Useful Metrics for Agile Teams

@Agile2022 :: @DocOnDev

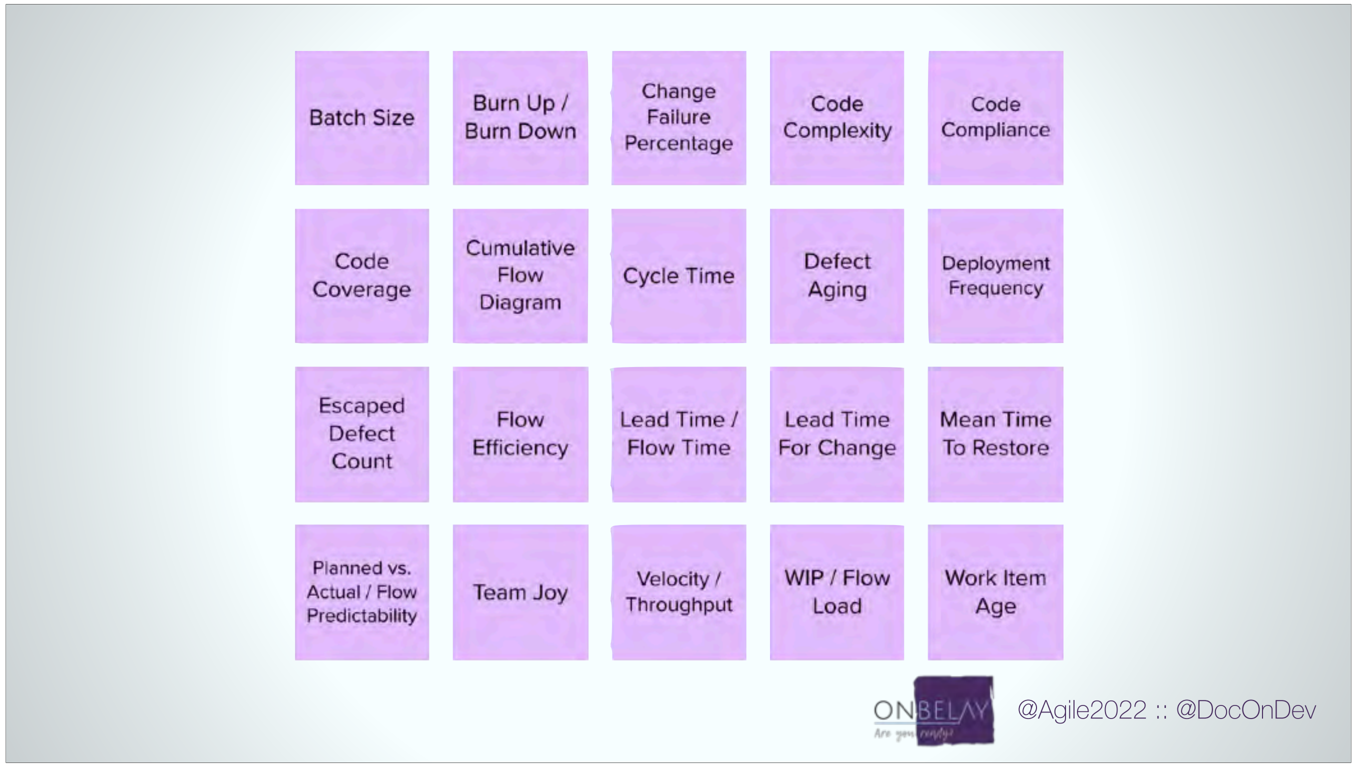
Which Metrics?

@Agile2022 :: @DocOnDev

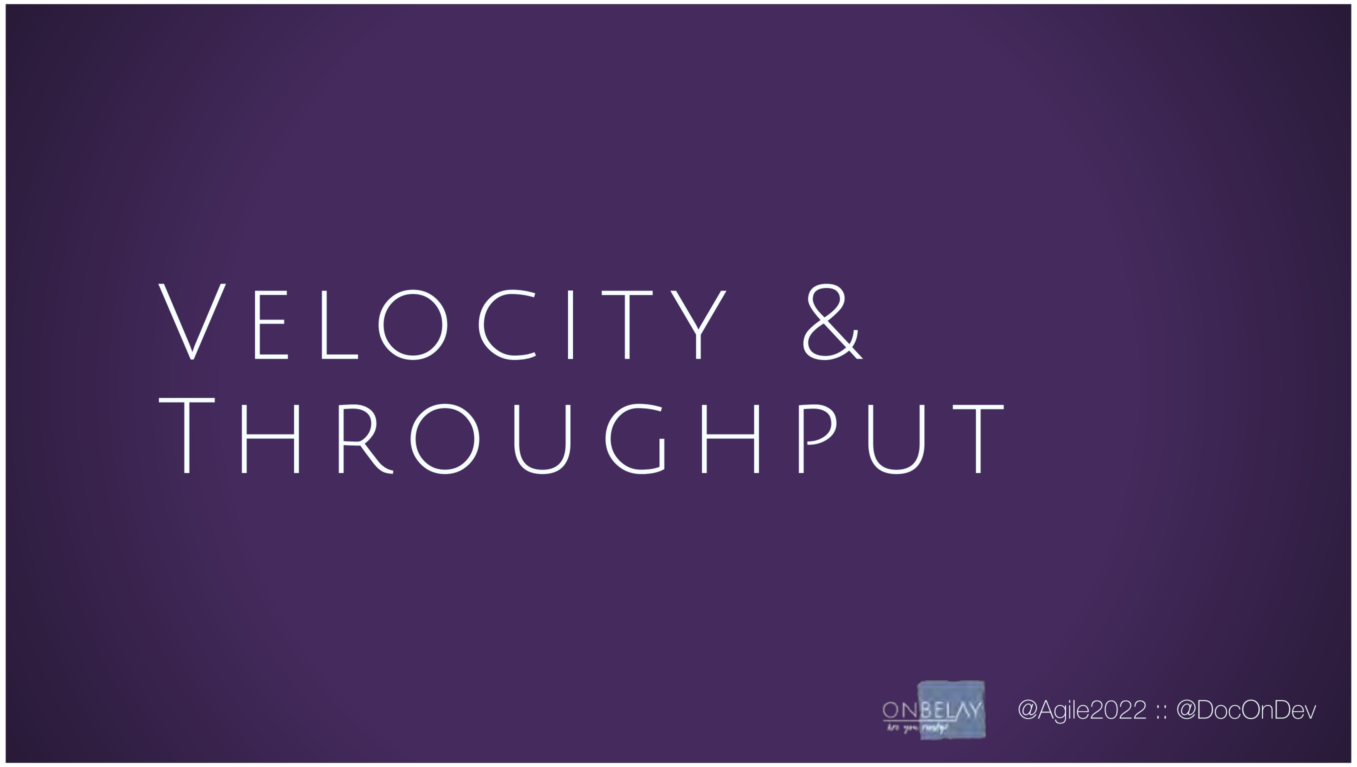
Come up with a list of two or three metrics at each table.

Metrics you use frequently or metrics you want to know more about.

Doc - Record metrics from each table in Mural



Batch Size	Burn Up / Burn Down	Change Failure Percentage	Code Complexity	Code Compliance
Code Coverage	Cumulative Flow Diagram	Cycle Time	Defect Aging	Deployment Frequency
Escaped Defect Count	Flow Efficiency	Lead Time / Flow Time	Lead Time For Change	Mean Time To Restore
Planned vs. Actual / Flow Predictability	Team Joy	Velocity / Throughput	WIP / Flow Load	Work Item Age



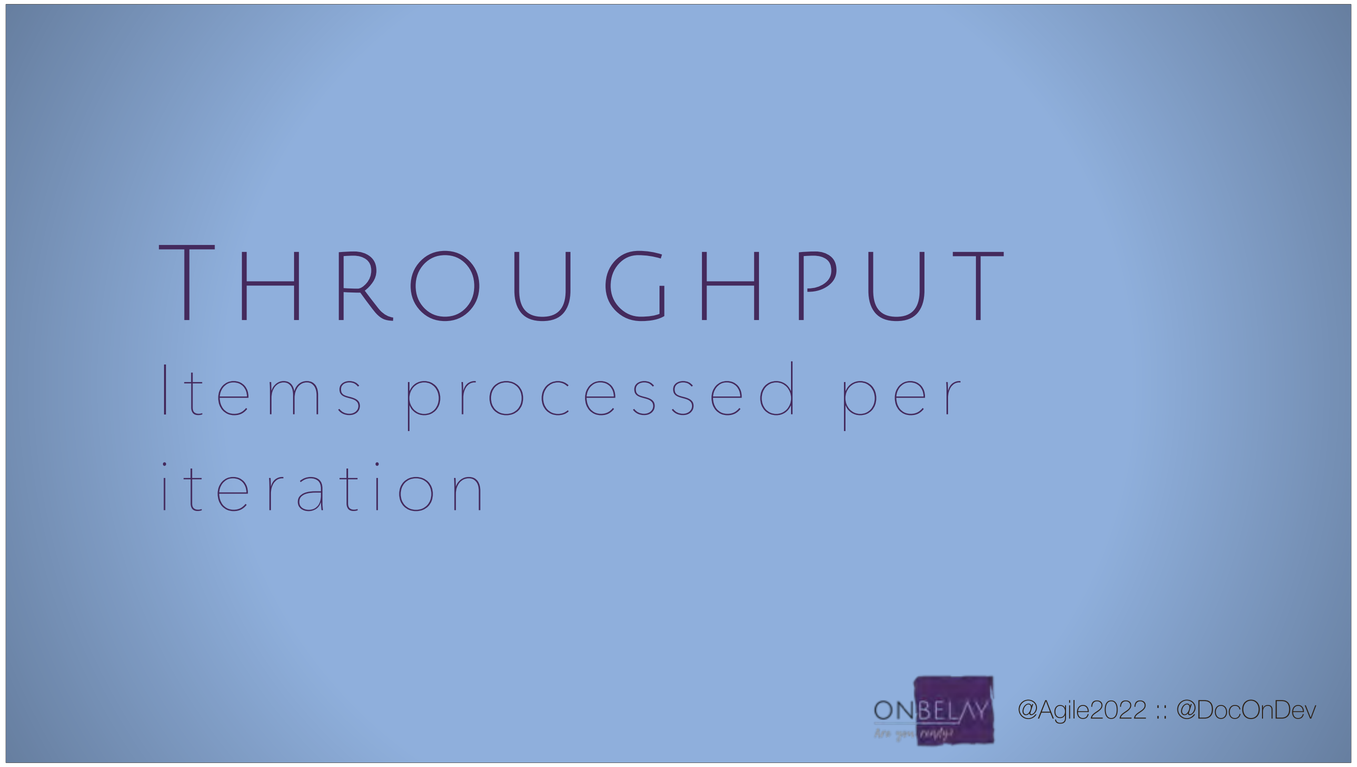
Velocity & Throughput

@Agile2022 :: @DocOnDev

Velocity & Throughput are essentially synonymous.

Velocity is Scrum vernacular that has leaked into the industry - Sprint is another example of this

Throughput - the rate at which a team delivers



THROUGHPUT

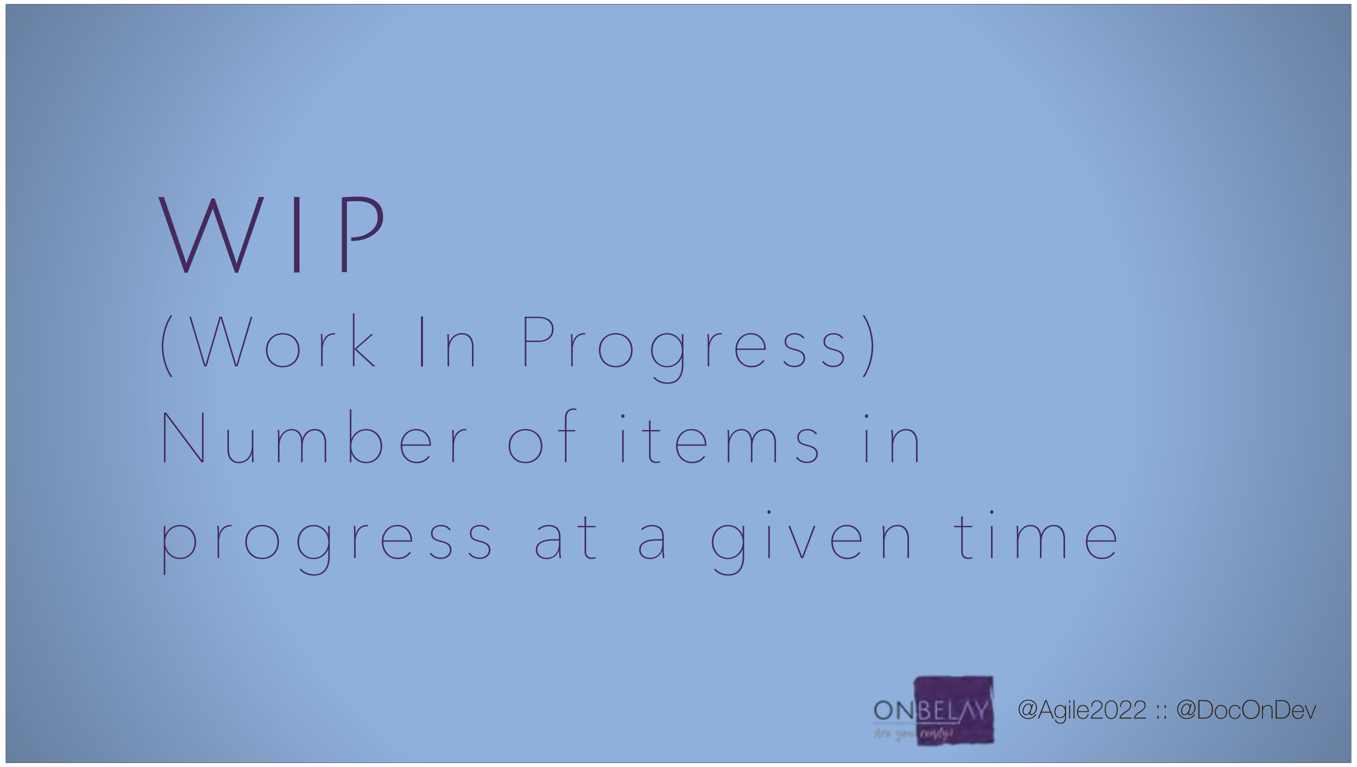
Items processed per iteration

ONBELAY @Agile2022 :: @DocOnDev

Throughput can be expressed as a single measure or as an average
If you process three items in a given iteration, your throughput is 3 for that iteration
If you process an average of 4.2 items per iteration, your average throughput is 4.2

WIP & Flow Load

WIP or Work in Progress is the same as Flow Load in SAFe



WIP

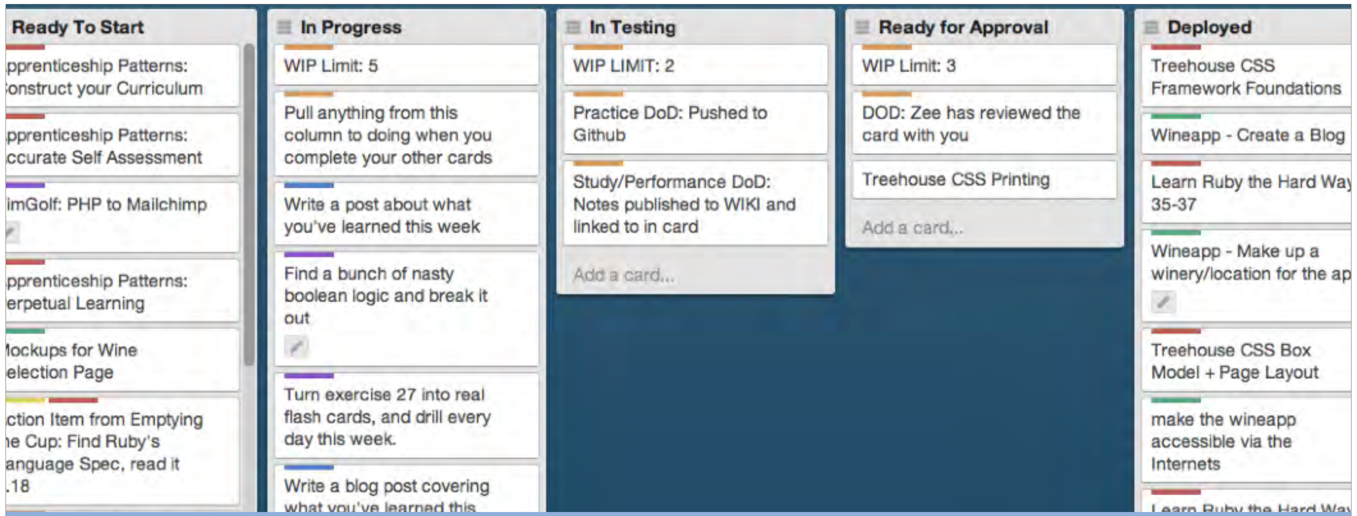
(Work In Progress)
Number of items in progress at a given time

@Agile2022 :: @DocOnDev

Work In Progress is the measure of the number of items in progress at a given time

This can be the number of items in a specific state, such as in development
This can also be the number of items in all states

For example <NEXT>



Ready To Start

- pprenticeship Patterns: onstruct your Curriculum
- pprenticeship Patterns: ccurate Self Assessment
- imGolf: PHP to Mailchimp
- pprenticeship Patterns: erpetual Learning
- lockups for Wine election Page
- ction Item from Emptying ne Cup: Find Ruby's anguage Spec, read it .18

In Progress

- WIP Limit: 5
- Pull anything from this column to doing when you complete your other cards
- Write a post about what you've learned this week
- Find a bunch of nasty boolean logic and break it out
- Turn exercise 27 into real flash cards, and drill every day this week.
- Write a blog post covering what you've learned this

In Testing

- WIP LIMIT: 2
- Practice DoD: Pushed to Github
- Study/Performance DoD: Notes published to WIKI and linked to in card
- Add a card...

Ready for Approval

- WIP Limit: 3
- DOD: Zee has reviewed the card with you
- Treehouse CSS Printing
- Add a card...

