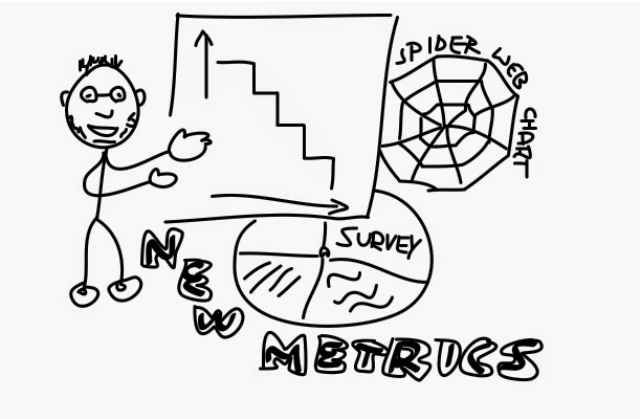
Unleash predictability by using Actionable Agile Metrics (6 Key Learnings from Daniel S. Vacanti's awesome book)

ontheagilepath.net/2015/04/unleash-predictability-by-using-actionable-agile-metrics-6-key-learnings-from-daniel-s-vacantis-awesome-book.html

By Sebastian Radics



I just finished reading the awesome book Actionable Agile Metrics for Predictability by Daniel S. Vacanti. For me a must read to get a deep insight in metrics for agile environments.

With this blog post I summarize my highlights and key learnings. It's just a fast overview and maybe provides already some ideas for you too, what to read next 😉
Predictability is of high importance! As an important question from our customers, we need to answer: **When will it be done?**

"The only metrics that entrepreneurs should invest in are those that help them make decisions" (Eric Ries – e.g. on Vanity Metrics vs. Actionable Metrics)
"If a metric does not offer predictive power, capturing that metric is waste" (Troy Magennis)

1 – Definitions

For me after a longer journey on the web a simple, speaking and coherent definition for cycle time, WIP, throughput and flow efficiency.

FLOW: Flow is the movement and delivery of customer value through a process – (as we build for the customer) therefore our whole process should be oriented around optimizing flow.
ACTIONABLE METRICS: The set of metrics that will suggest specific interventions that will result in the outcomes you are expecting

ARRIVAL POINT: A specific point where a unit of work transforms from being just some arbitrary idea into being a legitimate work item that is to immediately be acted on and

completed.
DEPARTURE POINT: Defined as delivery to an actual end user or delivery to some other downstream team or process
WORK: Any direct or indirect discrete unit of work of customer value is a candidate of work – named as work item (story, epic, feature, requirement, use case, enhancement,…)
WORK IN PROGRESS – WIP: The total count of items currently being worked on; the number of items that we are working on at any given time; all discrete units of customer value that have entered a given process but have not exited.

- predictor of over overall system performance
- all work items between arrival point and departure point
- can be segmented across different types

CYCLE TIME: The amount of elapsed time that a work item spends as Work in Progress.

- How long it takes each of those items to get through our process?
- How long to complete?
- When will it be done?
- Can also be used as a predictor of cost
- The amount of time it takes to get customer feedback

Unpredictability lies it the time an items spends waiting to be worked on – that's why it's the elapsed time that is important.

THROUGHPUT: The amount of WIP completed per unit of time.

- How many of those items complete per unit of time?
- How many features am I going to get in the next release

Understanding throughput at each step will help to identify the constraints in the workflow – find spots for process improvements
FLOW EFFICIENCY: Ratio of total elapsed time that an item was actively worked on to the total elapsed time that it took for an item to complete.

- not actively worked = waiting to be pulled, waiting for feedback
- often a starting point of 15% flow efficiency

LITTLE'S LAW:

Average cycle time (CT) = Average WIP (WIP)/Average Throughput (TH)

TH = WIP/CT

WIP = CT*TH

implies that increasing WIP leads to a higher CT and vice versa – check to reduce WIP to increase CT … in order to get stuff done faster, you need to work on less (on average)

