Engineering Processes

"Whether you think you can or think you can't - you are right."--Henry Ford

- There are a lot of opinions on "good" processes
- Any of them can be backed up with success stories
- Go with the one that fits your project's needs

Source: http://moria.whyayh.com/cgi-bin/viewvc/public.cvs/app/eng-methods/

My Experience

- 18 years at HP in 5 R&D Labs – HP 150 PC
- 5 startups in the last 12 years – one was Napster
- Current title: Release/Operations Manager at TrustedID (an Identity Theft Protection company)
- Release job role: manage QA, manage 70 Dev and QA virtual servers, build packages, push code changes to production
- Operations job role: manage 25 production servers, and architect the migration to AWS.

FLURPS

- Functional – works as designed
- Localizable – languages and formatting
- Usable – users want to use it, even with “defects”
- Reliable – you don't lose data
- Performance – good enough to meet Functional req.
- Supportable – problems can be reproduced for any version.

Product Lifecycles

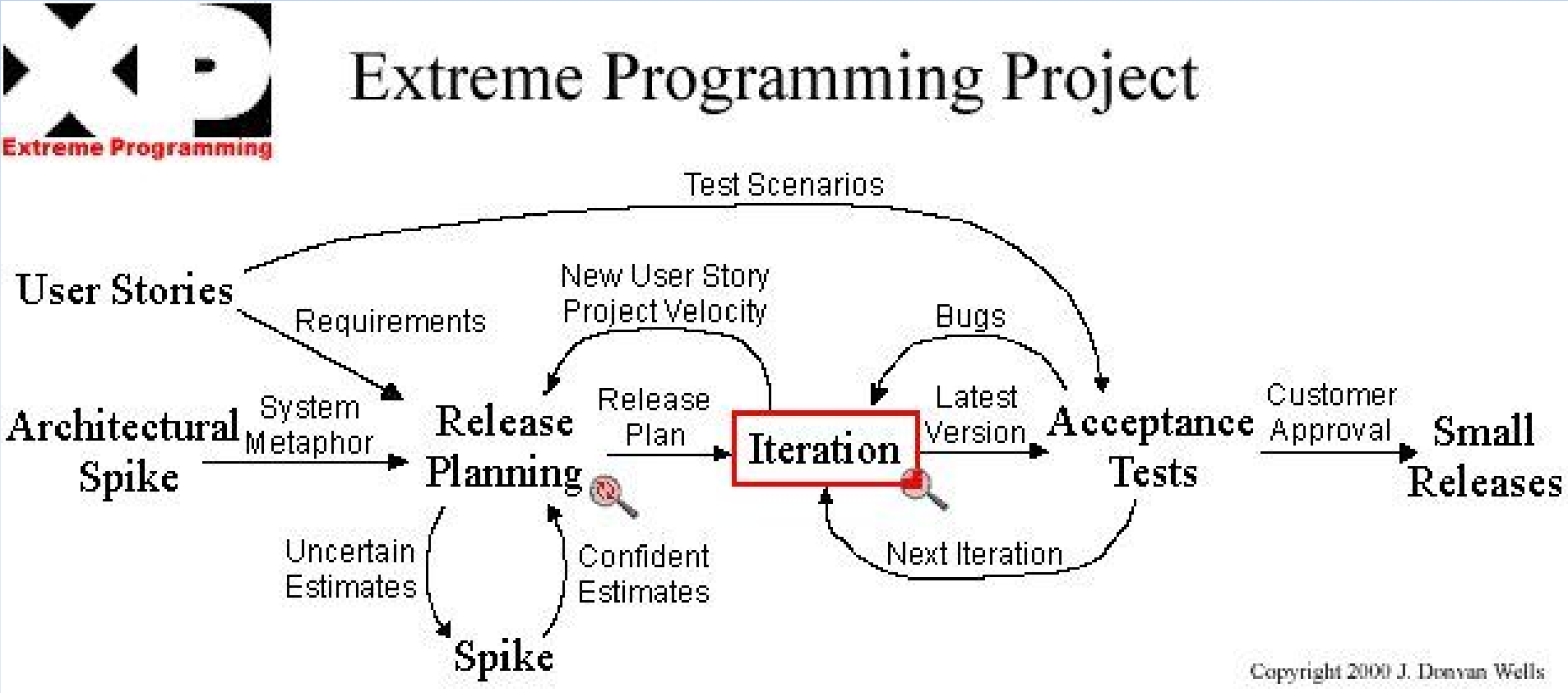
"When all is said and done, more usually gets said than done."--unknown

- Waterfall – elements of this are in all lifecycles
- Agile – these are the preferred lifecycles