Deployed

- Treehouse CSS Framework Foundations
- Wineapp - Create a Blog
- Learn Ruby the Hard Way 35-37
- Wineapp - Make up a winery/location for the ap
- Treehouse CSS Box Model + Page Layout
- make the wineapp accessible via the Internets
- Learn Ruby the Hard Wa

Work In Progress (WIP)

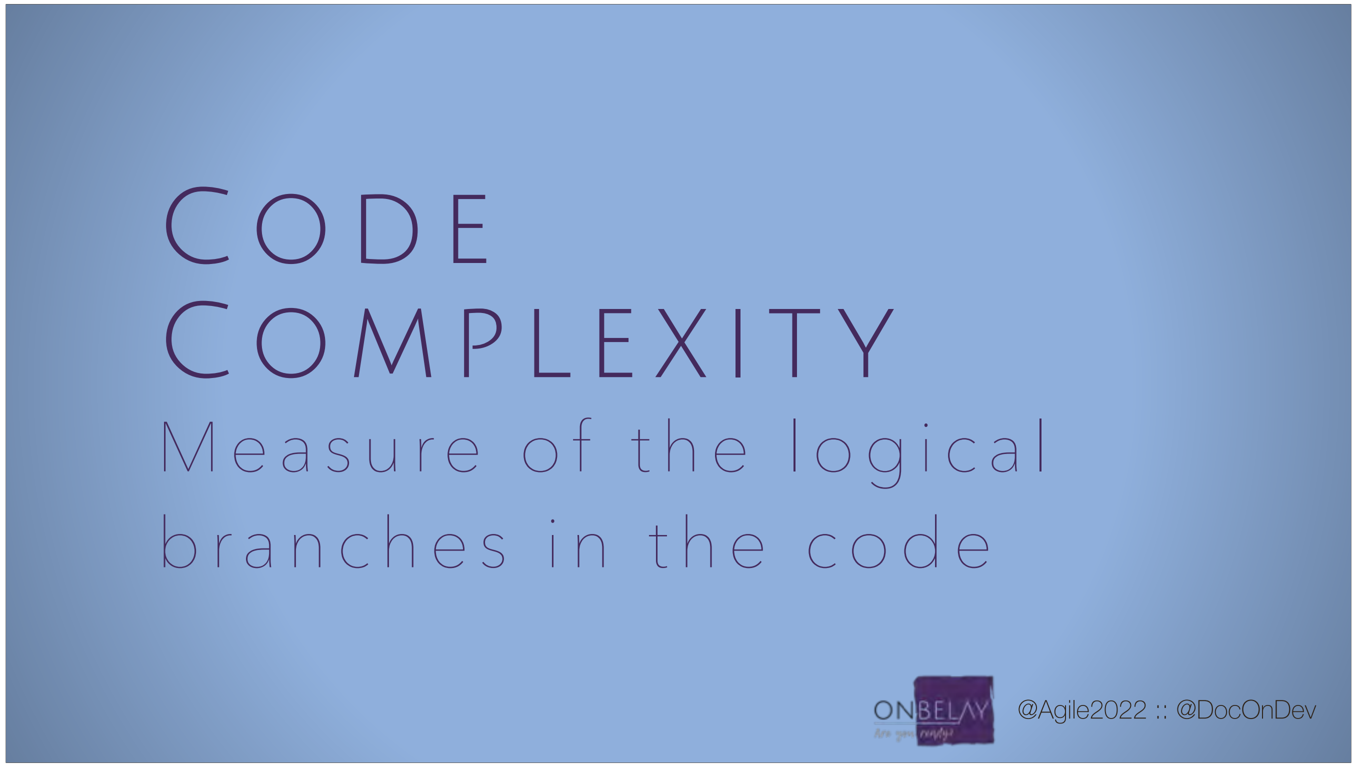
a.k.a. Flow Load

ONBELAY

This team has
5 items in WIP for In progress
2 items in WIP for Testing
2 items in WIP for Ready for Approval

That is a total of 9 items in WIP for the team

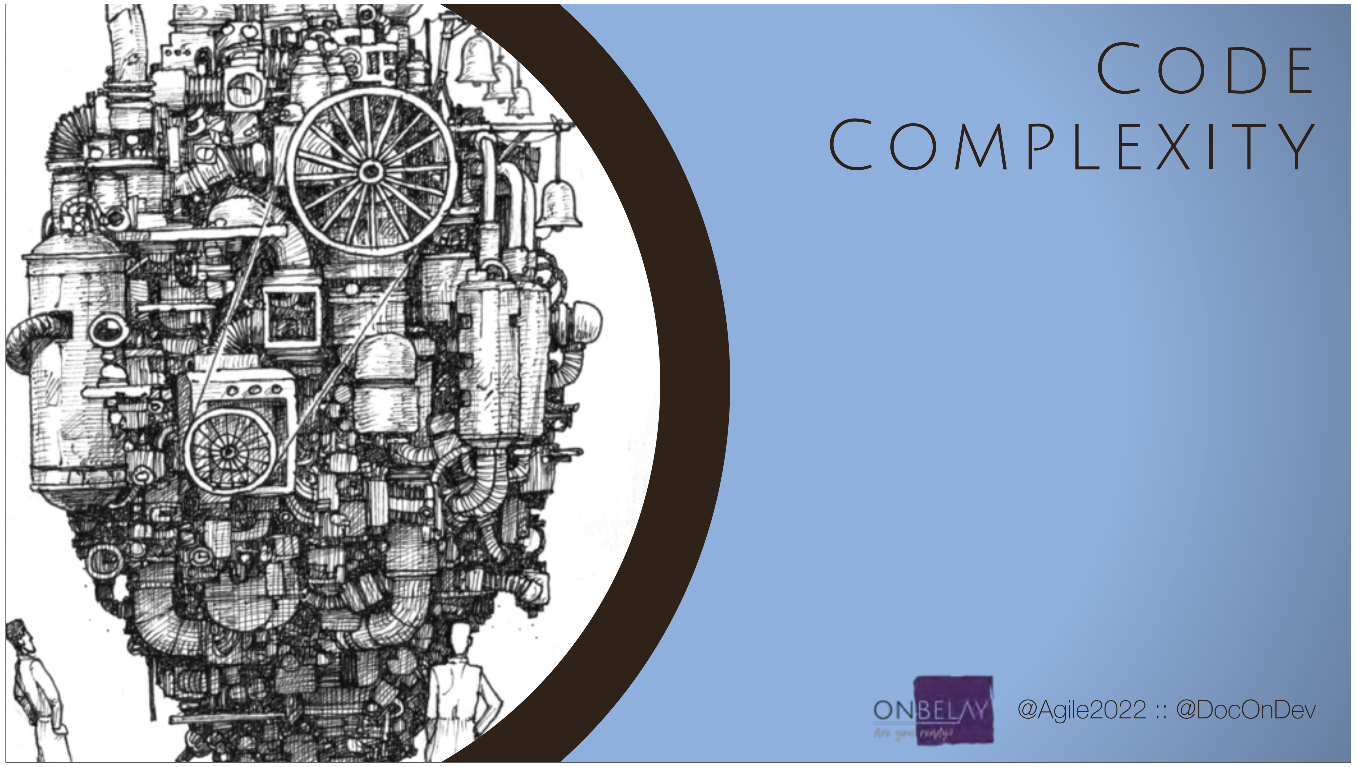
Code Complexity



Code Complexity

Measure of the logical branches in the code

This can be done with static analysis tools against the code



Code Complexity

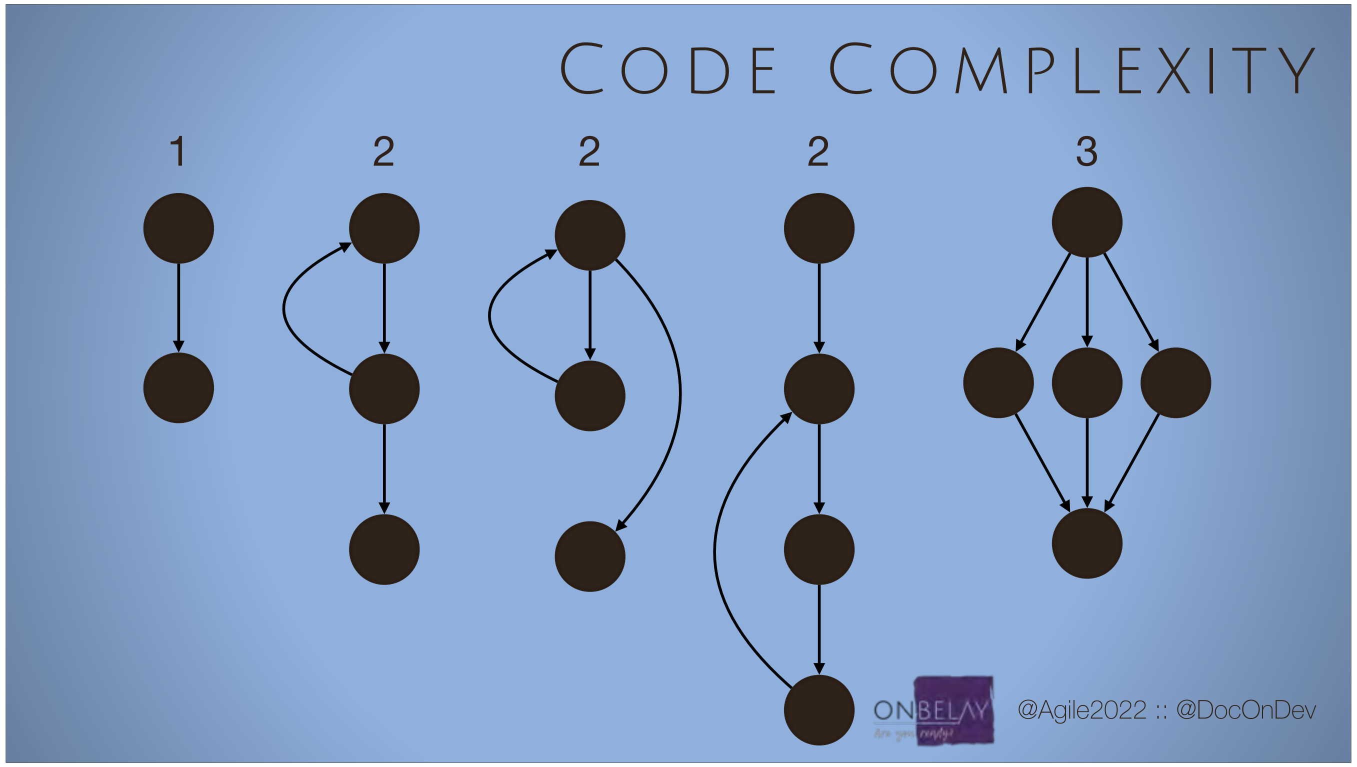
@Agile2022 :: @DocOnDev

You can use Cyclomatic or ABC Complexity.

Cyclomatic Complexity, also known as McCabe's number is based on nodes and edges in the code tree. It is essentially a count of linearly independent paths in the code. It was designed as a means of determining the number of tests you need for a piece of code.

ABC is similar to Cyclomatic Complexity, but is based on Assignments, Branches, and Conditionals, so it is a bit more robust. It was originally intended to be used as a means of forecasting.

Today, both of these measures are used as a proxy for code quality.



Code Complexity

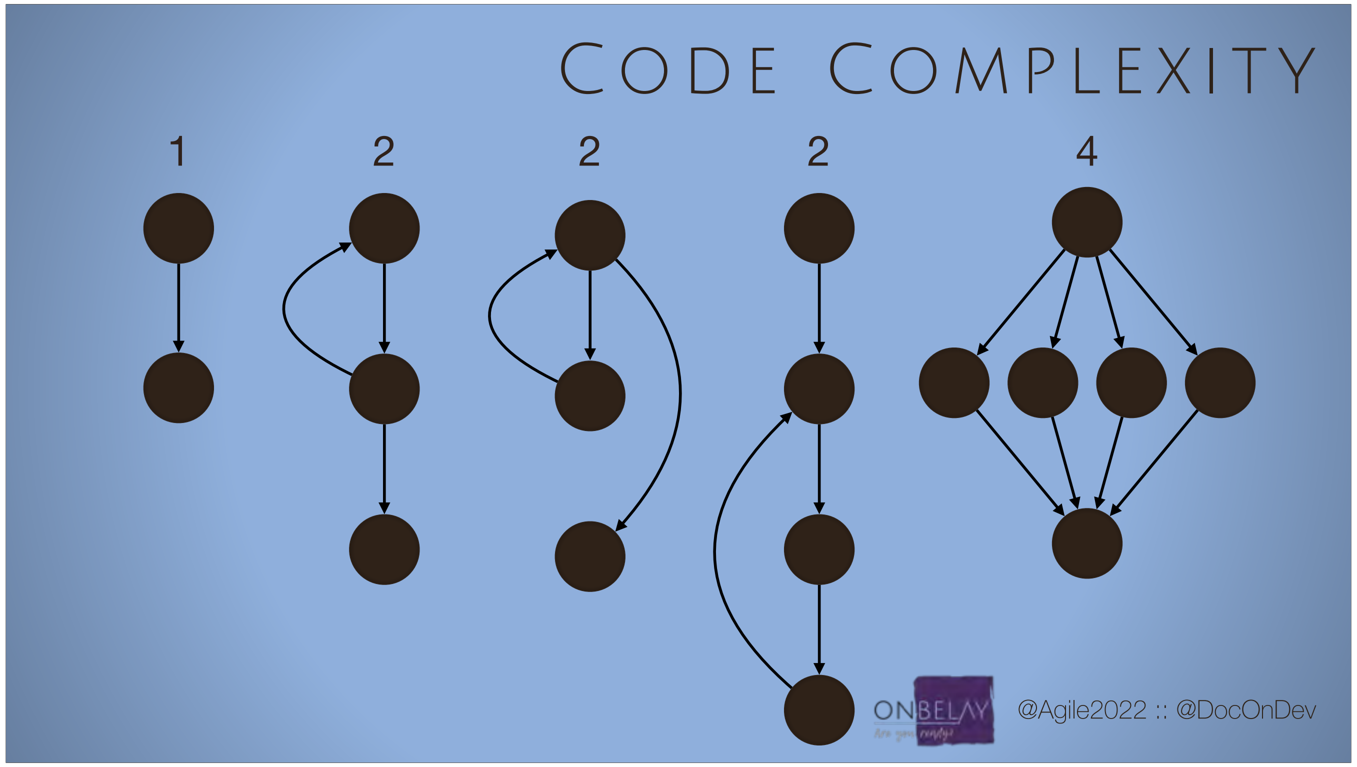
ONBELAY

@Agile2022 :: @DocOnDev

A strictly linear program has a cyclomatic complexity of 1 <next>

Whereas a Do Until, While, and For all have <next> a complexity of 2

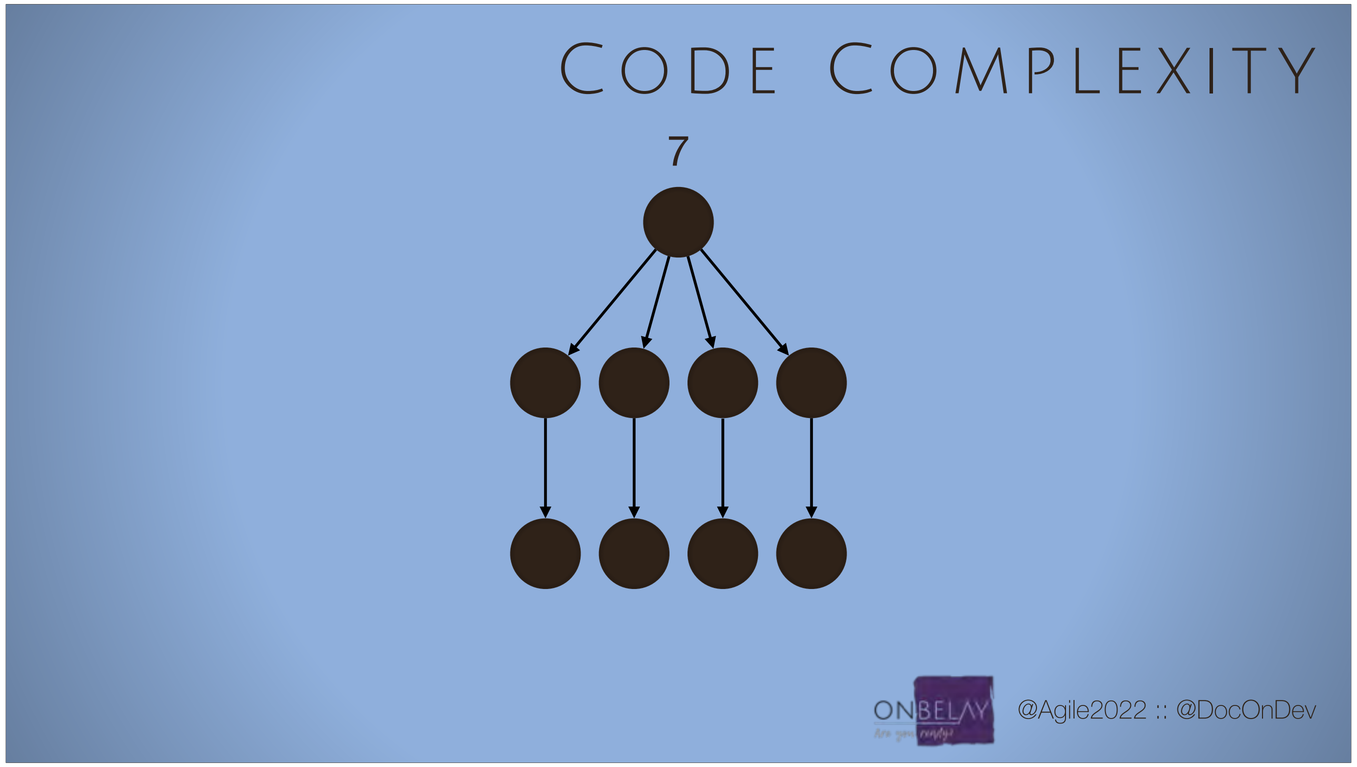
A case statement <next> grows in complexity by one for every option<next>



Code Complexity

@Agile2022 :: @DocOnDev

And if those case statements each exit the program, <next>

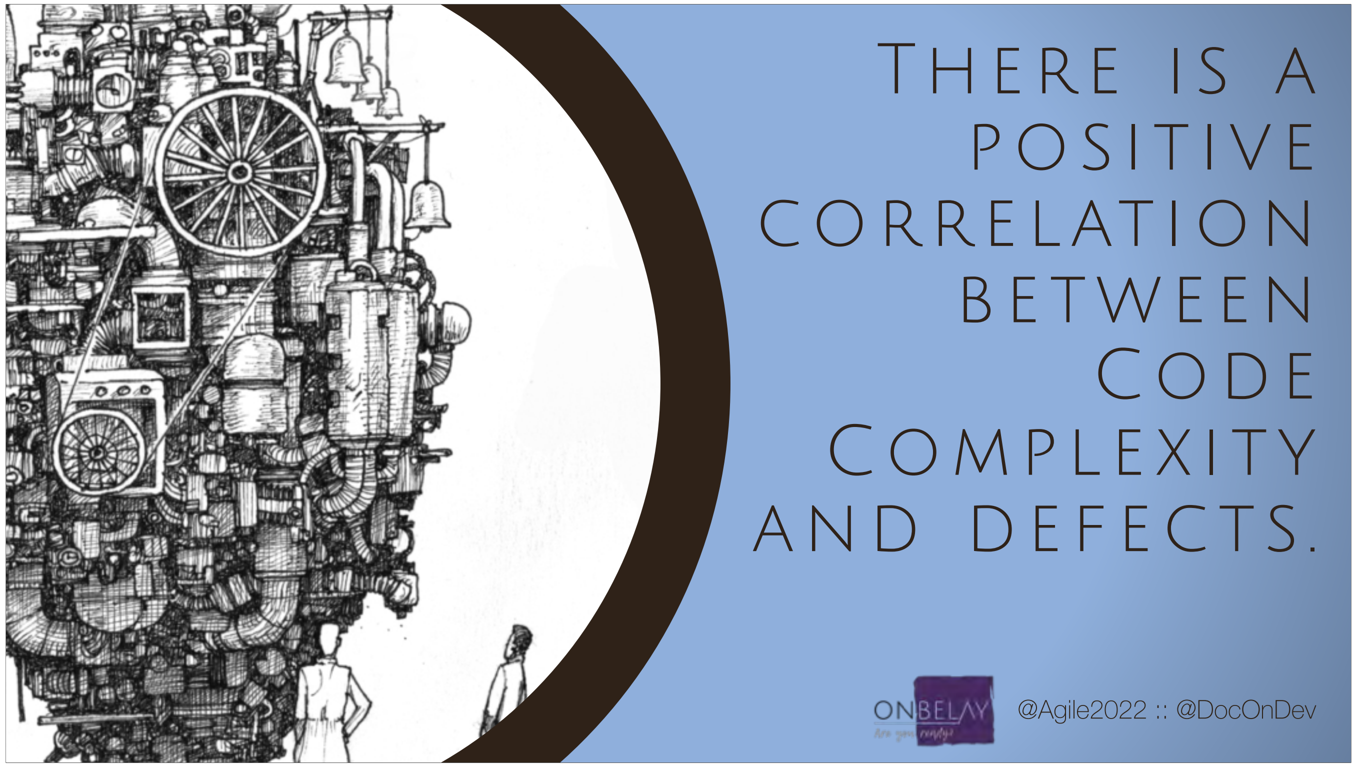


Code Complexity

@Agile2022 :: @DocOnDev

the complexity increases by 2 for every new option

Why do we care about this?
Because <next>



There is a positive correlation between code complexity and defects.