Average items in queue = Average Arrival Rate * Average Wait Time
L = lambda * W

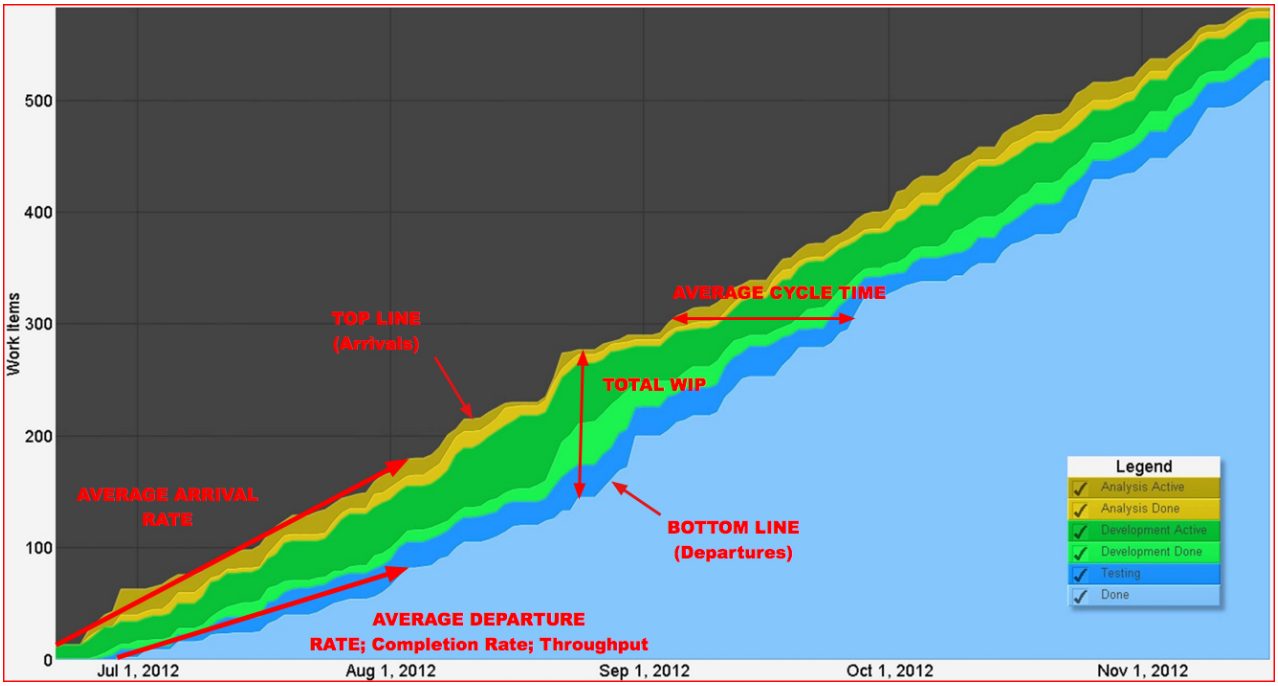
- L = average number of items in the queuing system
- lambda = average number of items arriving per unit of time
- W = average wait time in the system for an item

Thanks Daniel S. Vacanti for your explanation!

2 – Cumulative Flow Diagrams (CFD)

- offer a concise, coherent visualization of the three metrics of flow (Avg. Cycle Time, WIP, Throughput)
- provides qualitative and quantitative insight into problems with flow
- shows cumulative process arrivals and departures over time
- not a tool for projection (but introspection)
- Backlog should not be part of the CFD
    - they are not committed too (but in the diagram it looks like they are)
    - it will destroy cycle time calculation
- Try to show active and done states (as it shows areas of delays)

Avoid trap of drawing conclusions just by looking at the CFD! It's a tool to ask the right questions



Important Properties

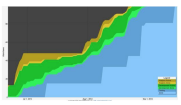
- Top line = cumulative arrivals
- Bottom line = cumulative departures
- No line can ever decrease! (it's a cumulative chart) … If it happens the chart is wrong (very likely work items disappeared in the process)

- The vertical distance between any two lines is the total amount of work that is in progress between the two workflow steps represented by the two chosen lines
- The horizontal distance between any two lines represents the approximate average cycle time for items that finished between the two workflow steps represented by the chosen two lines (average cycle time = bottom line date – top line date + 1)
- The data displayed depicts only what has happened (no projections allowed)
- The slope (top line) of any line between any two reporting intervals represents the average arrival rate
- The slope (bottom line) of any line between any two reporting intervals represents the average departure
- Average throughput = rise of average departure / run (Nr. of days)

