The Next Iteration

Kenneth M. Anderson
University of Colorado, Boulder

CSCI 5828 — Lecture 25 — 04/14/2009

Goals

- Review material from Chapter 10 of Pilone & Miles
  - The Next Iteration
    - Planning
      - Recalculating Velocity
      - Talking with your Customer
    - Dealing with Change
      - Using Third Party Code

The Next Iteration

- You have work to do to get ready for the next iteration
  - Indeed, given the material covered in Lecture 24 and the issues we'll discuss today
    - a prudent team will allocate at least a day, if not two,
    - to all the tasks that need to be performed at the end of an iteration
- After your iteration review is complete,
  - your biggest job is planning for the next iteration

Planning

- Elements of Planning
  - How much work is left over from the previous iteration?
  - Is the customer satisfied with the work accomplished?
  - What bugs have been found by the testing team?
  - What new stories were (previously) planned for this iteration?
  - Does the customer have anything new for you?

First Step: Estimates

- Your first task is to revisit all prior estimates of all user stories (both existing and new)
  - With at least one iteration under your belt, you are
    - better prepared to decompose stories into tasks
    - better prepared to play planning poker and assign estimates
- You may have spent a lot of time in your first iteration performing set-up and configuration tasks
  - various aspects of the system will now be in place and that will allow you to reduce some of your prior estimates

Second Step: Velocity

- You now need to recalculate your velocity
  - If
    - estimated work = developers x working days x old velocity
  - then
    - new velocity = actual work / (developers x working days)

Example

- Back in chapter 3, we calculated
  - 3 devs x 20 days per iteration x 0.7 velocity = 42 days
  - That was 42 days of potential work per iteration
- In actuality, our team managed to complete 38 days worth of work during the first iteration
  - 38 days actual work / (3 devs x 20 days) = 0.6333
  - New velocity: 0.6 (You can use .6333 if you want)
    - Remember: 0.7 was just a guess
- New estimate for next iteration: 3 x 20 x 0.6 = 36 days

Step 3: Big Board

- Now you need to prepare the big board for the next iteration
  - The book recommends taking a picture of the old board before taking it down (you can then archive the picture)
- All stories and tasks should have new estimates
  - All stories should have priorities that have been reviewed by the customer and updated if needed
- Allocate stories for the next iteration taking into account the new value for estimated work
- Finally, present new plan for iteration to customer

Step 4: Disaster?

- In most cases, the customer will approve the plan and you'll be ready to go when Monday comes around
- Every now and then, the customer will do what customers do best:
  - Change Everything!
- Disaster?
  - NO!
  - Listen to the change requests, create stories and tasks, assign estimates, get priorities, create new plan, then go!

Example

- In the book, the customer springs a request that
  - requires the team to pick up a library developed by another company
  - integrate that software into our system
  - demo the integrated functionality at the end of the next iteration
- The team has to drop everything and re-plan
  - One smart thing they do is to give a large estimate to the task of integrating the new software
    - Smart because: you have to build it, learn it, code to it & test it!

Example, continued

- The team learns the library API, writes code against it, and
  - their system hangs
- As a result, they now have to debug third party code
  - (fortunately they have the source code)
  - If they didn't, they would have to either
    - submit a bug report and hope for a fix
    - drop the library and create its functionality from scratch
- Ultimately, you are responsible for all the code you deliver with your system; trust no one and test extensively

Example, continued

- There are risks but reuse is almost always a GOOD thing
  - Consider, the libraries that come with modern PLs
- Recently I wrote an app that searches twitter for “tweets” that match particular criteria; it took me less than two days
  - Reused the following:
    - Twitter’s Search API
    - io, json, os, sys, time and urllib modules from python
    - list and map data structures, list comprehensions, and functional programming techniques from python

Wrapping Up

- Before moving on to the next iteration
  - create new estimates for all tasks and stories
    - get new priorities from customer
  - calculate velocity based on performance
  - allocate stories based on updated velocity
  - get customer approval
- Welcome change; your process supports it!

Coming Up

- Lecture 26: Alternate approaches to Concurrency
  - No reading assignment
    - MapReduce
    - Agent model of Concurrency
      - Examples from Erlang and Scala
- Lecture 27: Bugs
  - Chapter 11 of Head First Software Development