@Agile2022 :: @DocOnDev

The higher the complexity, the more likely a method is doing too much or has low cohesion.
The more responsibility a method has or the lower the cohesion, the higher the likelihood there are defects lurking therein.

Many conditionals can be collapsed with some refactoring - You can take look at basic inheritance and the factory pattern for more on this.

Escaped Defect Count

@Agile2022 :: @DocOnDev

Ways they forecast now.

Ways they know that you’re going to make it to the deadline given some quantity of work and a target date.

Escaped Defect Count

Number of defects introduced (or existing) in production

This is a trailing indicator of issues.

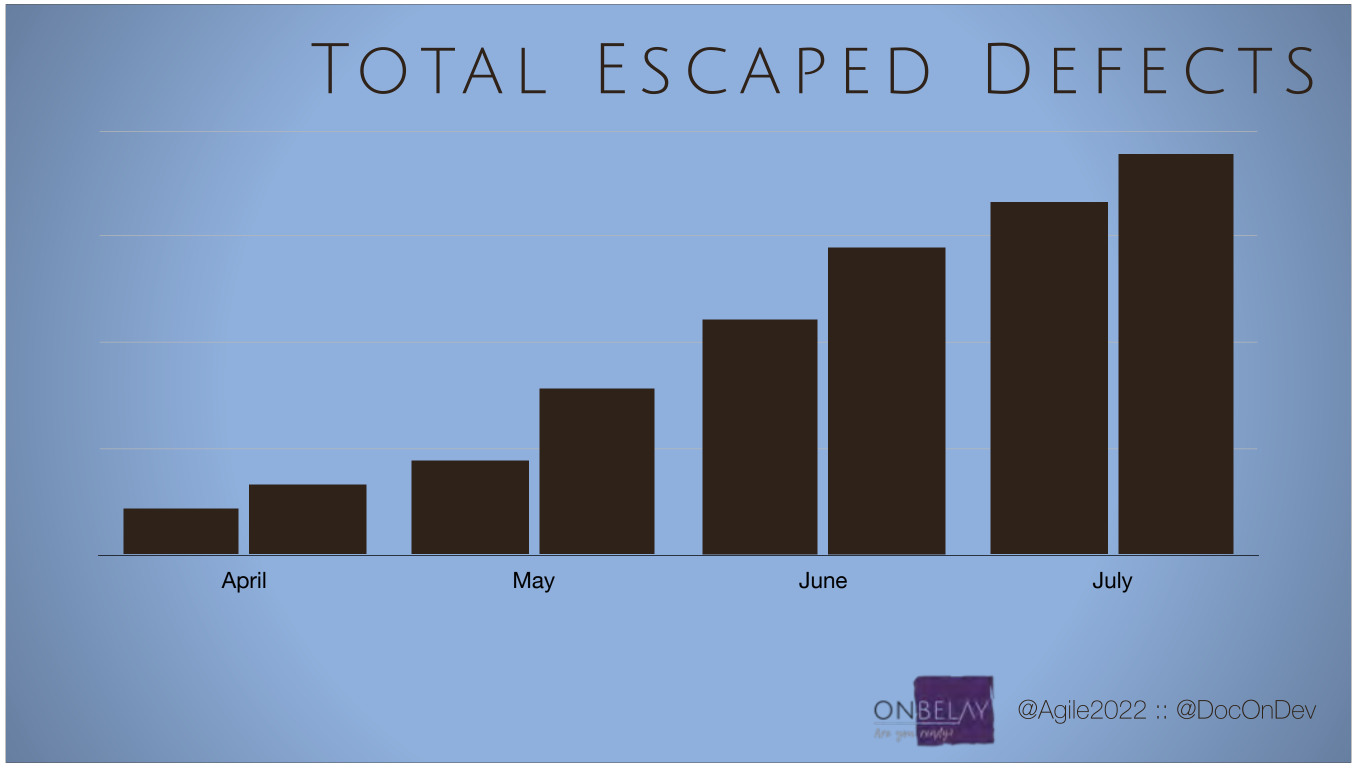
Escaped Defects

@Agile2022 :: @DocOnDev

Count of defects found in production.

If at all possible log the defect against the release during which it was introduced. For some defects, this will be easy, but others might not rear their heads for weeks or even months.

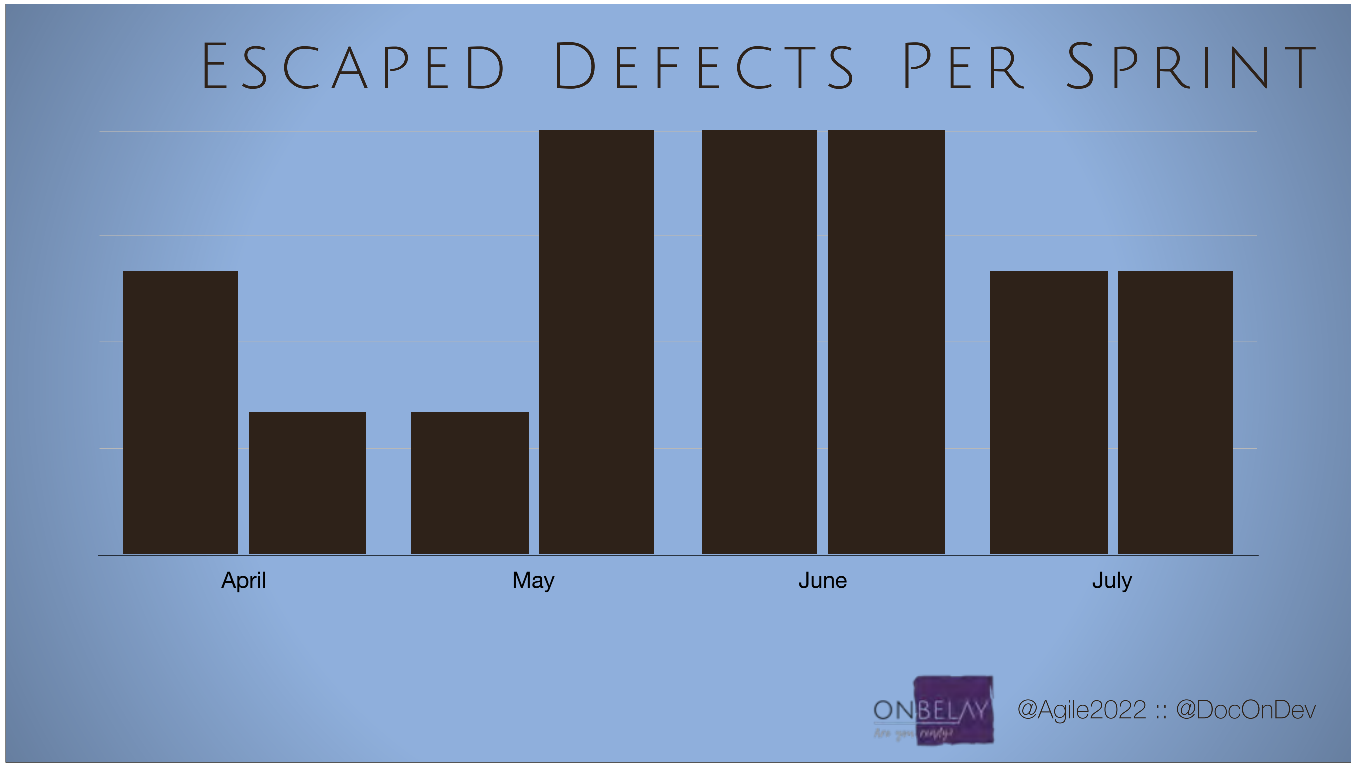
If it is in Development or UAT, it is not an escaped defect.



Total Escaped Defects

ONBELAY @Agile2022 :: @DocOnDev

This is helpful. We can see a trend here and forecast into the future. This can inform our remediation efforts.



Escaped Defects Per Sprint

ONBELAY @Agile2022 :: @DocOnDev

We can also look at defects released per sprint.

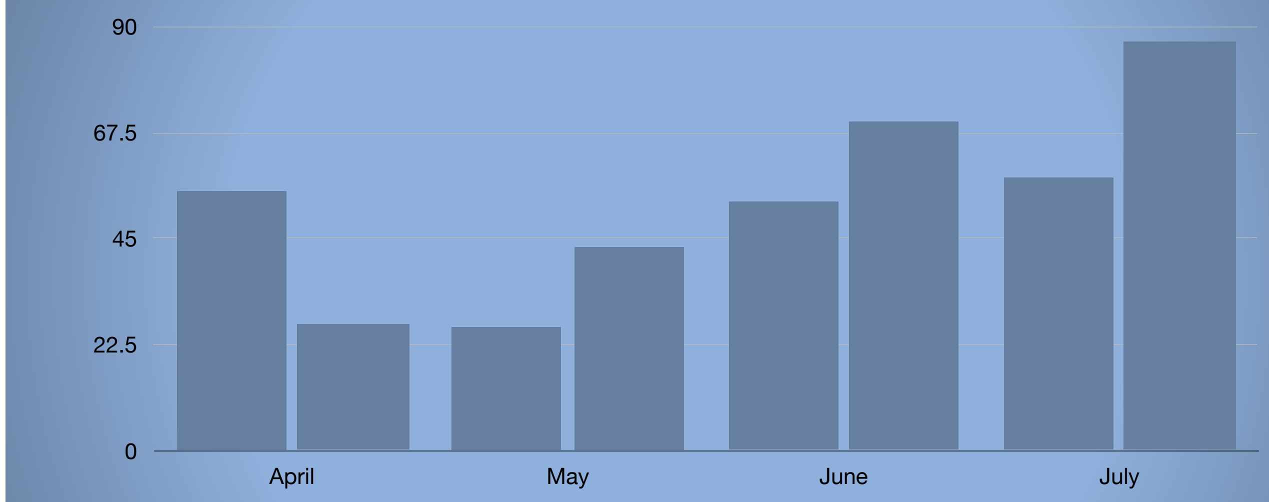
Both total and defects per iteration are helpful, but they both can lead us to invalid conclusions.

Total might be getter better because we’re cleaning them up, while the number of defects introduced us actually going up each iteration.

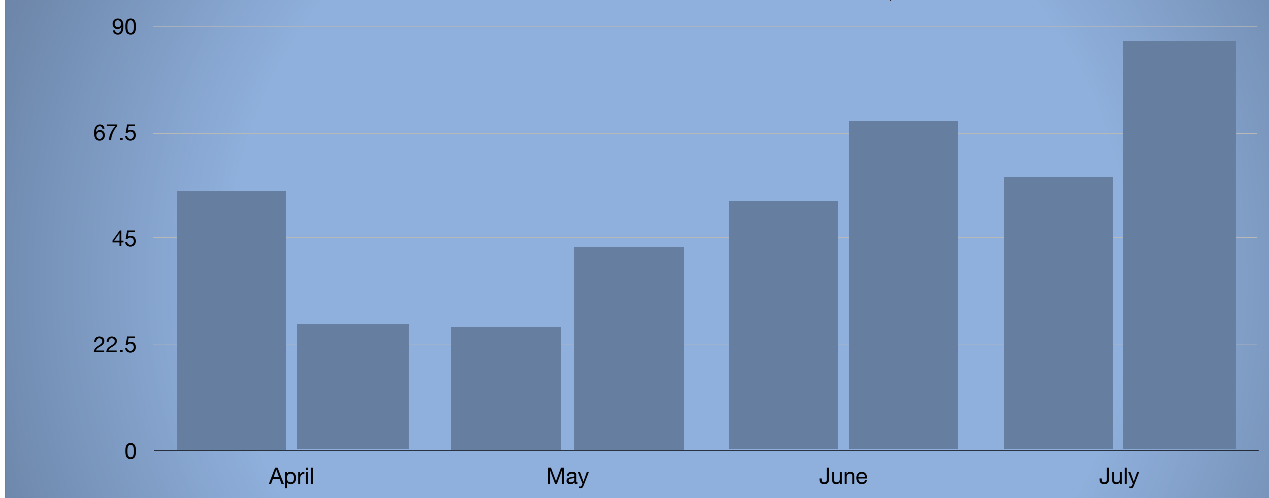
Count per iteration may be dropping, but our throughput might be dropping faster, meaning we’re actually not doing as well.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare iteration to iteration a bit better for trending.

Throughput

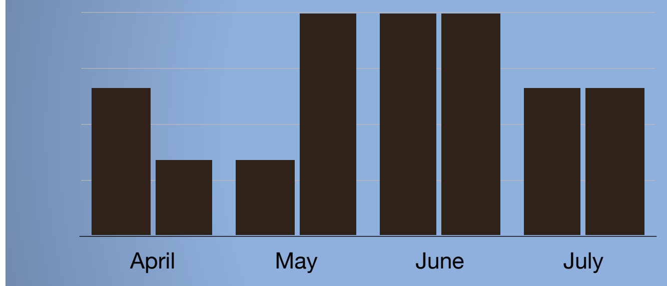


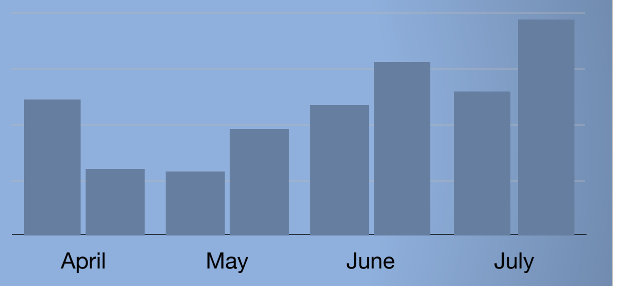
Throughput (Velocity)



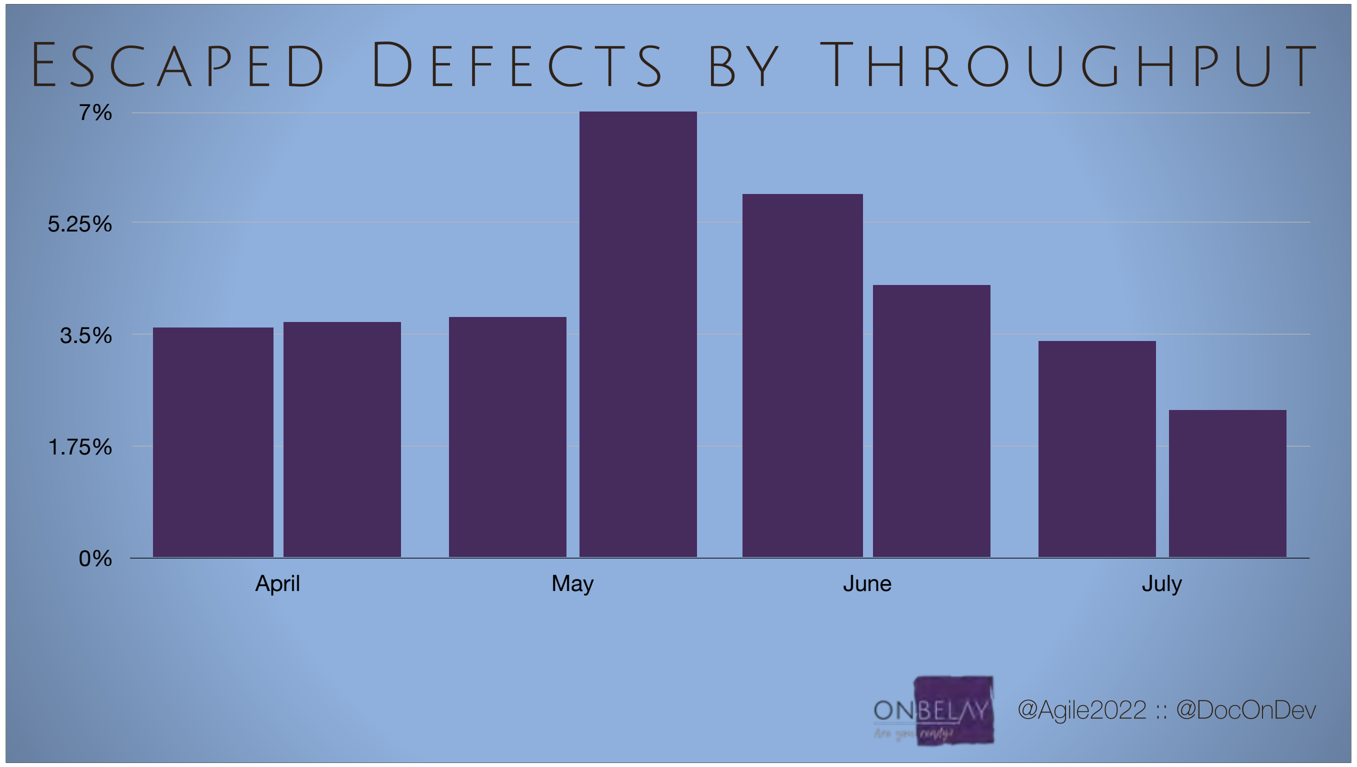
ONBELAY @Agile2022 :: @DocOnDev

Defects: Sprint & Throughput





@Agile2022 :: @DocOnDev



Escaped Defects by Throughput

@Agile2022 :: @DocOnDev

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare iteration to iteration a bit better for trending.

