Principles of Product Development Flow

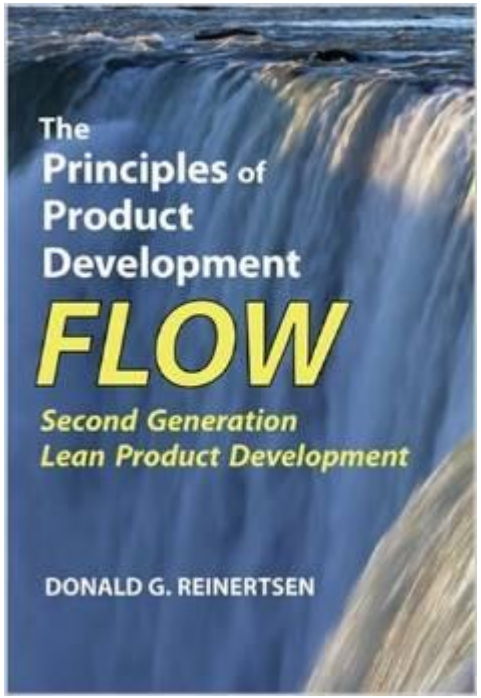
Part 5: Reducing Batch Size

About Me

- Started programming in 1981
- Owner of Enoki Solutions Inc.
  - Consulting and Software Development
- Exposed to several industries
- Running VanDev since Oct 2010

Book:
The Principles of Product Development Flow

- ~$45 on Amazon.ca
- Published in 2009
- Award winning
- Difficult material
- Generally ignored :(
- Awesome IMNSHO



“Don’t test the water with both feet.”
-- Charles de Gaulle

B1 The Batch Size Queueing Principle: Reducing batch size reduces cycle time.

- Doing less should take less time

B2 The Batch Size Variability Principle: Reducing batch size reduces variability in flow.

- Smaller batches are more likely to be in the linear segment of the production curve
- High variability tasks are more isolated
  - Reduced impact of noisy neighbors

B3 The Batch Size Feedback Principle: Reducing batch size accelerates feedback.

- Finishing more often
  - publishing sooner
- Publishing sooner
  - earlier feedback

B4 The Batch Size Risk Principle: Reducing batch size reduces risk.

- Early feedback
  - discover issues early
- Early discovery
  - early correction
  - less compounding of the issue

B5 The Batch Size Overhead Principle: Reducing batch size reduces overhead.

- Lower work in progress
- Less to manage and track
- Less to verify changes against
- Less context switching

B6 The Batch Size Efficiency Principle: Large batches reduce efficiency.

- Larger batches lead to stalls
  - A large transfer of work from Dev to QA
  - large queue in QA
  - block QA from accepting more work
  - stalls other teams
- Packet switched networks won.

B7 The Psychology Principle of Batch Size: Large batches inherently lower motivation and urgency.

- Saving an hour on a task that won't be touched for another month is not perceived as important
- Saving an hour on a task that will immediately be released when it is done is perceived as important

B8 The Batch Size Slippage Principle: Large batches cause exponential cost and schedule growth.

- Historic data of large batch system show an exponential curve in lateness vs original projected delivery
- His point is purely empirical
- No mention of data on small batches
  - IME: Small batches lead to changing the plan

B9 The Batch Size Death Spiral Principle: Large batches lead to even larger batches.

- Batch priority grows with size
- Want something to be high priority?
  - Add it to the batch with highest priority
  - Batch is now even bigger
- Large batches are not allowed to fail
  - Too costly to let it fail
  - Reality gets obscured

B10 The Least Common Denominator Principle of Batch Sizes: The entire batch is limited by its worst element.

- Delivery is based on the slowest element
- Restrictions on the batch are the union of all restrictions of its parts
- Dependencies of a batch are the union of all the dependencies of its parts

B11 The Principle of Batch Size Economics: Economic batch size is a U-curve optimization.

- Lots of math
- Result: You don’t need to get batch size perfect
- Pressure towards small batches is best

B12 The Principle of Low Transaction Cost: Reducing transaction cost per batch lowers overall costs.

- Focus on reducing transaction costs
  - automation, ci, cd, tests, etc.
- Enables smaller batches

B13 The Principle of Batch Size Diseconomies: Batch size reduction saves much more than you think.

- Transaction costs scale with batch size
  - More overhead in the transfer as the manifest grows
- Holding costs increase faster than linear
  - Deferring releasing work
    - decreases its market value
    - hides bugs longer
    - delays feedback longer

B14 The Batch Size Packing Principle: Small batches allow finer tuning of capacity utilization.

- Schedule large batches first
- Fill in the gaps with small ones
- “Rocks in a jar”
  - Count is limited by the smallest sized rocks
- Packet switching
  - Small packets exploit available capacity windows better.

B15 The Fluidity Principle: Loose coupling between product subsystems enables small batches.

- Independant systems can be developed and tested in parallel
  - Requires stable interfaces
- Packet network example:
  - divide, number packets
  - each packet takes independent routes
  - reassembly at receiver

B16 The Principles of Transport Batches: The most important batch is the transport batch.

- Change in state vs Change in location
  - fix bug vs release bug fix
- Individual test results vs Test acceptance
  - results of tests reported as the run
    - earlier reaction possible
  - overall acceptance reported at the end

B17 The Proximity Principle: Proximity enables small batch sizes.

- Collocated teams
  - faster communication
- Cross functional teams
  - direct communication

B18 The Run Length Principle: Short run lengths reduce queues.

- Limit WIP
- Mix task types in sequencing
  - Less certain
  - More certain
  - Opportunities for catching up

B19 The Infrastructure Principle: Good infrastructure enables small batches.

- Automated infrastructure
  - daily build
  - daily test reports
  - daily deployment
- Automated setup of automated infrastructure
  - new product setup
  - new component setup

B20 The Principle of Batch Contents: Sequence first that which adds value most cheaply.

- Highest risk first
  - failure cut-off
- Perpetual Motion Car: which do you do first?
  - 5% chance of success, $1M perpetual motion engine
  - 95% chance of success, $99M design car

B21 The Batch Size First Principle: Reduce batch size before you attach bottlenecks.

- Reduced batch sizes lower variability
  - Reduced variability lowers peak utilization
- Lunch lines
  - Double you cafferia size (capacity)
  - Stagger lunch breaks (batch size)

B22 The Principle of Dynamic Batch Sizes: Adjust batch size dynamically to respond to changing economics.

- Market conditions over the production lifetime change
- Change batch sizes with them
- Regulation introduction (Apple, etc.)
  - transaction costs raised
  - pressure to larger external releases

Conclusions

- Batches should be a small as economically reasonably
- Push to too small; then pull back
- Decouple dependencies

→ In software: SOLID principles

Q&A