Principles of Product Development Flow

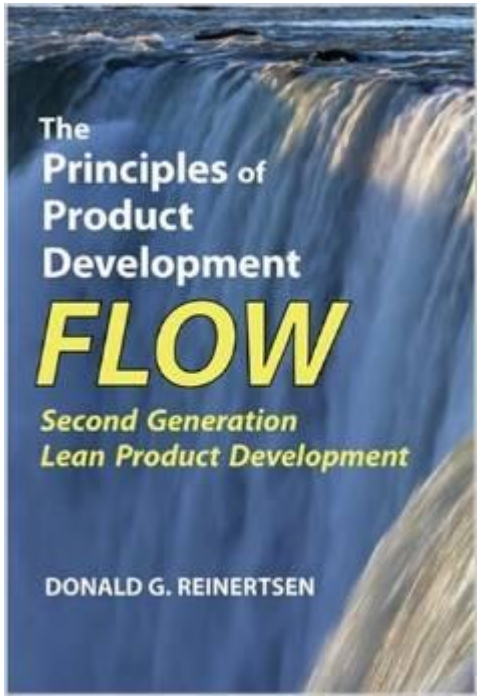
Part 6: Applying WIP Constraints

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



W1 The Principle of WIP Constraints: Constrain WIP to control cycle time and flow.

- WIP reduces cycle time at the expense of permanently rejecting some demand and reducing utilization
- Total economic benefit of setting the WIP limit at 2x the average WIP without a limit is approximately 10 fold

W2 The Principle of Rate-Matching: WIP constraints force rate-matching.

- If you can’t get the work out of your queue because upstream isn’t ready do **not** start new work
- This prevents you from getting ahead of your consumer and forces your production rate to match their consumption rate

W3 The Principle of Global Constraints: Use global constraints for predictable and permanent bottlenecks.

- Theory of Constraints: release work into the system at the rate of the worst bottleneck
- Great for stable bottlenecks
- Not so great for the dynamic ones we encounter in software development
- WIP limits expose dynamic bottlenecks

W4 The Principle of Local Constraints: If possible, constrain local WIP pools.

- Consumption rate changes reflect upstream quickly
- Systems upstream of a blockage hold maximum WIP
- When a blockage is removed flow resumes quickly
- Kanban beats TOC in dynamic systems

W5 The Batch Size Decoupling Principle: Use WIP ranges to decouple the batch sizes of adjacent processes.

- Allow for work to spin down (WIPL from 0-X)
- Allow for batch decomposition
  - Batch of 100 in (A), batches of 10 out (to B)
  - WIPL for A is 0-2, B is 0-11
- Batch sizes may be the same, but task time may vary

W6 The Principle of Demand Blocking: Block all demand when WIP reaches its upper limit.

- Telephone system (fast busy signal)
- ~~Push~~, Just say NO
  - packet networks drop packets!
- Holding in a queue further back reduces costs
  - Cheaper to hold a project before it starts

W7 The Principle of WIP Purging: When WIP is high, purge low value projects.

- High WIP -> high queues
- High queues -> higher cost of delay (Q15)
- Tasks that made economic sense during low cost of delay may no longer make sense
  - Remove them
- Avoid zombies: projects not good enough to get resources, not bad enough to kill

W8 The Principle of Flexible Requirements: Control WIP by shedding requirements.

- Order features by priority
- Loosely couple features
- Component based architecture
- Know what is likely to be dropped ahead of time (expectation setting)

W9 The Principle of Resource Pulling: Quickly apply extra resources to an emerging queue.

- Move resources to blockages
  - Devs -> Testing/Validation
- Prefer to move resource to later processes
- Reducing emergent queues keeps flow stable
- Added workers do not have to be as efficient due to non-linear impact of queue sizes

W10 The Principle of Part-Time Resources: Use part-time resources for high variability tasks.

- I disagree with his examples
  - splitting resources over different work loads is usually bad in software
- I agree with bringing in contractors to attack predictable bottlenecks
  - Final-ing a game
  - Post launch support

W11 The Big Gun Principle: Pull high-powered experts to emerging bottlenecks.

- Lightly load your experts!
- Use them to attack the problems no one else can solve
  - These problems are rarely predicable!
  - Wait from the to emerge and then deploy the expert
- Don’t waste them on small problems

W12 The Principle of T-Shaped Resources: Develop people who are deep in one area and broad in many.

- Experts need broader knowledge so they can come up to speed on any part of the system faster
- Broader knowledge allows a person to be used to fight emerging queues in other areas
- All devs should know good testings practices!

W13 The Principle of Skill Overlap: Cross-train resources at adjacent processes.

- Devs should have some understanding of
  - qa (testing)
  - ba (requirements)
  - operations
  - system administration
  - persistent data store setup and maintenance
  - analytics
  - ...

W14 The Mix Change Principle: Use upstream mix changes to regulate queue size.

- Pre-evaluate work and hold back work that will create queues
- Mix (alternate):
  - high design/low development
  - low design/high development
- Based on work in flight try to fill the gaps

W15 The Aging Principle: Watch the outliers.

- Don’t rely on averages!
  - avg {1,1,1,1,1,1,1,1,1,10} = 1.9
  - that 10 is a signal
- Outliers indicate something that should be investigated
- Black Swans
  - 999x 1’s + 1x 1000
  - Don’t rely solely on history either

W16 The Escalation Principle: Create a preplanned escalation process for outliers.

- Fire escape plan
- Fire drills
- Fire detection
  - It’s a smoke detector, not a fire detector
  - By the time it’s a fire it’s too late!
- Automated escalation

W17 The Principles of Progressive Throttling: Increase throttling as you approach the queue limit.

- Slow down before you hit the limit
  - smoother flow
  - leaves rooms for bursts
- E.g. if WIPL for a dev team is 5 and 3 spots open up, only allow 1 spot to be filled each day rather than all 3 in 1 day.

W18 The Principle of Differential Service: Differentiate quality of service by workstream.

- QoS is great in theory
  - expedite vs standard vs “low”
  - in practice everything is expedite!
  - non-expedite variability is greatly increased
- Only works if
  - service levels must have different costs
  - costs differences must be large enough to matter
- Must use PFIFO

W19 The Principle of Adaptive WIP Constraints: Adjust WIP constraints as capacity changes.

- Lower WIP around bottlenecks
- AIMD: Additive increase, multiplicative decrease
  - WIP goes up slowly (linearly: +1)
  - and down quickly (nonlinear: *0.8)

W20 The Expansion Control Principle: Prevent uncontrolled expansion of work.

- Set an upper limit on task time
- Re-evaluate the task after that time
  - Split and Reschedule
  - Re-prioritize
- “Timeboxing”
- **not** Parkinson's “law”
  - which has been disproven

W21 The Principle of the Critical Queue: Constrain WIP in the section of the system where the queue is most expensive.

- See V16
  - Cheaper to hold a plane on the ground than circling in the air
- Lower WIP around expensive queues
  - Ensure these processes PULL work!

W22 The Cumulative Reduction Principle: Small WIP reductions accumulate.

- A small difference in departure rate over arrival rate will reduce queue size
- Reduced WIP allows for quicker turnaround of tasks and reduces queue size
- Start at the last output and work backwards towards the inputs

W23 The Principle of Visual WIP: Make WIP continuously visible.

- With reduced WIP the work can be visualized physically
- Whiteboards and sticky notes becomes viable

Conclusions

- Limiting WIP is a powerful tool
- Slow down to go faster

Q&A