Swift Sync
from digité

IMPROVING PREDICTABILITY USING KANBAN

3/7/2013

Sudipta Lahiri

Work falls in 2 buckets

Keep the lights on…

- Production Support or Maintenance project
  - Pipeline changes frequently
  - Prioritization changes frequently
- Release planning is done in the following few ways:
  - Either a defined frequency
  - Around some important functionality
- Estimation limited to primarily "key" work items

Projects

- Discretionary or Value Enhancing
- Traditionally, waterfall based
  - "MS Project" like scheduling
    - WBS is made
    - Estimate is made
    - Dependencies are defined
    - As project progresses, MSP keep computing a scheduled date
- Increasingly Agile based

Forecasting…

Keep the lights on…

- Quite non-committal
- Whatever can be "fitted" in the given timeline, goes in the release
- Trust deficit between business and IT or customer and vendor
  - Business/customer always feels that work items are over estimated
    - Extremely difficult to justify every incremental
  - Delivery always feels that customer/business does not appreciate the details
- Estimation basis is past time sheet data
  - This too is error prone and massaged

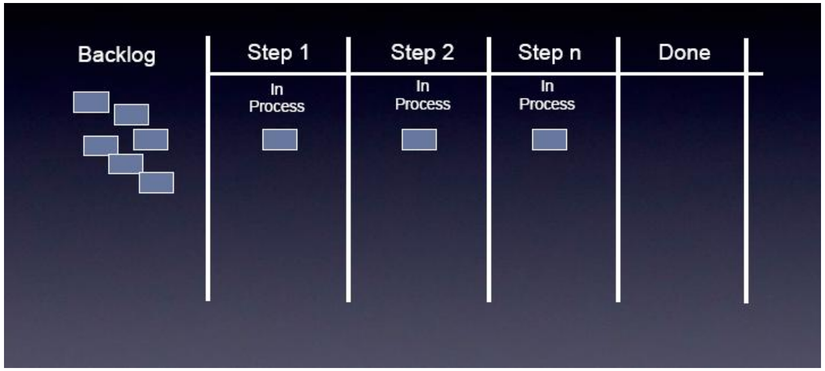
Projects

- Scheduling done with critical path tracking
  - Two key issues:
    - Scheduling is very deterministic
      - Though, it is completely based on "estimates"
    - Trust deficit and issues with estimation continue
- With this background, there is an inherent contradiction between the estimate and the forecast
- Further, as work progresses and projects start slipping, focus is around controlling critical path
  - Limited ability to do scenario based planning

A quick Kanban primer

- Visualize the Work
  - Map your value stream
- Limit Work in Process (WIP)
- Manage Flow; Establish a Cadence
  - Remove bottlenecks and improve the flow
  - Increase throughput
- Make Process Policies Explicit

- Implement Feedback Loops
- Improve Collaboratively, Evolve Experimentally

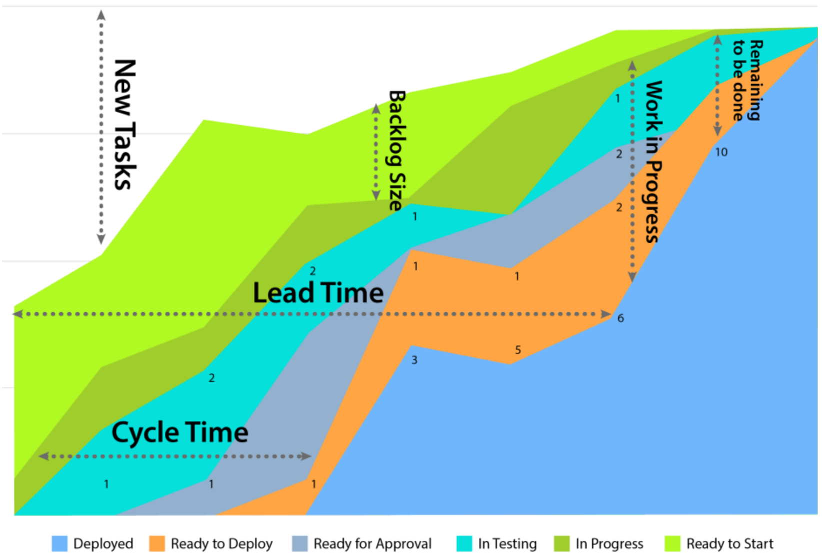


So, how does Kanban help?

