate feedback on what to dig into in more detail

# Prioritization Doesn't Stop

- The product owner re-prioritizes after each iteration
  - We've learned more about the business
  - Let's take advantage of that
- The further down the list something is, the less defined it will be and the less important it is to prioritize precisely





# Questions?

Walter Bodwell  
Planigle

[wbodwell@planigle.com](mailto:wbodwell@planigle.com)

Twitter: @wbodwell

[www.planigle.com](http://www.planigle.com)

[www.walterbodwell.com](http://www.walterbodwell.com)