Code Compliance

Code Compliance

Adherence to coding standards

@Agile2022 :: @DocOnDev

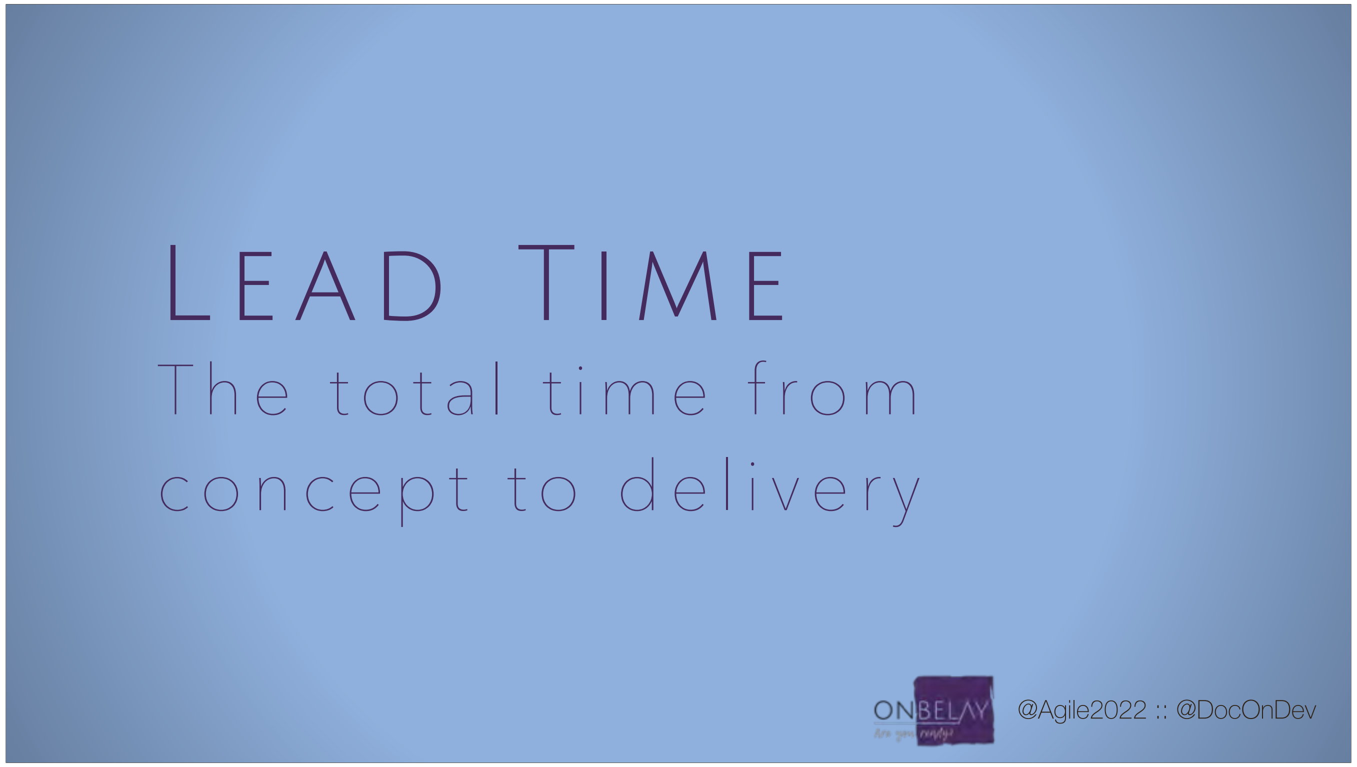
This is a way of using static analysis to ensure code meets certain standards

- Indentation / Formatting
- Method length
- Parameter Count
- Code Duplication

Lead Time & Flow Time

@Agile2022 :: @DocOnDev

Lead Time and Flow Time are the same concept



Lead Time

The total time from concept to delivery

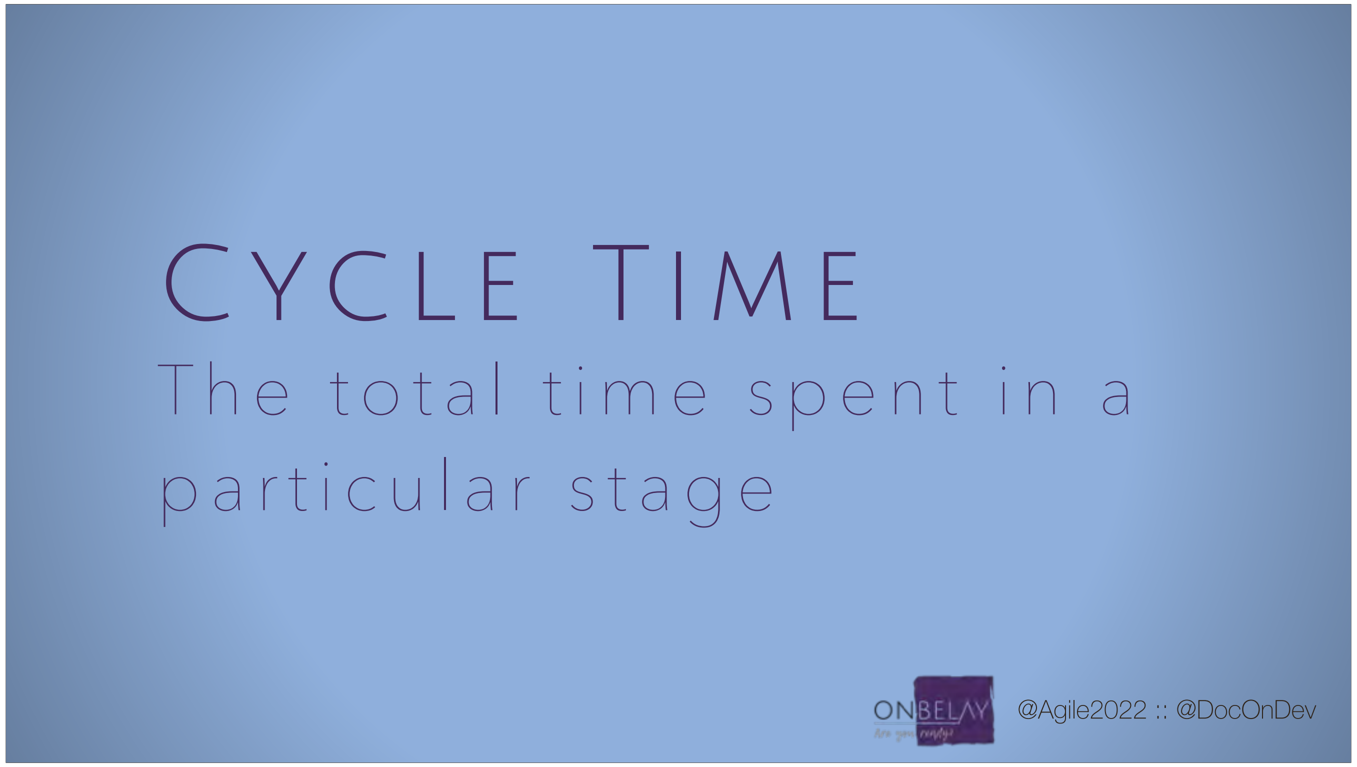
@Agile2022 :: @DocOnDev

Lead time is often considered the time from when a story enters the backlog to when it is in production.
Lead time includes all stages and wait states.

Lead Time helps organizations understand how quickly they can deliver software. It gives you a sense of the efficiency of the teams. Shorter lead times enable faster feedback on what is getting built and allows for quicker course correction. Conversely, longer lead times signify bottlenecks in the process.

When you have long lead times, you need to look at the cycle time and wait states to figure out where to focus.

Cycle Time



Cycle Time

The total time spent in a particular stage

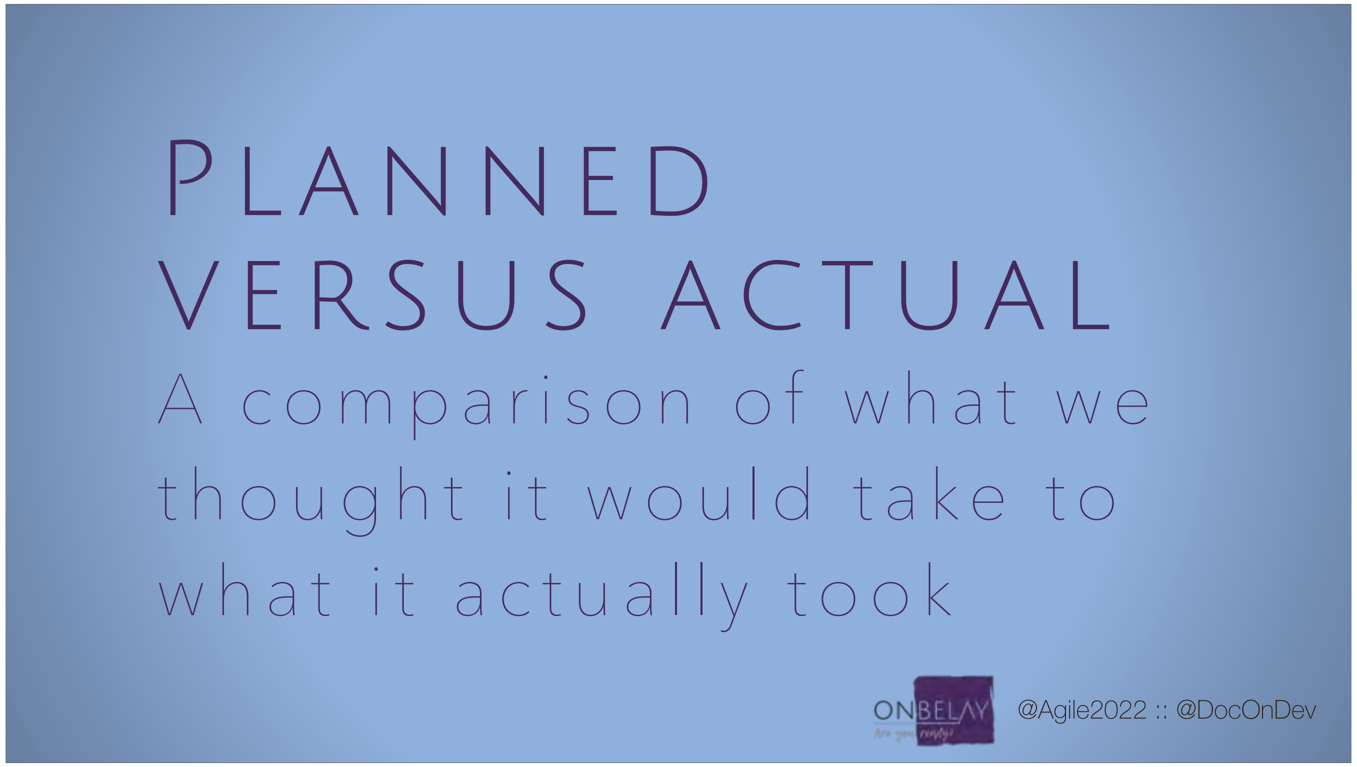
@Agile2022 :: @DocOnDev

Cycle Time is often considered the total time a story spends in a stage.

For example - an environment where discovery, design, development, testing, and deployment are the stages. Lead Time would be how long an item takes to get through all of the stages. And the time spent in development would be the development cycle time.

Cycle time helps us understand how long any given item might spend in a given stage. Stages with longer or highly variable lead times indicate bottlenecks. Optimizing any stage other than the bottleneck will have minimal impact on overall delivery. Focus on the bottleneck. Optimizing in front of the bottleneck will overload the bottleneck, slowing it down more. Optimizing after the bottleneck will provide no improvement overall.

Planned versus Actual & Flow Predictability



PLANNED VERSUS ACTUAL

A comparison of what we thought it would take to what it actually took

ONBELAY @Agile2022 :: @DocOnDev

Organizations need to be careful when they do this.

The purpose of this metric is to inform your planning, It is NOT to figure out what the team did wrong.

Plans are a guess
Actuals are reality
The intent is not to bend reality, but to make better guesses.

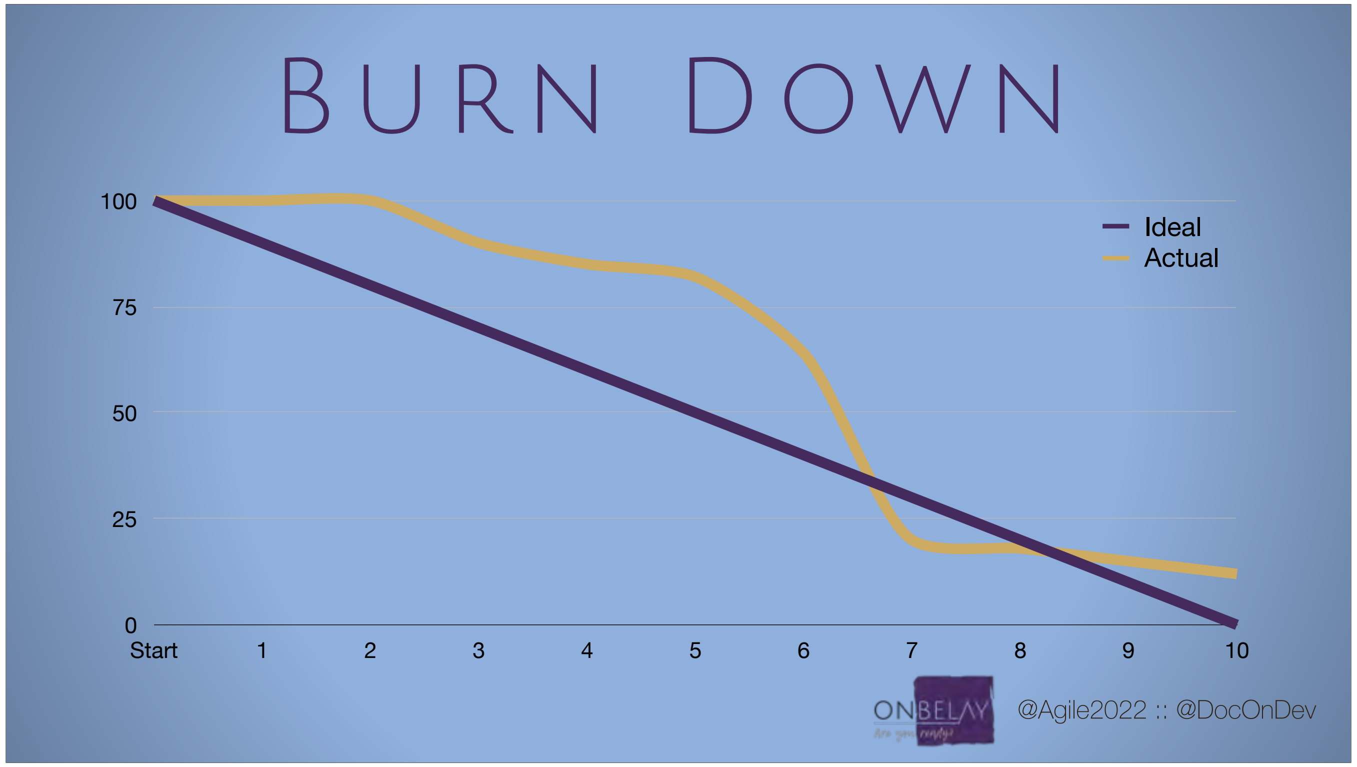
Burn Up & Burn Down

Burn Up & Burn Down

A visual means of tracking progress toward a goal

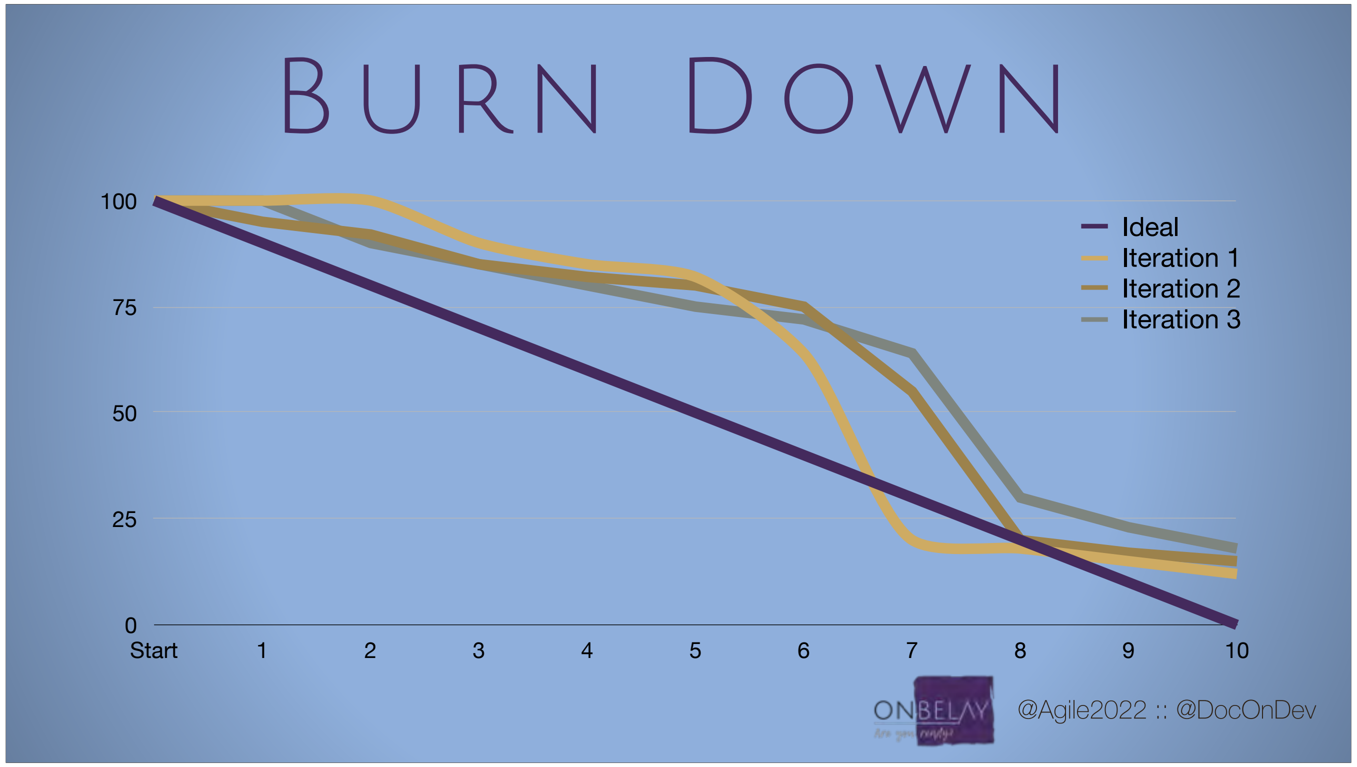
ONBELAY @Agile2022 :: @DocOnDev

Burn up and burn down are both simple ways of tracking progress toward a goal. We might use a burn chart within an iteration to see how we are tracking toward completion of the anticipated work. We might use a burn chart to see how we are progressing toward completion of a feature or release.



Burn Down

The intent is to inform us early so that we can make adjustments to the plan. The intent is NOT to inform us so that we can put pressure on the system to extract more work or force compliance to a commitment.