- Flowing of cards is at the heart of a Kanban system
- As cards flow, system tracks:
  - How much each card of different type, of each size, spends on each stage of the value stream
    - Sizing is based on a approx T shirt sizing
    - Wait time and blocked time can be excluded by the system
- Focus on low WIP limits discourages people from multitasking
  - “Stop starting; start finishing”
  - Result: calendar time = approx. actual effort

Metrics from a Kanban system

How to Read a Cumulative Flow Diagram



- □ Cycle Time = WIP/Throughput
- □ For a given inventory of work, Cycle Time and Throughput are inversely proportional
  - Will be used interchangeably

Predicting with CFD

Product demo

Question remains....

□ We know what we need to do...

□ But how we go about doing it?

□ Introducing "Simulation with Swift-Kanban"

Simulation with Swift-Kanban

- Basis:
  - Estimates are not accurate
  - Even an accurate estimate fail for multiple reasons!
  - But…
    - We know that variation in effort takes a distribution pattern
- Next:
  - We derive base data for the simulation from “actual” data
    - Not by estimates
    - Actual data based on progress of work on the board
      - Accuracy driven by the rigor of Stand-up call
    - Team/work profile
  - We do make some assumptions…
- Result:
  - Simulation helps you to evaluate different options that impact your throughput/cycle time
    - You can change your team profile and see the impact
    - You can change your working model and see the impact
  - Gives you more concrete information “behind” your decision

Lets do some “what-if” analysis…

Product demo

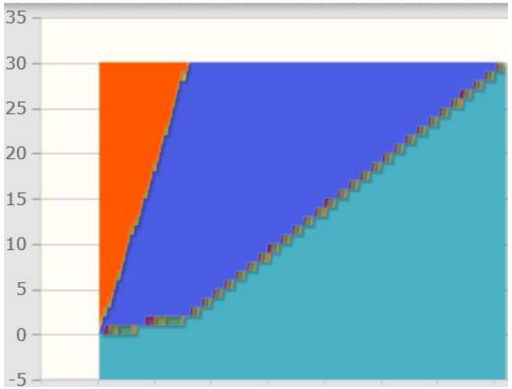
In summary…

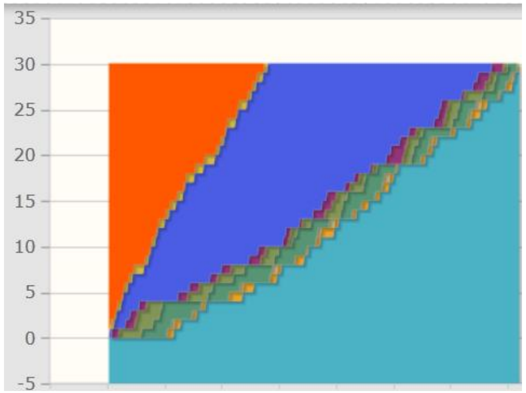
- 🟧 Ready for Development
- 🟨 Design
- 🟦 Coding + Junits
- 🟪 Functional Test Automation
- 🟫 Done
- 🟩 Code Review
- 🟩 Automation Test Review
- 🟨 Dev Complete
- 🟧 Validation
- 🟦 Ready for Deployment

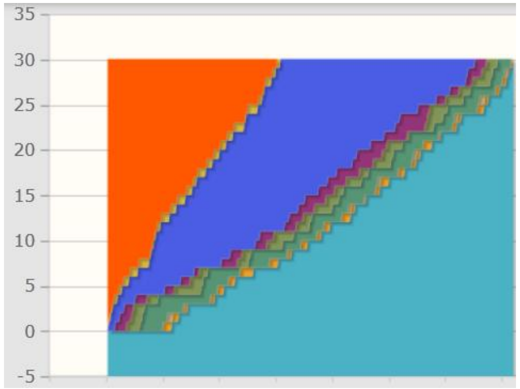
- 30 cards
- With a team of 3:
  - 1 Designer
  - 1 Developer
  - 1 Tester
- Cycle time: 1756

- 30 cards
- With a team of 5:
  - 1 Designer
  - 4 Developers
  - 1 Tester
- Cycle time: 495

- 30 cards
- With a team of 5:
  - 1 Designer
  - 4 Developers
  - 1 Tester
- Plus, Designers can do Development…
- Cycle time: 459







Make prediction with confidence!

- Extend the algorithm to do multiple runs
  - Eliminates anomalies from a single run
  - Gives you a 95% confidence number over a large number of runs
- Product demo

Thank you…

Q&A