Necessary assumptions

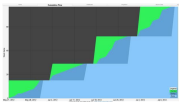
- The average Arrival Rate (Lambda) should equal the average Departure Rate (TH) = we will only start new work at about the same rate that we finish old work
- Needs a more late binding (commitment) approach.
- Monitor policies around the order in which we pull items through the system – so that work items do not sit and age unnecessarily
- all work started will be completed and exit the system
- WIP should be roughly the same in the chosen interval
- average WIP is neither increasing nor decreasing
- CT,WIP,TH are measured using consistent units

Some patterns



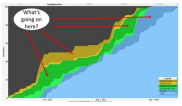
Flat lines

- check for mismatches of arrival and departure rates (items arrive faster than departure … means increasing WIP over time … leading to an increasing cycle time)
- check for flat lines … meaning no departures for a longer time (could be that nothing gets done, but also shows release phases or several public holidays)
    - Why isn't anything getting done?
    - What can we do to get things flowing again?



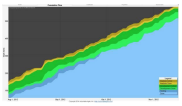
Stair steps

- e.g. with sprints and fixed arrival and departure dates
- Can batch periods be reduced? Eliminated? Check impact on cycle time…



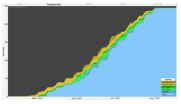
Bulging bands

- explosion of WIP in a particular workflow step
- maybe work is progressing slowly due to poor requirements or poor design?
- must not be in workflow step where it appears! Could also be due to a push from a previous step or a blockage in a downstream step(s) … try to separate Active and Done states (queuing) … Done bands should be as thin as possible



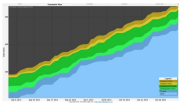
Disappearing bands

- Reporting interval could be too big (e.g. one week, but work flows rather quickly e.g. one-two days)
- upstream variability causes downstream steps to be starved or
- team decides to skip one step in the process frequently (you can remove this step from the workflow very likely)



S-Curve

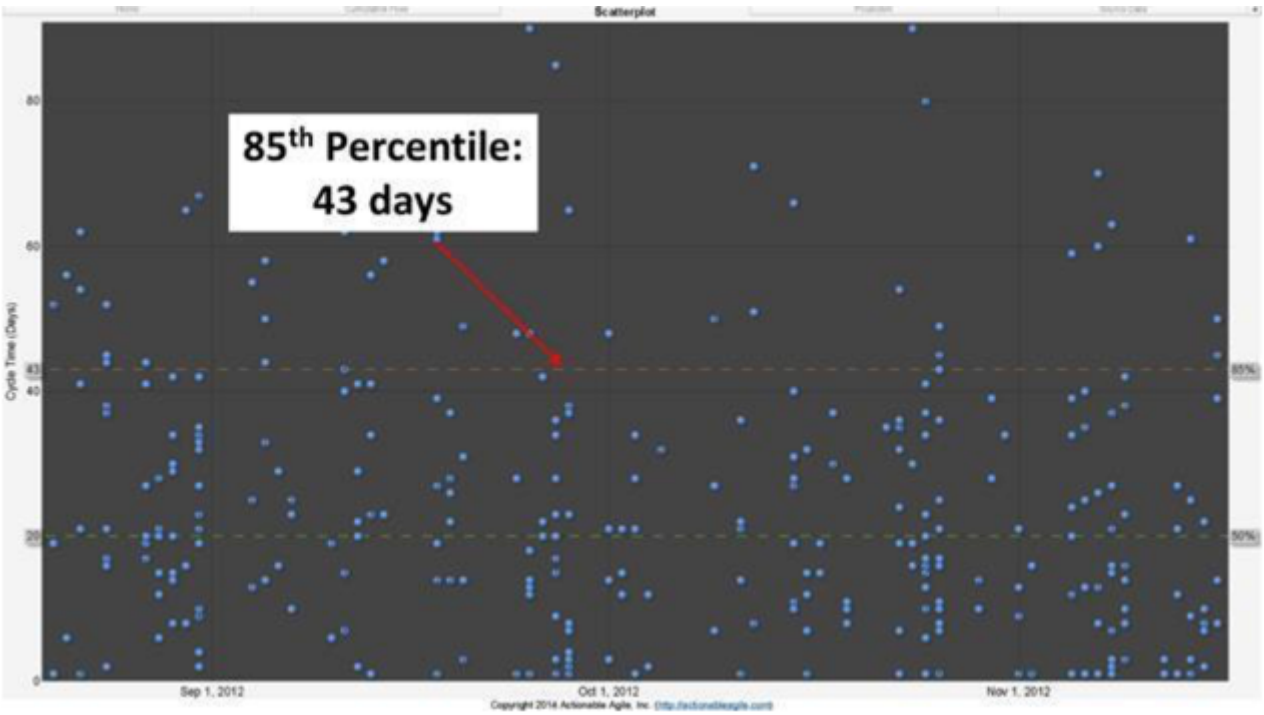
phases with zero WIP kill predictability and add inefficiencies