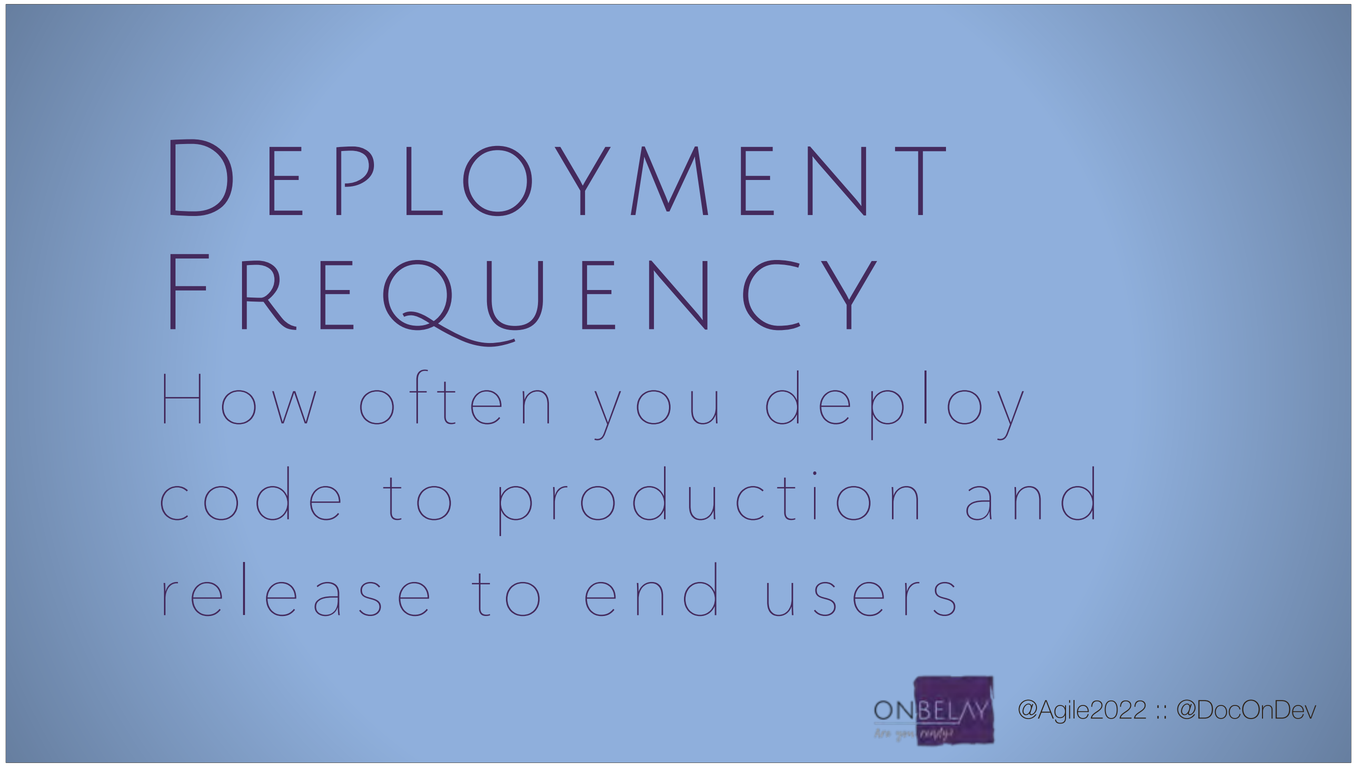


Burn Down

@Agile2022 :: @DocOnDev

Looking at burn charts over time can expose patterns and give us insight into issues in the system.

Deployment Frequency



Deployment Frequency

How often you deploy code to production and release to end users

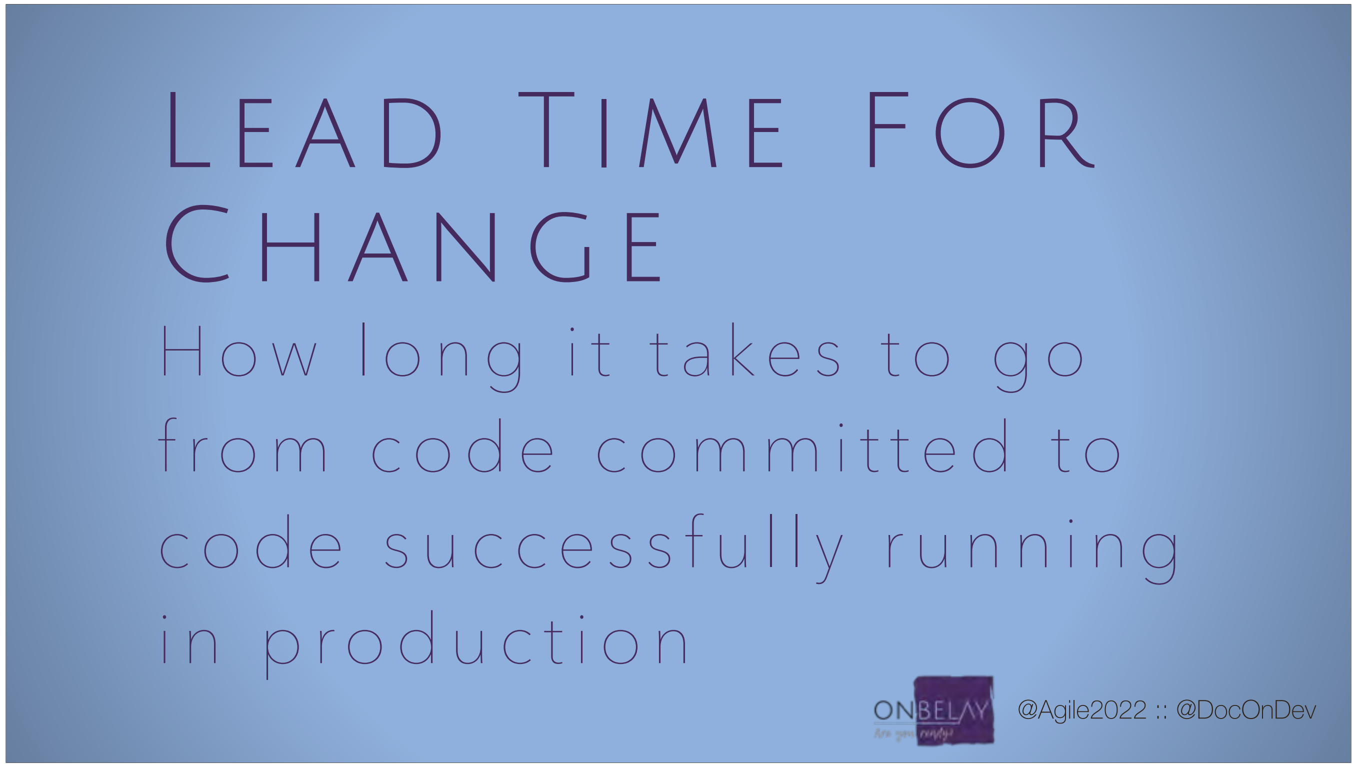
ONBELAY

@Agile2022 :: @DocOnDev

One of the DORA (DevOps Research & Assessment) Metrics

Expressed as count per time period
Once per six months - low performing
Between once per month and once per six months - medium performing
Between once per week and once per month - high performing
Multiple times per day - elite

Lead Time For Change



Lead Time For Change

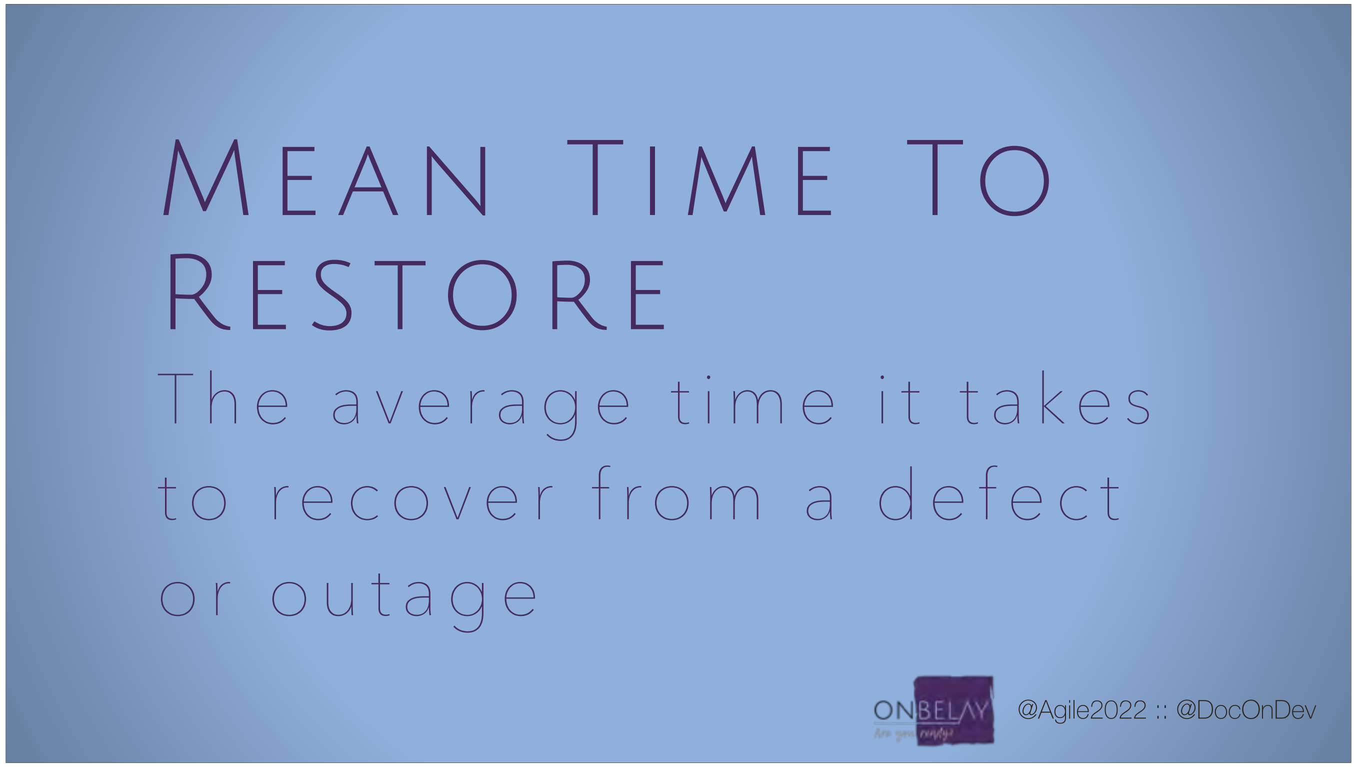
How long it takes to go from code committed to code successfully running in production

ONBELAY @Agile2022 :: @DocOnDev

One of the DORA (DevOps Research & Assessment) Metrics

More than six months - low performing
Between one and six months - medium performing
Between one day and one week - high performing
Less than one hour - elite

Mean Time To Restore



Mean Time to Restore

The average time it takes to recover from a defect or outage

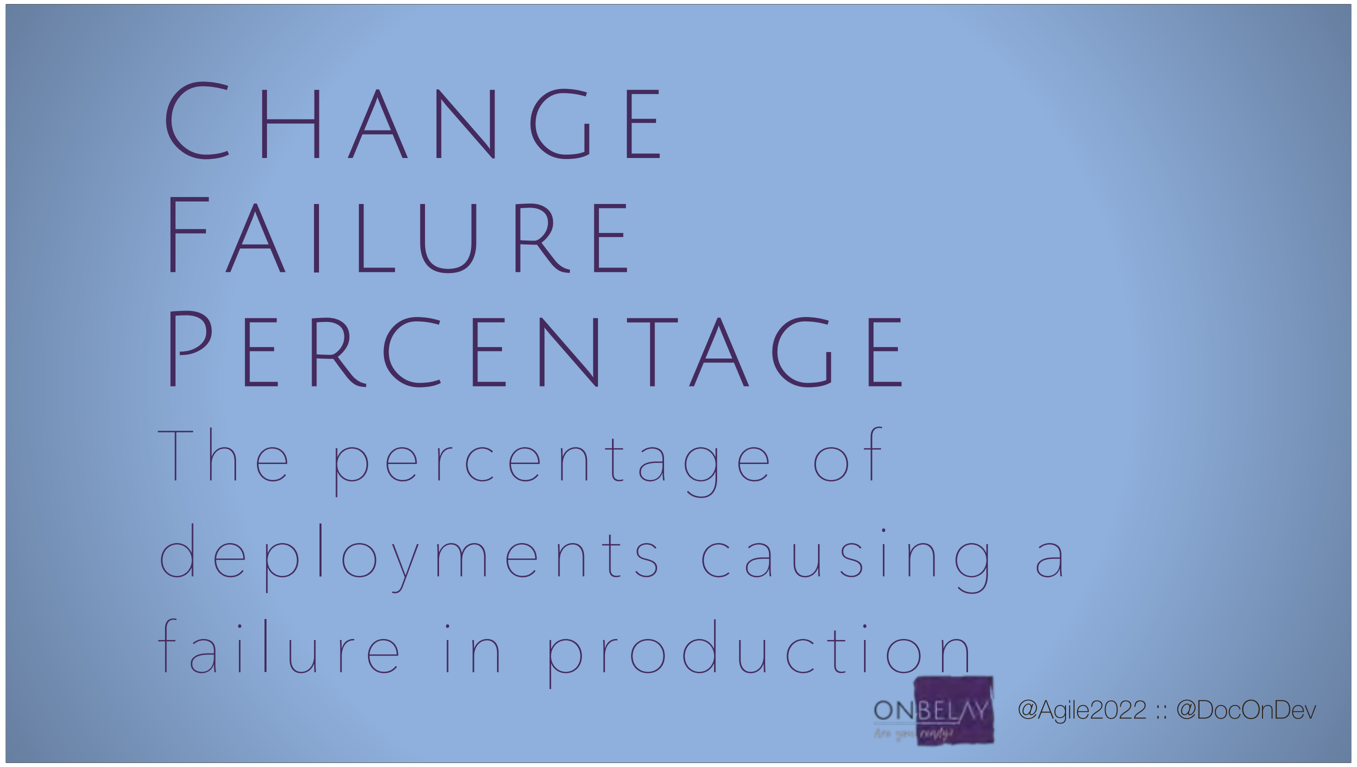
@Agile2022 :: @DocOnDev

One of the DORA (DevOps Research & Assessment) Metrics

Many places see defects and outages as separate from one another. But in a high-functioning environment, defects are considered outages - The service is sub-standard and needs to be brought back to standard.

This is an indicator of overall solution quality as well as the effectiveness and efficiency of the team.

Change Failure Percentage



Change Failure Percentage

The percentage of deployments causing a failure in production

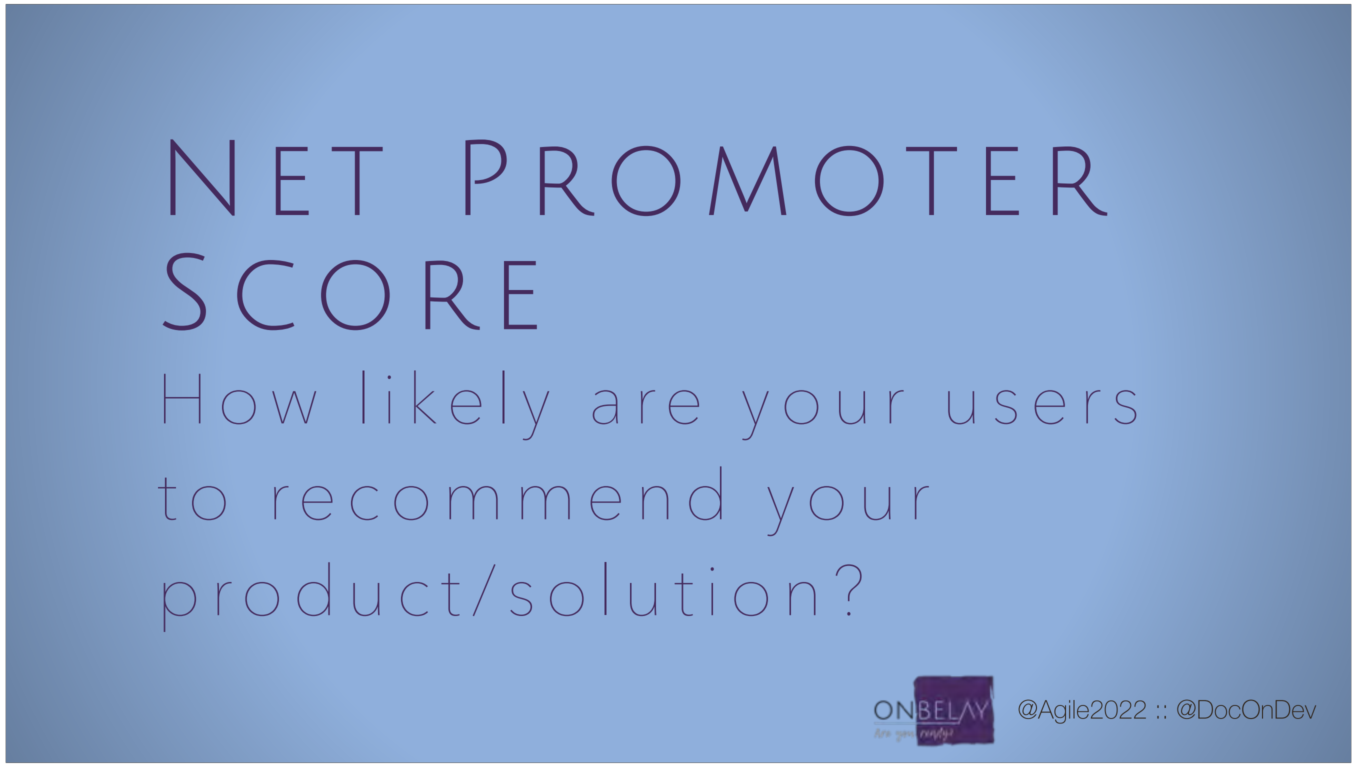
@Agile2022 :: @DocOnDev

One of the DORA (DevOps Research & Assessment) Metrics

The measure of the number of times “a hotfix, a rollback, a fix-forward, or a patch” is required after a software deployment or a service change.

Keep in mind - this number can be misleading in a lower-performing environment. Some places are so accustomed to defects in production, that they actually have CLASSES of defects and will NOT issue a hot-fix for many of their defects. These environments may not count anything but a critical defect in their change failure percentage, giving them a false sense of quality.

Net promoter Score



Net Promoter Score

How likely are your users to recommend your product/solution?

ONBELAY @Agile2022 :: @DocOnDev

NPS is a measure of customer satisfaction and considered a leading indicator of growth.

On a scale of 1-10, How likely is it that you would recommend [brand] to a friend or colleague?

Customers who answer
9 or higher are “promoters”
7 or 8 are “neutrals”
6 or lower are “detractors”

NPS is calculated by taking the percentage of promoters and subtracting the percentage of detractors.

A score can range from -100 to 100, with a higher score being more positive. Scores below 0 mean more people are detractors than promoters.

