Table of Contents

Tim Ottinger - Agile Productivity

Premise

More, faster: everybody wants that. Low productivity makes products late and expensive, frustrating both teams and customers. But what is productivity? Why is it so hard to measure? Organizations want higher productivity, but use industrial-age models better-suited to assembling widgets than developing software. We'll explore a model that works for software teams. Learn several ways to boost - or drag down - productivity in your projects. You'll walk out with concrete techniques to apply in your context.

Learning Outcomes: Begin thinking in terms of impact, not output Move focus from "working harder individually" to "accomplishing more together" Recognize organizational habits that limit productivity Discover ways to organize teams and their work to increase productivity

Summary

- Content rating (0-no new ideas, 5 - a new ideas/approach, 9-new ideas): 8
- Style rating (0-average presentstion, 5 - my level, 9-I learned something about presenting): 7

Action / Learning

- Check out continuous development on their web site
- See Esther Derby on metrics - http://www.estherderby.com/2011/10/metrics-for-agile.html
- Like the concept of outcomes vs inputs as measure of productivity

Presentation

agileproductivity.pdf

And Bill Wake at Industrial logic

Notes

What does it mean to be unproductive?

Sitting meetings Talking Going to conference Unresponsive

Too. Long too get what we want

What is productivity - how is it measured

Units / developer Velocity Bugs reported Time to market Lines code Function points Customer sat

Cars in parking lot at 6pm Overtime % time heads down typing

What's wrong with it

Chandeliers / pound Measurement Built heavy that couldn't support

What is productivity Business? No Lots getting done? No Outputs / inputs? Closer but no

Productivity Value (outputs) / cost (inputs) Over some period of time

Productivity - outcomes vs inputs

Old thinking based on widgets Worth about the same Some variation Value is clear Order doesn't matter that much Output almost same as outcome

Code Wildly varying worth Much less certainty about needs Order significantly affects value - eg ship really valuable items first Output not same is outcome

How improve productivity Increase outcomes Decrease inputs

Helium balloon theory

People want to be accomplished Productivity rises to a the point where something stops them Remove problem and more productivity

Focus on value Smooth turbulent workflow Don't stockpile pain

A word game - see game 4 letter word from base word - a word game → grow Count words Talk like a pirate day

What did we learn One came next Permutations Little words are worth the same Patterns identified help Didn't know scoring Heard others Some people are much than others More effective over time Effort not equal to scoring value

Focus on value Didn't say “plus one the r words” Teams don't know if they are working on something of value Play with box, not the present - kid on birthday

Lean startup - build - measure - learn Cycle - no beginning Feedback from customer - we just need X (measure)

11/12 of what we do is thinking as developer - not typing

Work to plan Force to hit target we've identified

Agile Target moves as we learn more Hopefully we are converging

Develop features from a, b, c

Split for value Do a value focused plan Ship a bit of each to start with Decide to do infrastructure items Deliver value early Cut off any time

Smooth the turbulent workflow

Game Pick a bunch of volunteers

Programmers to left Qa to right Specialist step back

A lot of stories (blank cards)

Maximize efficiency Every developer each get our stories Developer dependency on architecture, dba go there A new card to be productive

Qa crickets

Card goes back to dev

See how efficient is this

How many stories have been done? How many in progress

Last day of sprint it will come together or “abandoned as done”

What if instead I went with my card to the dba Person getting next story doesn't take one requires dba Team and group and swarm

Sometimes people are paid to “clear the plate”

Value stream map See michaelnygard.com Value vs non value add

Efficiency - 97 days / 315 days = 30% Throughput - 100 fp / 315 days

Since nothing is happening put more stuff in

Work from outside to the outside - else suboptimize

Squeeze out the waste

Don't stockpile pain

Lawnmower in tall grass

People put it off

Multiple branches (not git) → pain Make decision for every branch

One customer - average 15 branches per developer

Interactions increase when you delay merging and refactoring and releasing

Small increases instead - do it as it goes

Release rates

Start with Development - code slush - code freeze - build / test - release Tail of process was months

CI Development days / weeks - build / test / release

CI + CD (deployment) Build / test / release at 20 mins

No benefit to us if we've developed a feature and we have not released it

Seek knowledge and share

Whole team perspective - working together Business - programmers - testers (3 amigos) have developed the right thing

Gold cards Pre-allocated learning time (one day free for learning) You get 2 cards per month - commit to sharing results with the team Take some serious time - 1 day Help deal with urgent vs important

We have to keep learning

Mob programming / pairing When we work together we can learning more quickly Helps with focus as well Separate machine on separate net so even email doesn't interrupt.

When you don't know how to do something tell it to the little rubber duck and by the time you finish you can do it

Pairing / mobbing - work pulls us together

Waste more time than you aware of

Down, down, down, paste, down, down, down How long will it take for you to use search / replace But someone watching will tell you

All of us together are smarter than each of us alone.

Estimation, Forecast, Conference, Metrics