A boring one

- Can we get lines even closer together? (less WIP)
- Can we get the throughput line steeper?

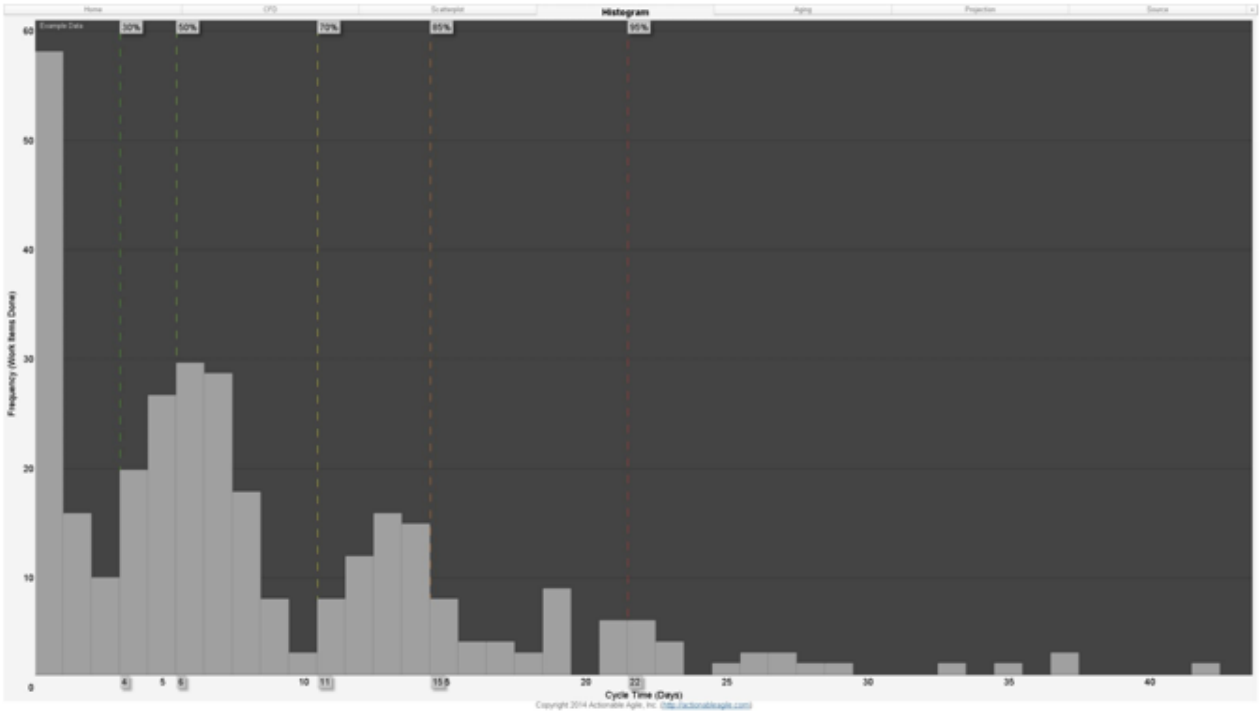
3 – Cycle time scatterplots



- x-axis it the timeline
- y-axis represents the cycle time (per item)
- percentile lines – e.g. 85% of all items finish within 43 days (per item)
- recommended is to use 50th, 70th, 85th, 95th percentiles
- work item cycle time data is not a normal distribution! – that's why applying standard deviation and arithmetic mean is not appropriate (as e.g. done with control charts)
- percentile lines are preferable – it's their robustness in the face of outliers