Typical scores fall between -1 and +50

Netflix	64
PayPal	63
Amazon	54
Google	53
Apple	49

Flow Efficiency

Flow Efficiency

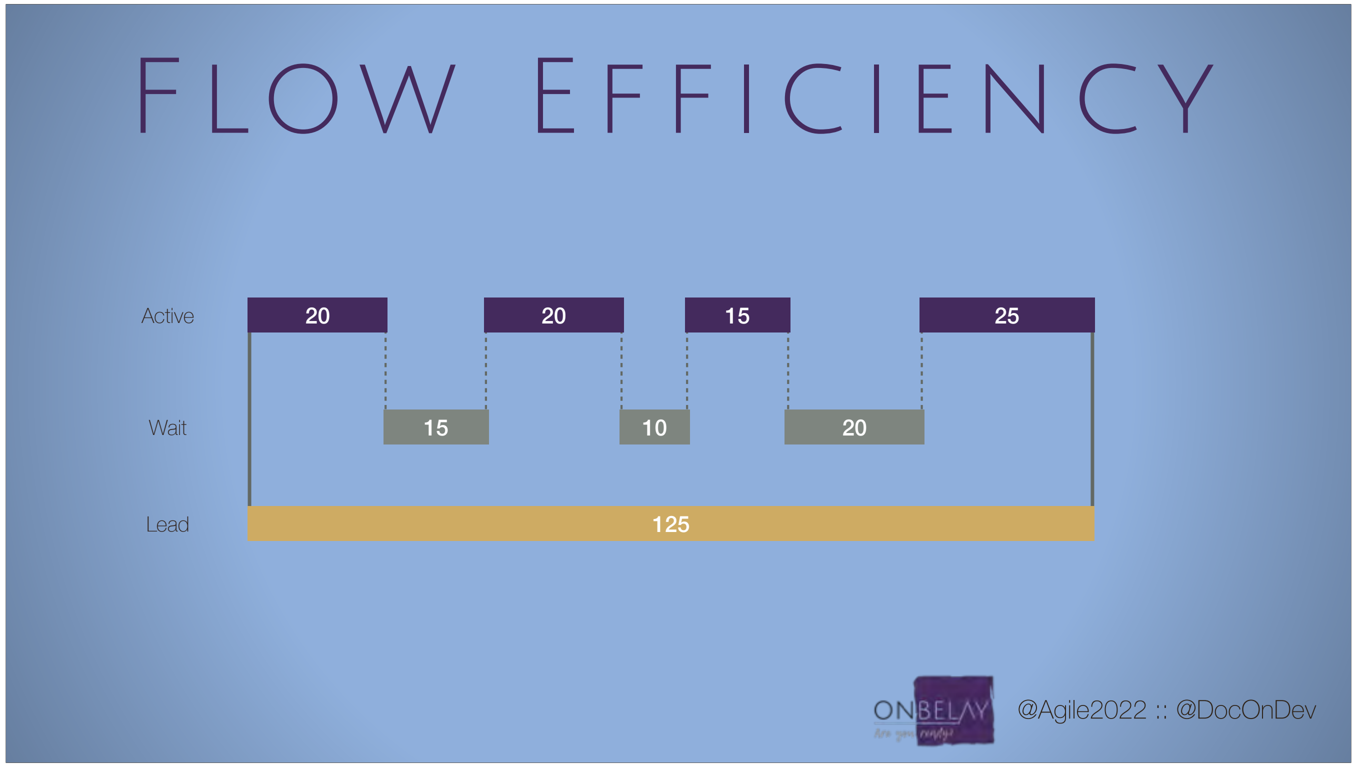
Ratio of active working time to Lead Time

ONBELAY @Agile2022 :: @DocOnDev

Flow Efficiency is a way of highlighting the delays in a process.

Map out your wait states as well as active states.

Flow efficiency is the time spent in active states divided by the lead time.



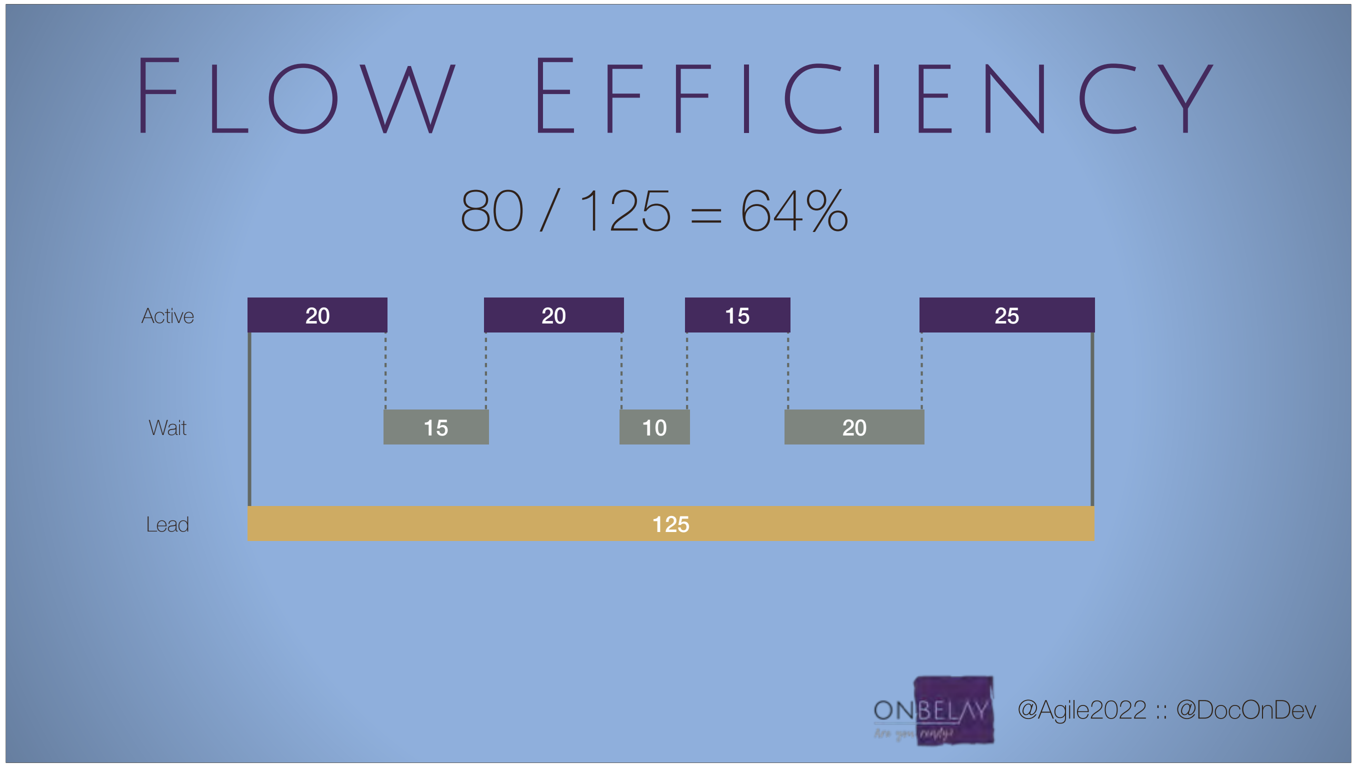
Flow Efficiency

@Agile2022 :: @DocOnDev

Flow Efficiency is a way of highlighting the delays in a process.

Map out your wait states as well as active states.

Flow efficiency is the time spent in active states divided by the lead time.



Flow Efficiency

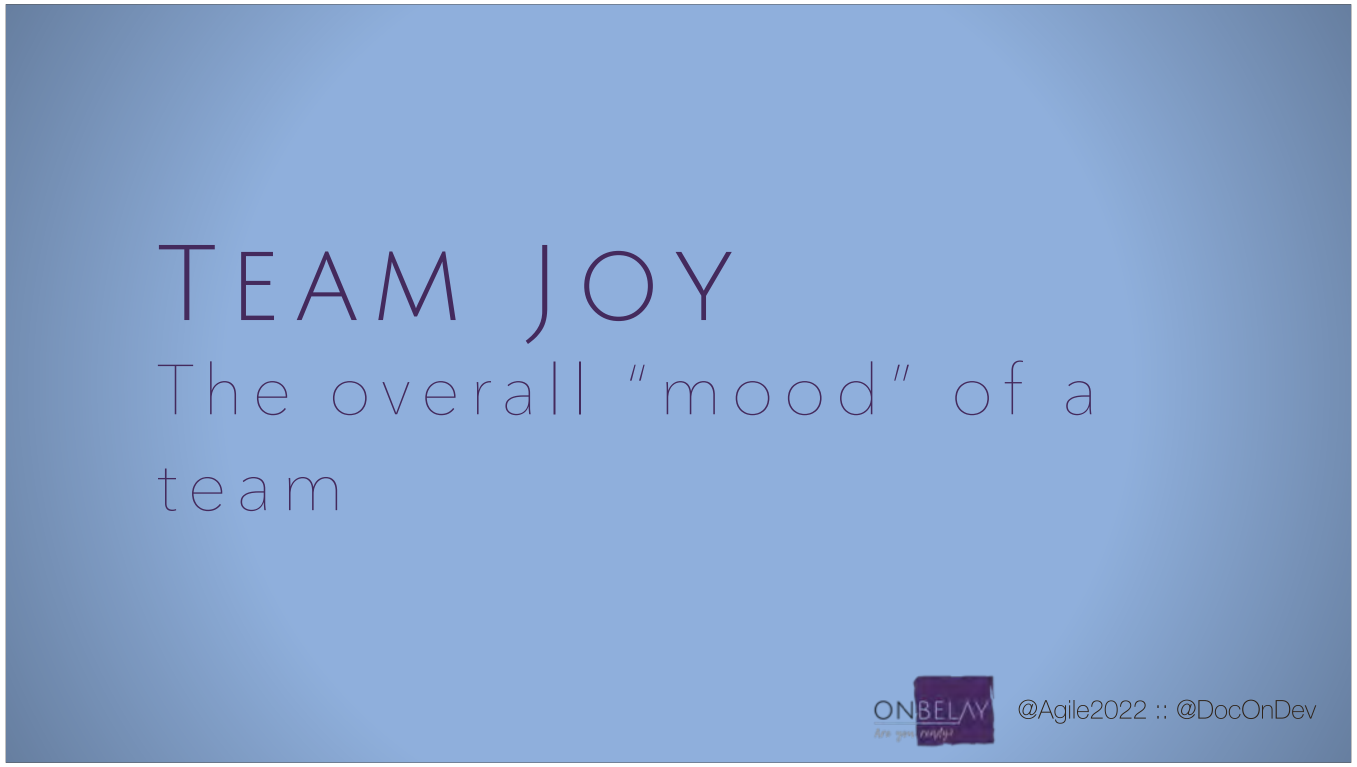
80 / 125 = 64%

ONBELAY @Agile2022 :: @DocOnDev

In this case, the flow efficiency is 64%

This is pretty good. In a lot of systems, flow efficiency can be as low as single digits. Often, focusing on making the work go faster can have less impact than reducing the wait states.

Team Joy



Team Joy

The overall "mood" of a team

ONBELAY @Agile2022 :: @DocOnDev

Team Joy is something we are still researching in the industry.

The idea here is to take regular samples of how a team is feeling about the work and the work environment. These should be on a short time scale, say weekly or even daily.

Use lightweight measurement techniques.

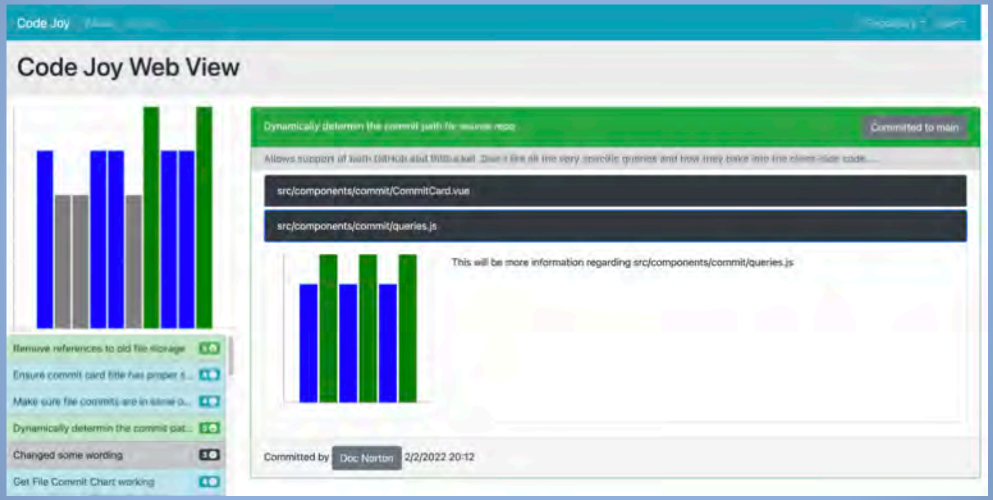
Niko Niko Calendars

Team Joy

@Agile2022 :: @DocOnDev

Have people rate their mood each day. This can be done anonymously, if the team needs that extra level of safety.

Team Joy



Code Joy

Code Joy Web View

Dynamically determin the commit path for source repo

Committed to main

Allows support of both GitHub and Bitbucket. Don't like all the very specific queries and how they bake into the client-side code....

src/components/commit/CommitCard.vue

src/components/commit/queries.js

This will be more information regarding src/components/commit/queries.js

Committed by Doc Norton 2/2/2022 20:12

Remove references to old file storage

Ensure commit card title has proper s...

Make sure file commits are in same o...

Dynamically determin the commit pat...

Changed some wording

Get File Commit Chart working

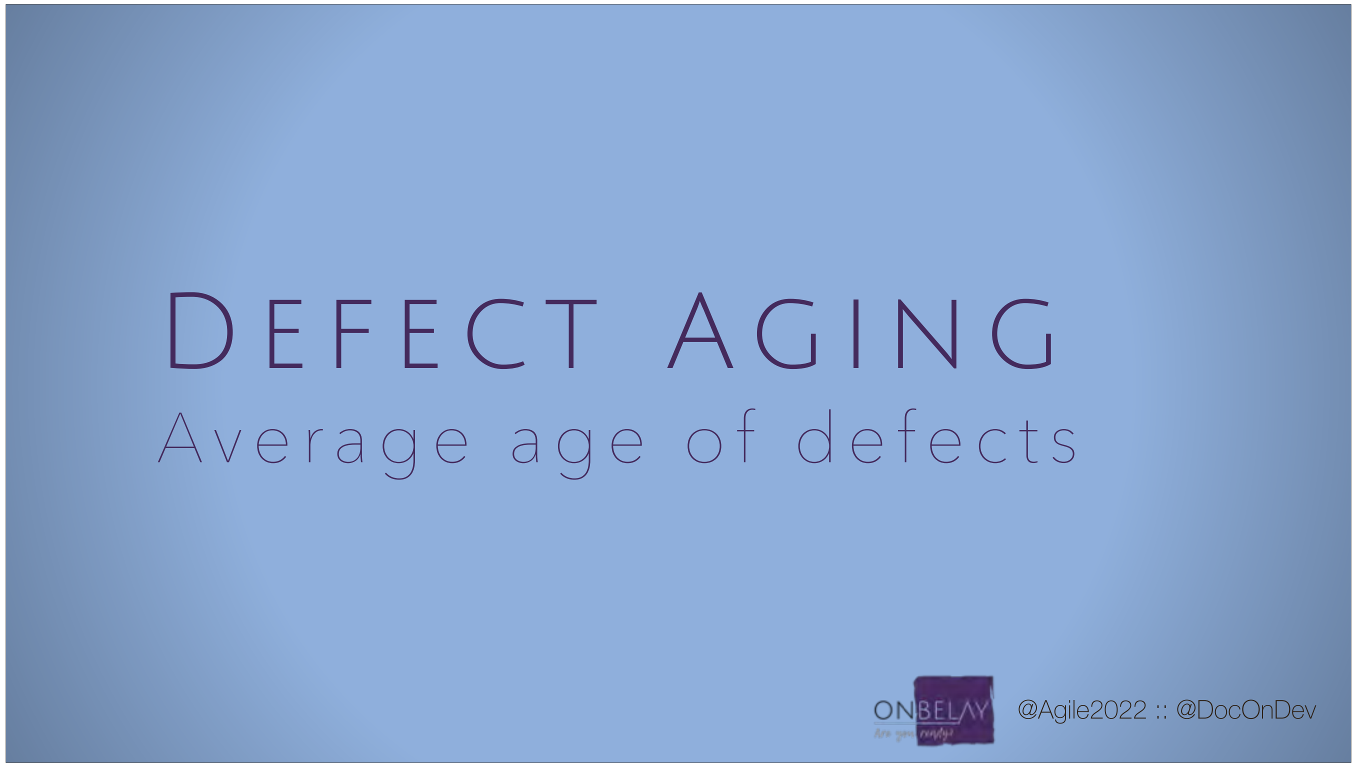
https://code-joy.app/ ONBELAY @Agile2022 :: @DocOnDev

This is a simple open source tool (still under development) that measures developer joy at every code check-in.

We ran an experiment like this at Groupon several years back and we found that “Code Joy” was a leading indicator of other issues. When code joy was trending down, we usually saw drops in throughput or increases in defects and rework about a week later.

By paying attention to the code joy metric, teams were able to discuss issues early.

Defect Aging



DEFECT AGING

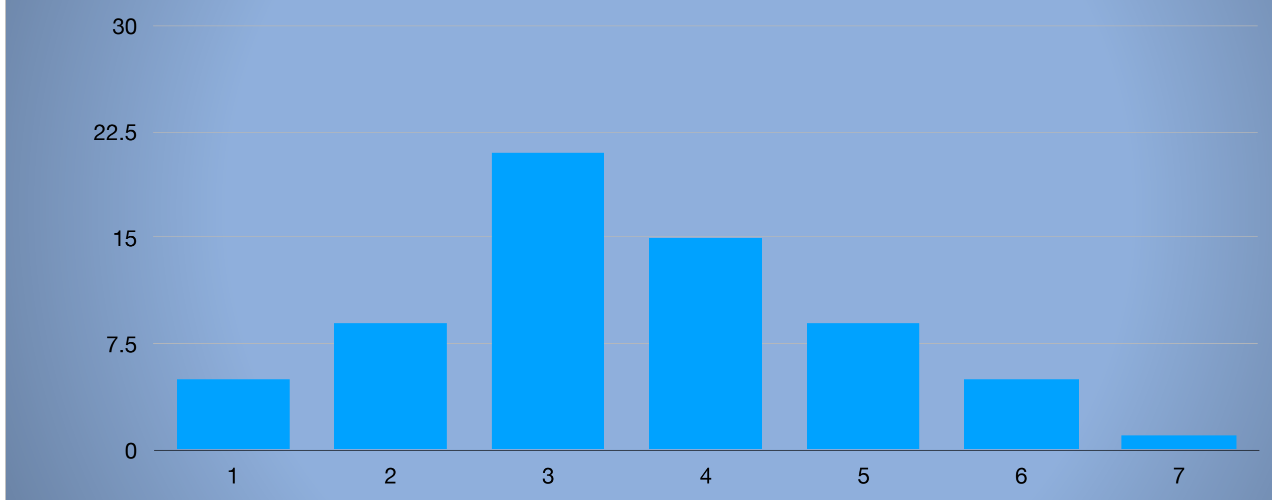
Average age of defects

@Agile2022 :: @DocOnDev

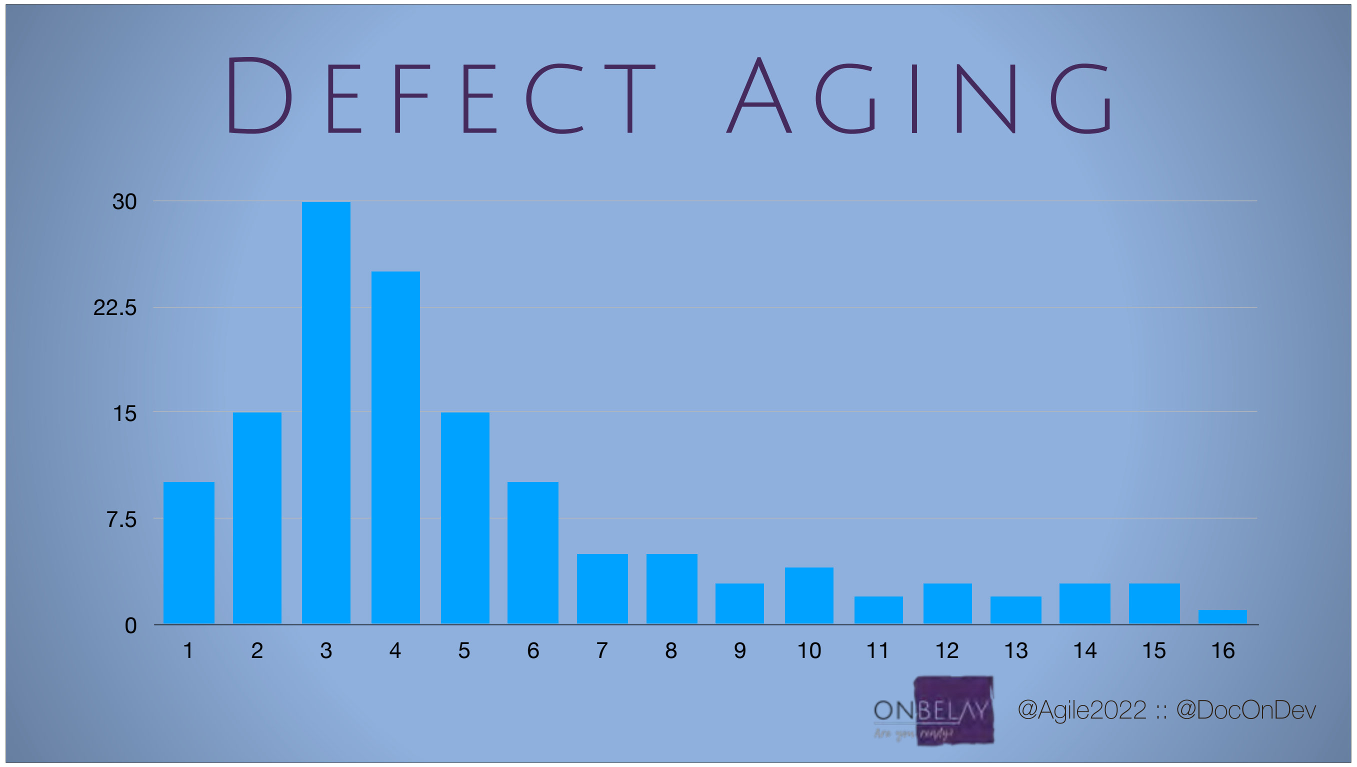
In a high functioning environment, this is similar to Mean Time To Recovery

Some organizations might want to look at this by severity of defect. That is usually an indicator that you have serious quality issues.

