

Steering With Numbers

Tim Bacon, ThoughtWorks
tbacon@thoughtworks.com

Are we at the same presentation?

- Metrics are evil
- But numbers are feedback and feedback is good
- Lots of numbers you could capture
 - Especially on an XP project
- I'll be giving an overview of ones that are useful
 - With qualifications!

Traditional Metrics

- Aim: measure X to encourage more (less) of it
 - Ranks people and projects
- Problem: unanticipated negative feedback loops
 - Formulae subvert success
- Problem: data alone is meaningless
 - Ranges / discontinuities, cause vs. effect
- Problem: people make projects incomparable
 - No silver bullet for software development

Steering with numbers

- Not measuring the distance from a line in the middle of the road
 - Relative not absolute numbers
 - Detecting changes not points in time
 - Guiding outcomes, not approaches
 - Not rewards or punishments
- Numbers provide a starting point
 - Raising questions
 - Prompting conversations

Customer viewpoint

- Traditional
 - When can I have it?
 - How much will it cost?
- Agile
 - What do I need to do to get what I want?

Customer numbers

- Tracking
 - Stories / Points / Cost / Value delivered & remaining
 - Points per iteration delivered: average, trend, current
 - Overall accuracy improves with time
 - But individual stories still subject to variation
- Done. Done? Done!
 - Developed – Tested? – Deployed!
 - Measure which activity takes time. How much? Why?
- Level of customer involvement requires feedback
 - Management tests?

Management tests

- Binary assertions about team relationships
 - E.g. “customer will be available 11-12 / 3-4 daily”
- Sets expectations, and provides feedback
- Too process-heavy / coercive?
 - Other ways to promote buy-in
 - Business / technical relationship requires collaboration

Customer steering

- “Scope changes but deadlines rarely alter”
 - Delivery & backlog numbers matter most, but which unit?
- Turnaround of ‘live issues’ is development work too
 - How are they prioritised / tested / deployed / tracked?
- Deployment is a non-trivial activity
 - How long does it take? Why? What can be done?
- Priority weightings are vital
 - Real-use feedback is available: which features are used in production (or not)?

Developer viewpoint

- Traditional
 - What do I need to do now?
 - How will I know when I am done?
- Agile
 - How should I develop?

Developer numbers

- Cards on an iteration story board
 - Planning sessions, Tasking out, Signing up
- Customer test passes
 - No programmer test fails
 - No other passing customer test fails
- Team values and practices
 - Measurable subjectively

Developer steering

- Test coverage should stay high
 - 100% for Programmer Tests, 70%+ for Customer Tests
- Taking ‘baby steps’ (red / green / refactor)
 - Check ins: small and frequent
 - Classes and methods: small and numerous
 - Build times: minimise aggressively

Development values and practices

- E.g. TDD, pairing, pace, integration, (in)tolerance, respect, simplicity, courage, feedback
 - How much importance should they have? How much importance are we giving them?
 - *None / Not Enough / Enough / Too Much*
 - Heartbeat retrospectives are invaluable
- Reinforced by example or eroded by neglect
 - Continuous improvement or lowest common denominator?

Project viewpoint

- Where have we come from?
 - Aim: remembering the past to avoid similar mistakes
 - What problems have we overcome?
 - What lessons have we learnt?
- Where are we now?
 - Aim: reflection / navel-gazing / sniffing for smells
 - What problems are we facing? Which are most important?
 - Are we fulfilling our own potential?
- Where are we going?
 - Aim: focussing / clarifying / planning
 - Is the road changing direction?
 - How are going to get to our next destination?

Process viewpoint

- Examine the bigger picture
 - 80% of variation is in the system, not the individuals
- Analyse the value chain
 - Where is value being added?
 - The rest is overhead (time / cost)
- Refocus the work so that proportion of value-adding activities is maximised
 - Lean Principles: minimise waste, reduce cycle time

Some lean measurements

- Defects
 - Amount outstanding / length of time outstanding
 - Developer defect or customer defect?
- Incomplete / untested / undeployed / unused code
 - Amount in each step / length of time in each step
- Task switching burden
 - 2 tasks 20%, 4 tasks 60%
- Time from customer 'want' to deployed code

An XP perspective

- Automated test suites => automated number collection
 - Programmer and customer test suites
- Low tech approach is simple and useful
 - Cards on a wall do highlight outliers
 - Wiki / ad-hoc tools / professional tools scale better
- Practice adherence is measurable
 - Coach role encourages feedback

Overview

- Agile projects are holistic not atomistic
- There are no Magic Metrics
 - Be careful what you measure!
- Why not ask your team?
 - They can tell you many problems
 - And suggest how to fix them
- Act at tipping points
 - “Do more of what is working and less of what isn’t”

Thank you!

- Any questions?