"If Bill Gates walks into a bar, then on average everyone in the bar is a millionare"

4 – Cycle time histograms



- a condensed, spatial view based on the frequency of occurrence of Cycle Times
- y-axis are the number of items
- x-axis is the cycle time
- vertical percentile lines (like in the scatterplot)
- in addition to the scatterplot the histogram shows the shape of the data. You can better detect patterns for Cycle Times over a given timeline … it's a more advanced cycle time analysis

5 – SLAs

- Cycle Time Target; Service Level Expectation
- it is expressed using a probability to meet a cycle time range
- e.g. with a 50% percent probability a work item finishes in 10 days (according to the 50% percentile of all previously finished work items in our system)
- can be used as a substitute for many upfront planning and estimation activities
- the choice of a teams SLA should be made in close collaboration with their customers

- get predictable at an overall system level first … very likely good enough … only optimize for subtypes if really necessary
- use the SLA for right sizing items too – SLA as the litmus test for right size of an item for flow through the process
- the older a work item gets, the greater chance it has of aging still more
- true definition of Agile is – to respond quickly to new information

One of the most common things we do that hinders our predictability is not pay attention to the order in which items are pulled through our process.

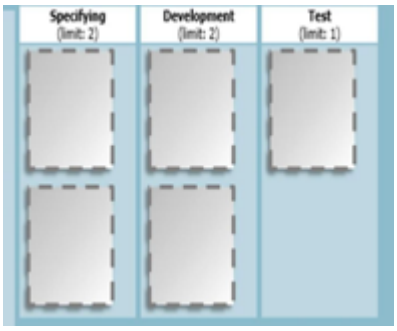
Mistakes

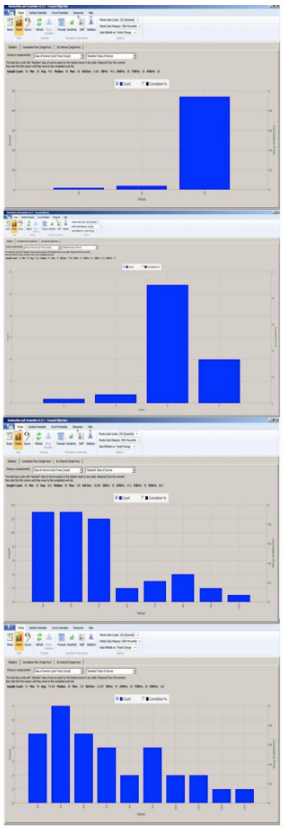
- set a SLA independent of analyzing cycle time data
- set by an external manager
- set without close customer collaboration

Classes of Services (CoS)

- **For all practical purposes, introducing COS is one of the worst things you can do to predictability**
- **CoS – every time you put a policy in place around the order in which you pull something**
- **will introduce variability and unpredictability into the process (e.g. will produce flow debt)**
- **Only introduce if you have operated your process for a while and are confident that CoS is necessary**

In his book Daniel S. Vacanti shows via a simulation the terrible effect on predictability of random pulling from queues in combination with having an expedite lane. A cycle time increase from 50 days to 100 days – meaning 100% more time.





FIFS (FIFO) – is the clear winner for cycle time predictability … the further you stray from FIFO, the less predictable you are

Slack

Slack is pretty much the only way to PREDICTABLY deliver in the face of variability introduced by different CoS is to build slack into the system.

6 – Forecasting

- a proper forecast includes a date range and a probability
- for forecasting a single items use SLAs
- do not use Little's Law and averages (as data is not in a normal distribution)
- straight line projections are problematic – they do not communicate a probability

Monte Carlo Method

The Monte Carlo Method is the future of forecasting in knowledge work.

Further readings

- Actionable Agile Metrics for Predictability by Daniel S. Vacanti
- my Book summary with more details and e.g. patterns for charts
- Troy Magennis work on Lean Forecasting e.g. Agile Metrics for the Metrically Challenged