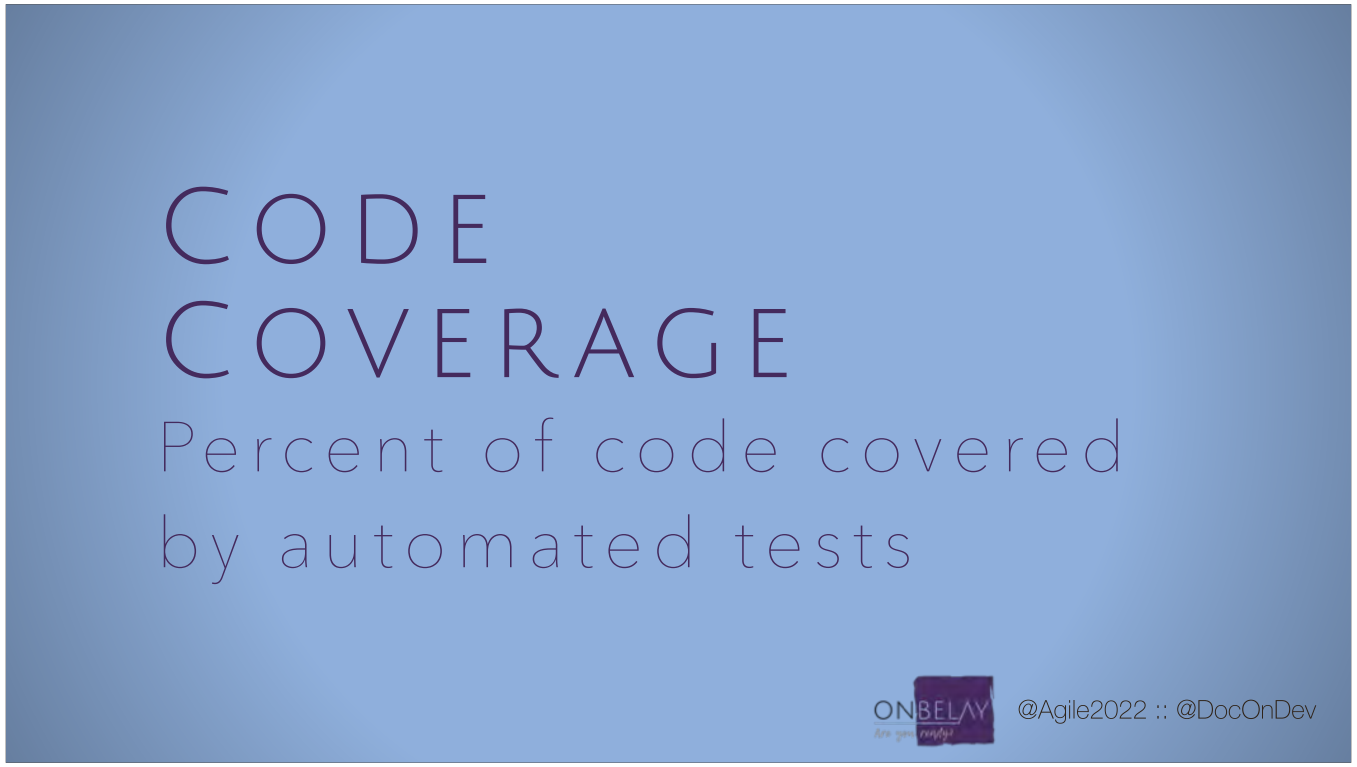
Defect Aging



Defect Aging



Code Coverage



Code Coverage

Percent of code covered by automated tests

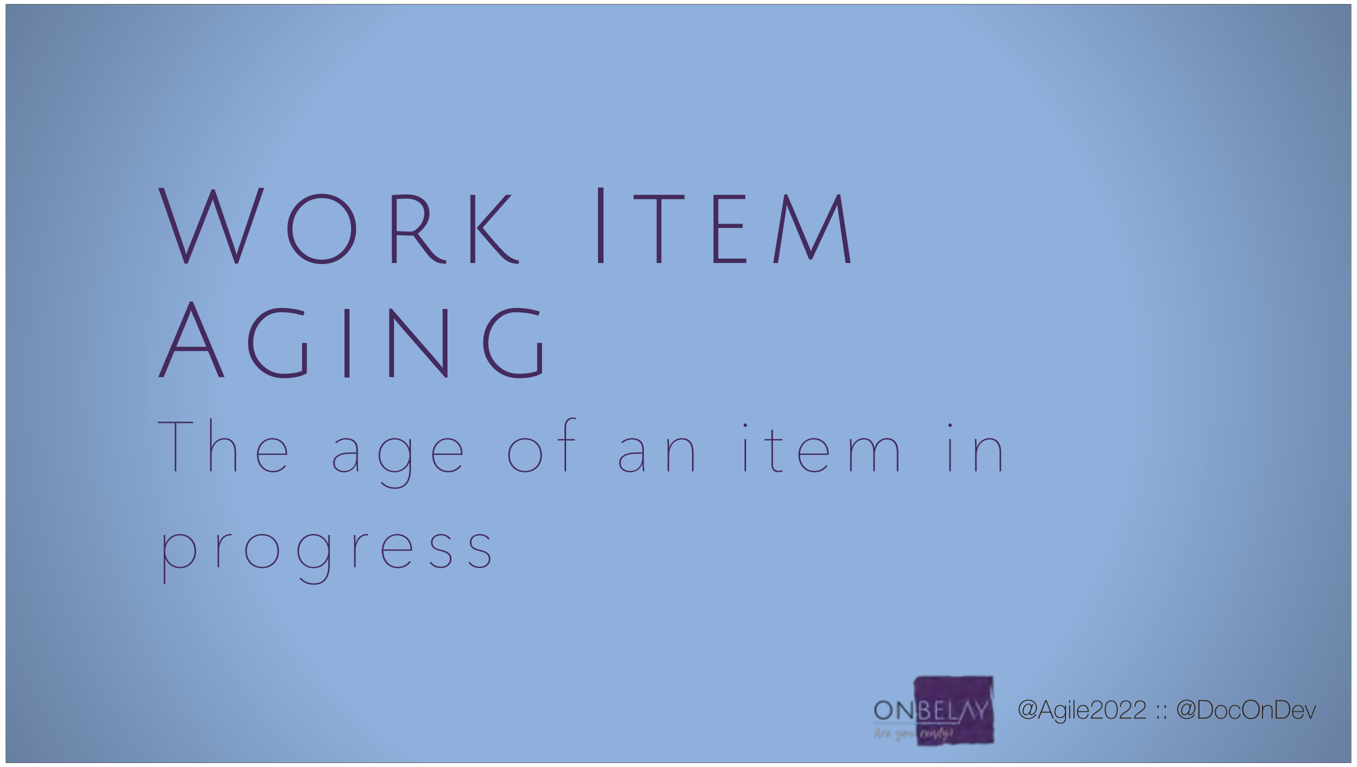
@Agile2022 :: @DocOnDev

Code coverage is an indirect indicator of quality.

Code coverage tools can help a team identify areas of the code that are not exercised by tests. In a Test first environment, this is extremely uncommon.

In some environments, you will see test coverage by type of test - so you'll have a report for unit tests and a separate report for acceptance tests. In other environments, you'll see one coverage report for the entire test suite, regardless of type of test.

Work Item Aging



Work Item Aging

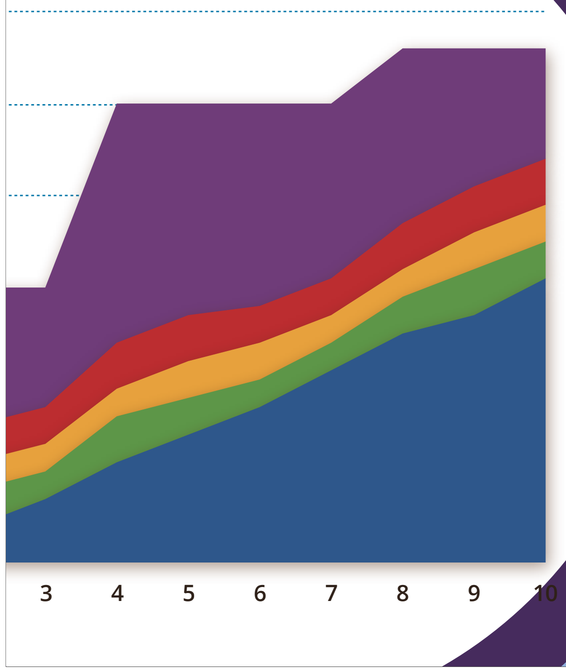
The age of an item in progress

ONBELAY @Agile2022 :: @DocOnDev

Work Item Aging is measured while an item is in progress. This is the elapsed time from the moment a work item was started until now. It includes active working time AND any idle time regardless of the cause.

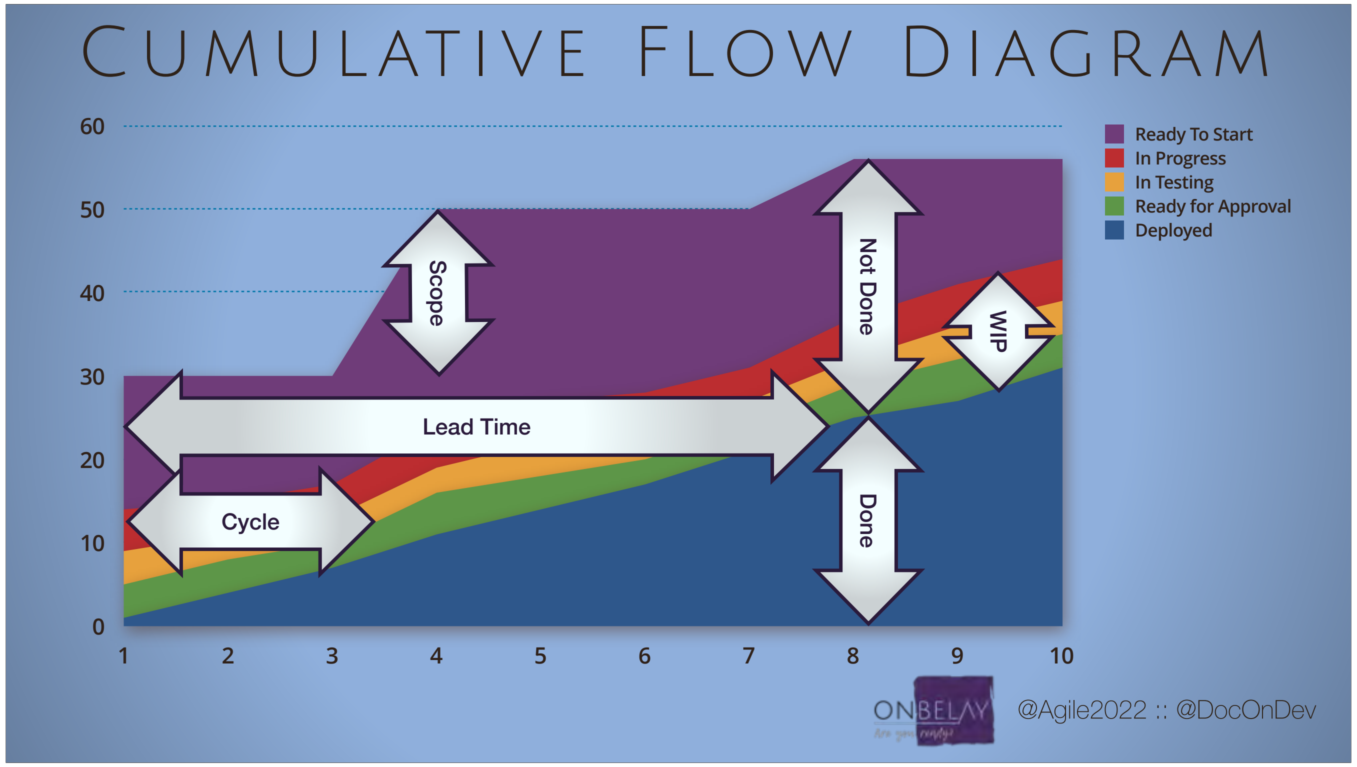
Work item age can highlight stories that need focus.

Cumulative Flow Diagram



Cumulative Flow Diagram

ONBELAY @Agile2022 :: @DocOnDev



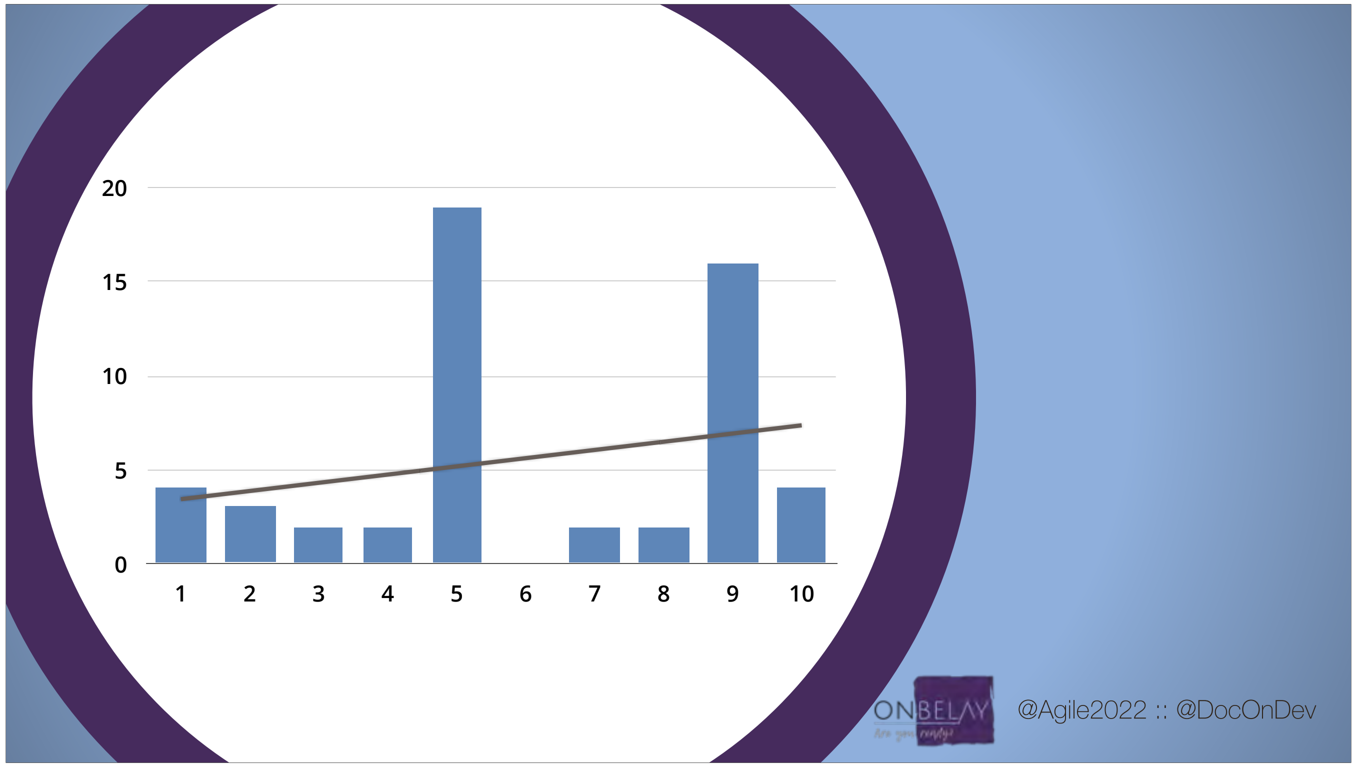
CUMULATIVE FLOW DIAGRAM

@Agile2022 :: @DocOnDev

Looking at this diagram, we can see <next>
work done and work not done <next>
The amount of Work in Progress at any given time <next>
The lead time - hey wouldn’t it be nice to hay, isn’t that nice <next>
And the cycle time - the amount of time an item is actually being worked on <next>

Finally, we can see changes in scope whenever our top line moves.