Waterfall Lifecycle

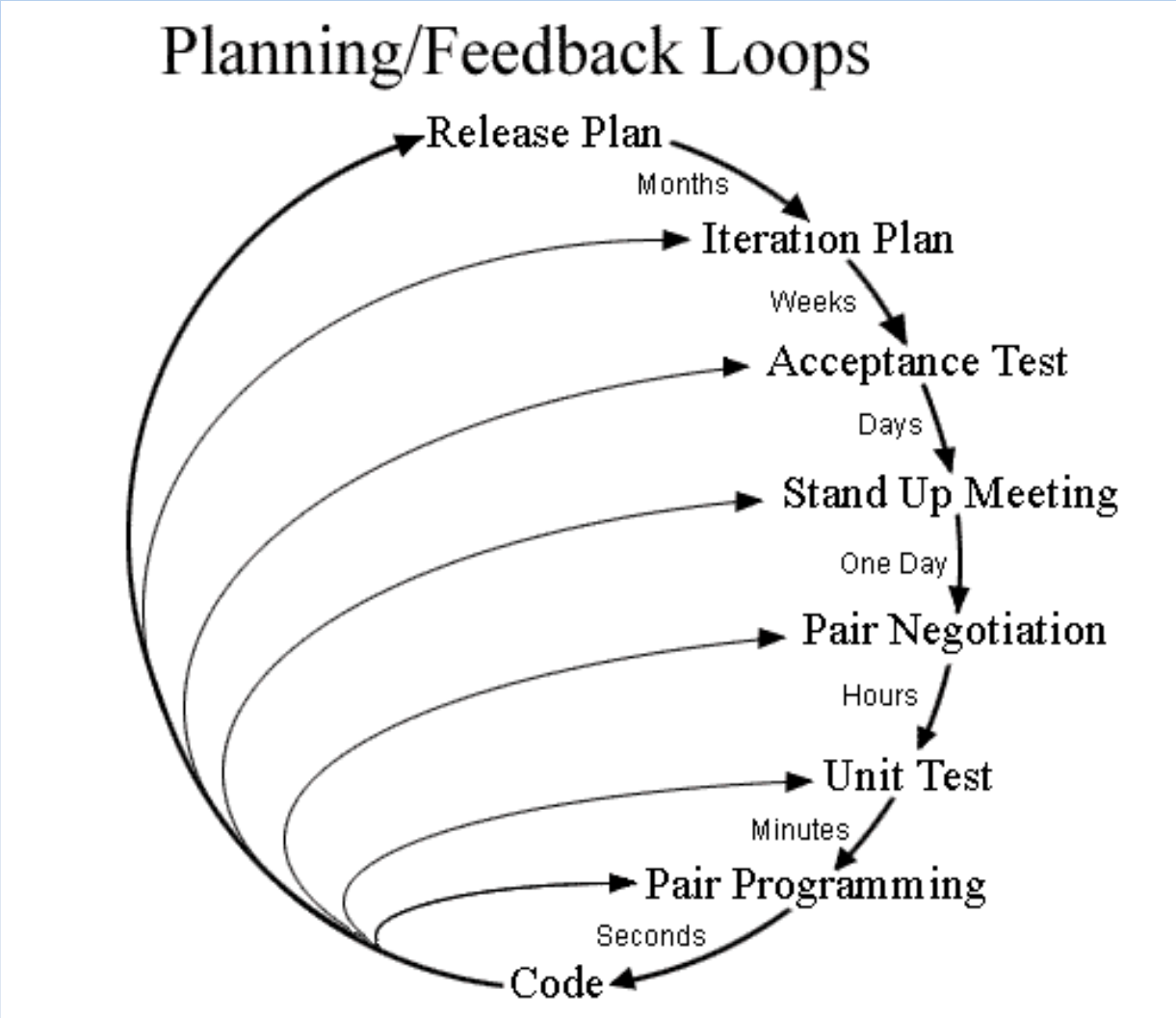
- Idea Phase
  - Proposal
  - Brainstorm
  - Prototype
- Design
  - Project Plan
  - External Design
  - Internal Design
- Implement
  - Create Modules
  - Dev. Integration
  - System Integration
- Alpha Test
- Beta Test
- Production Maintenance
- End of Life

Agile Lifecycles



Source: http://www.extremeprogramming.org

Agile Iterations



Source:
http://en.wikipedia.org/wiki/Extreme_programming

FLURPS Application

- Functional – yes, must meet the competition requirements
- Localizable – no
- Usable – most club members can run it
- Reliable – enough to handle likely problems
- Performance – only enough to meet requirements
- Supportable – most club members can fix it

Lifecycle Application (1)

- Brainstorm overall approach
- Breakup the project into prototype modules
- Repeat for each module (HW/SW)
  - Brainstorm
  - Pick the ones to try
  - Prototype until success or failure is “known”
  - Assemble with other modules?

Lifecycle Application (2)

- Improve on the assembled prototype(s)
- Repeat from the beginning, until the FLURPS requirements are met (i.e. evaluate the overall product and project process)

The baby duck syndrome

- Baby ducks imprint on the first moving thing they see
- Engineers often hang on to their first designs or first things built
- SW example: unwilling to rename things
- Solution: when enough things become inconsistent or unworkable, it is time to refactor or redesign

Prototyping

- Plan each prototype like a mini-project
- Make it Functional first
- Make it Robust next – risk analysis mitigations
- Make it “pretty” last (if manufacturing it)
- Use the cheapest and good-enough materials, don't over build
- Working (even ugly) prototypes should always be favored over beautiful unproven simulations

  (“Show it in code!” – IETF)

What about CAD?

- It is not required in a prototype phase of a project
- It is only required for manufaturing products with other defined pieces
- It can be a fun simulation tool, but paper drawings are faster for one-off designs
- If used in the prototype phase, limit it to small parts of the project, or as a rough picture of the assembled parts – don't get sucked into unneeded precision

Error Handling

- Design it in at the beginning – Murphy's Law will bite you, if you don't
- For SW there are structured ways of doing error handling so that your functional coded is still readable (see the reference in the footer)
- Planning for what could go wrong, is NOT “planning for failure”–it is called “risk analysis”-- Mitigate for likely and costly problems

- ”A transaction approach to error handling”

  Source: http://moria.whyayh.com/work/error-handling/