For a team that is operating well, this graph has relatively smooth lines that move together up and to the right

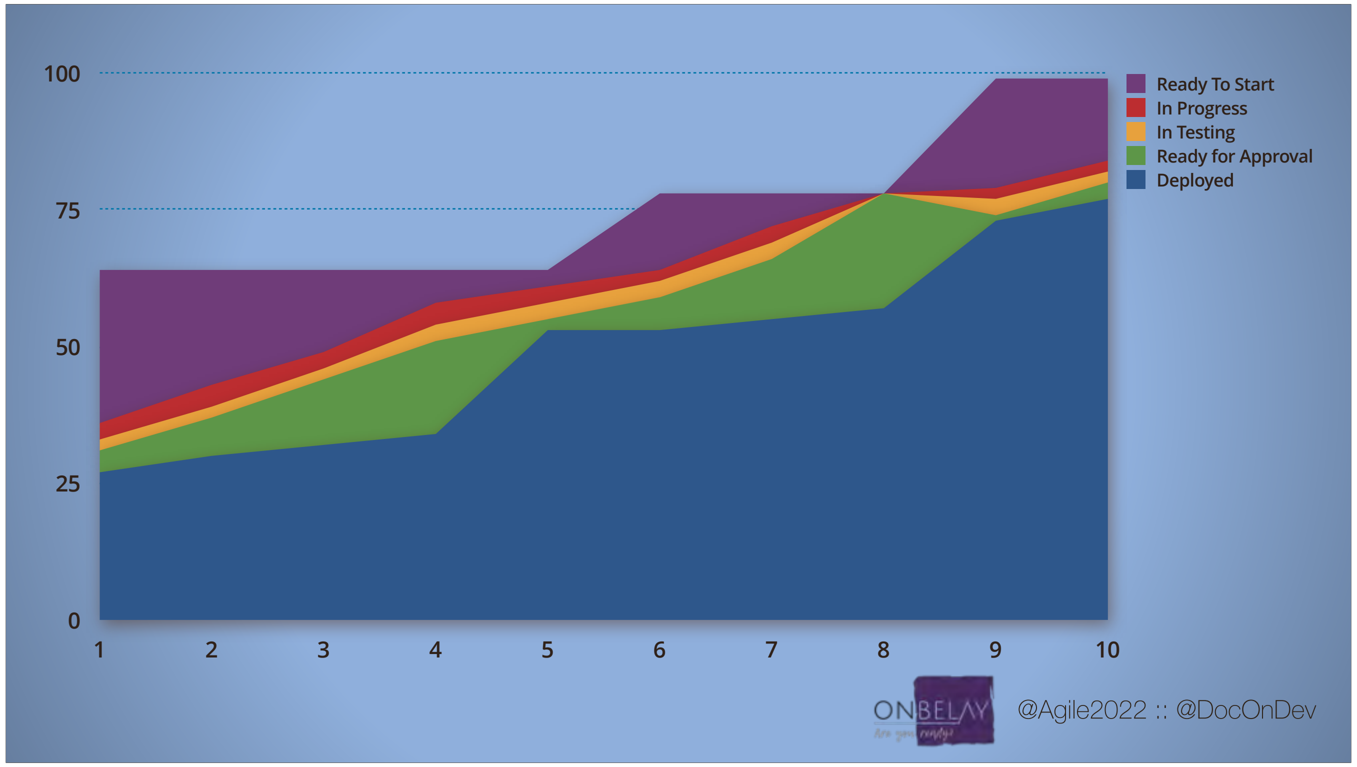


@Agile2022 :: @DocOnDev

Remember this chart from earlier?

We couldn’t say for sure what was the issue for this team.

Same Team
Same Data
Different Perspective



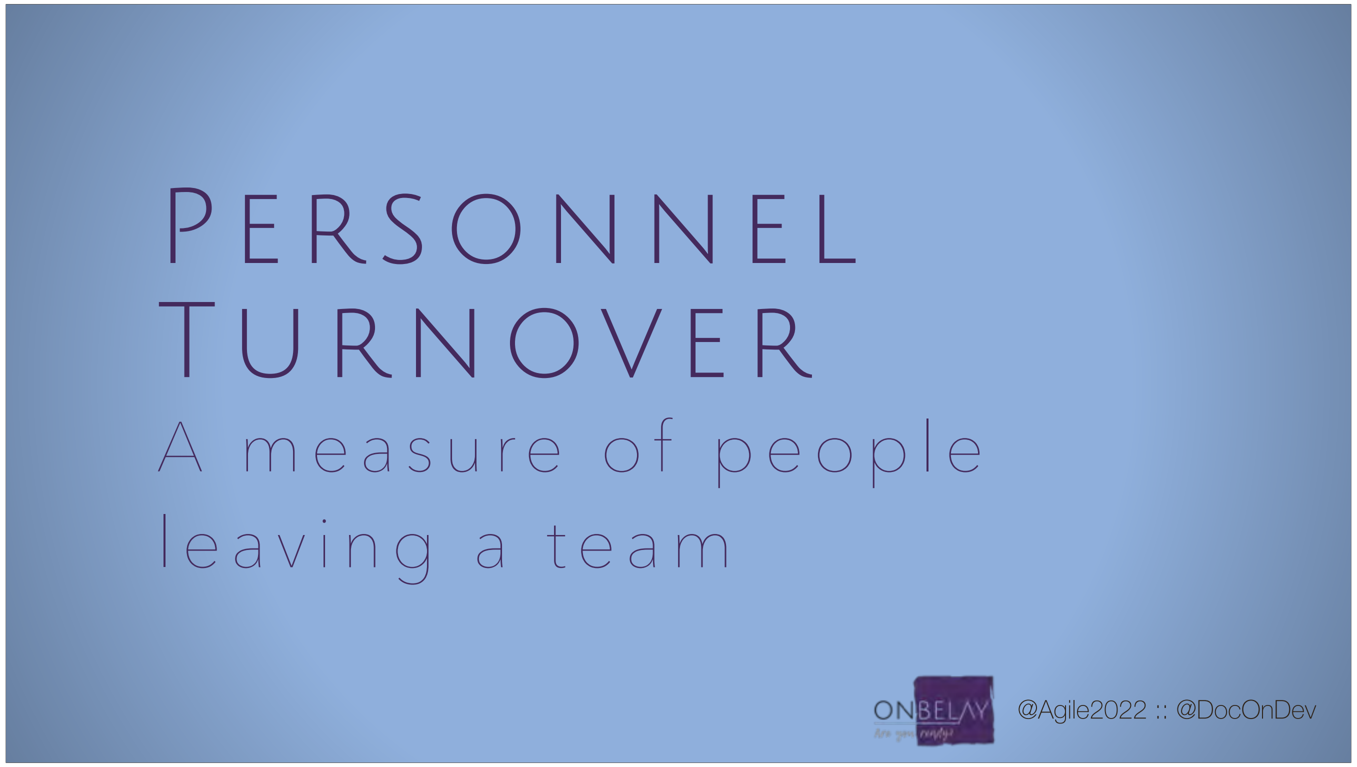
@Agile2022 :: @DocOnDev

Can anyone tell me what is “wrong” with this team?

Product owner is a traveling salesperson. On the road, doesn’t have time. Comes back and approves and then adds to the backlog.

Personnel Turnover

@Agile2022 :: @DocOnDev



Personnel Turnover

A measure of people leaving a team

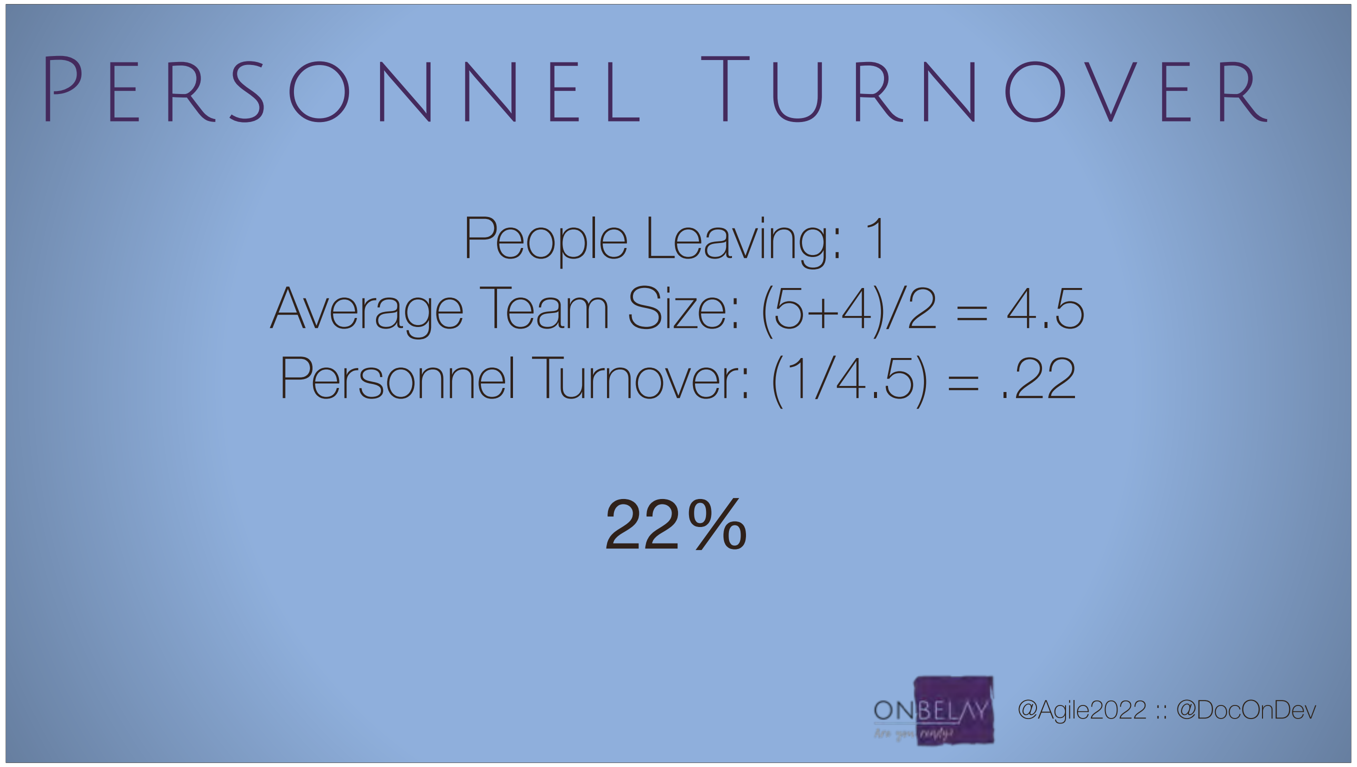
ONBELAY @Agile2022 :: @DocOnDev

Employee turnover is a ratio of the number of people who left a team in a given time period to the average size of the team during that same time period.

The average turnover for Technical Staff is around 20% annually, industry wide.

Depending on your environment, high turnover may be an indicator of issues - you’ll want to look into why folks are leaving. In other environments, perhaps where there is a more fluid staffing plan, team turnover is less of an indicator of issues.

You night also want to look at turnover in terms of leaving the department or organization rather than at the team level.



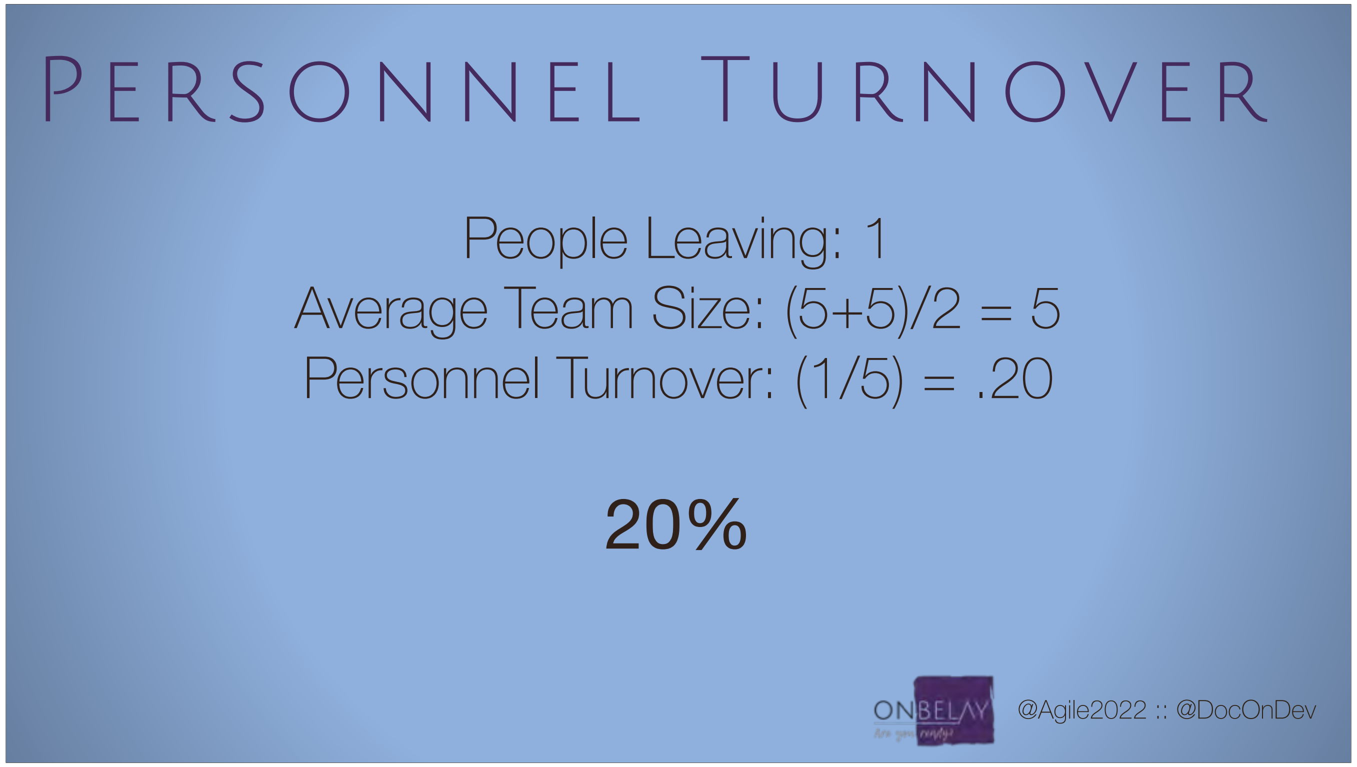
Personnel Turnover

People Leaving: 1
Average Team Size: (5+4)/2 = 4.5
Personnel Turnover: (1/4.5) = .22

22%

@Agile2022 :: @DocOnDev

Lets say in a given quarter that 1 person leaves a team of 5 and is not replaced, the turnover rate is 22% for that quarter.



Personnel Turnover

People Leaving: 1

Average Team Size: (5+5)/2 = 5

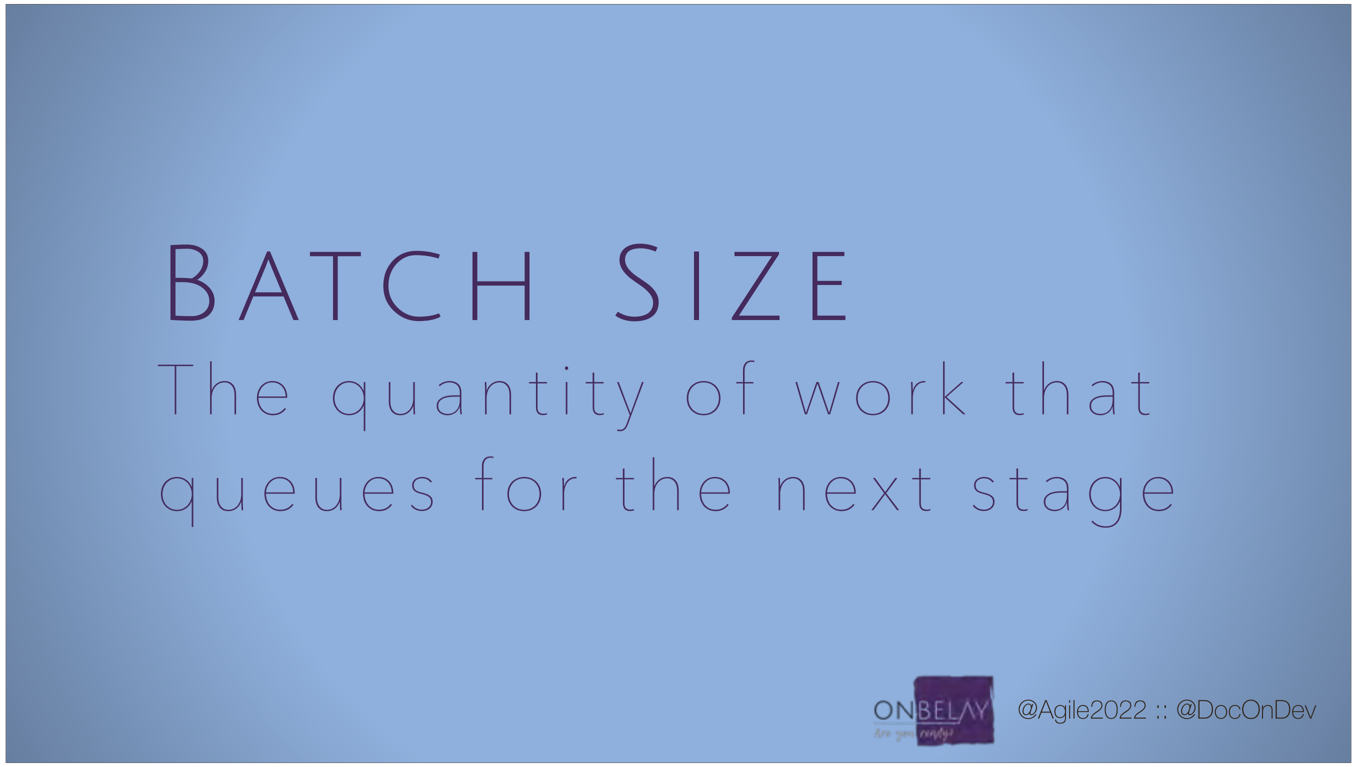
Personnel Turnover: (1/5) = .20

20%

@Agile2022 :: @DocOnDev

Lets say in a given quarter that 1 person leaves a team of 5 and is replaced in the same quarter, the turnover rate is 20% for that quarter.

Batch Size



BATCH SIZE

The quantity of work that queues for the next stage

@Agile2022 :: @DocOnDev

I do not LOVE this definition, and I am open to a different one.

The thing is, in manufacturing, batch sizes are pretty easy to see because they are about counts of items. In software, what constitutes a batch is a little less concrete.

BATCH SIZE

@Agile2022 :: @DocOnDev

Batches exist everywhere -
Each story is a batch of work
When stories are dependent on each other, they make up a batch
What gets tested together is a batch
Collections of stories like features or epics are a batch
Releases are a batch

The bigger the Batch Size, the greater the risk.

The bigger the batch, the greater the risk. The greater the risk, the more planning required. The more planning required, the more time it takes. The more time it takes, the bigger the batch.

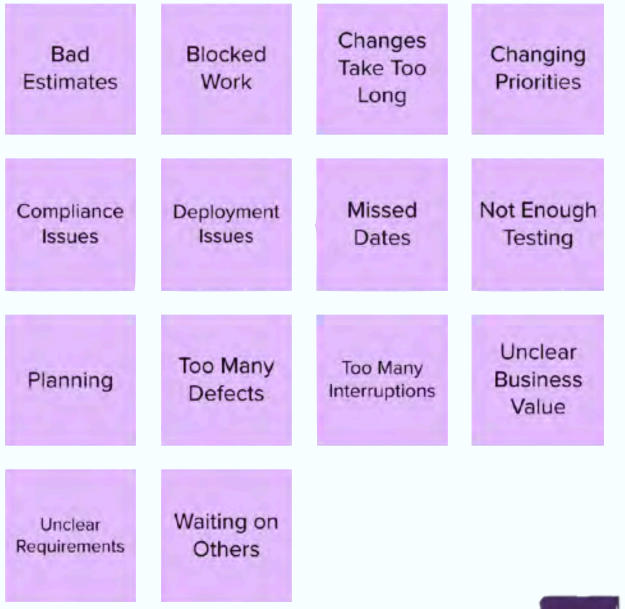
This is a vicious cycle

CHALLENGES?

@Agile2022 :: @DocOnDev

Select a couple of challenges from this list - what resonates with you in your environment?

Form Groups
Doc ask for a volunteer to list their top challenge - Anyone else who wants to work on that challenge join them



Bad Estimates

Blocked Work

Changes Take Too Long

Changing Priorities

Compliance Issues

Deployment Issues

Missed Dates

Not Enough Testing

Planning

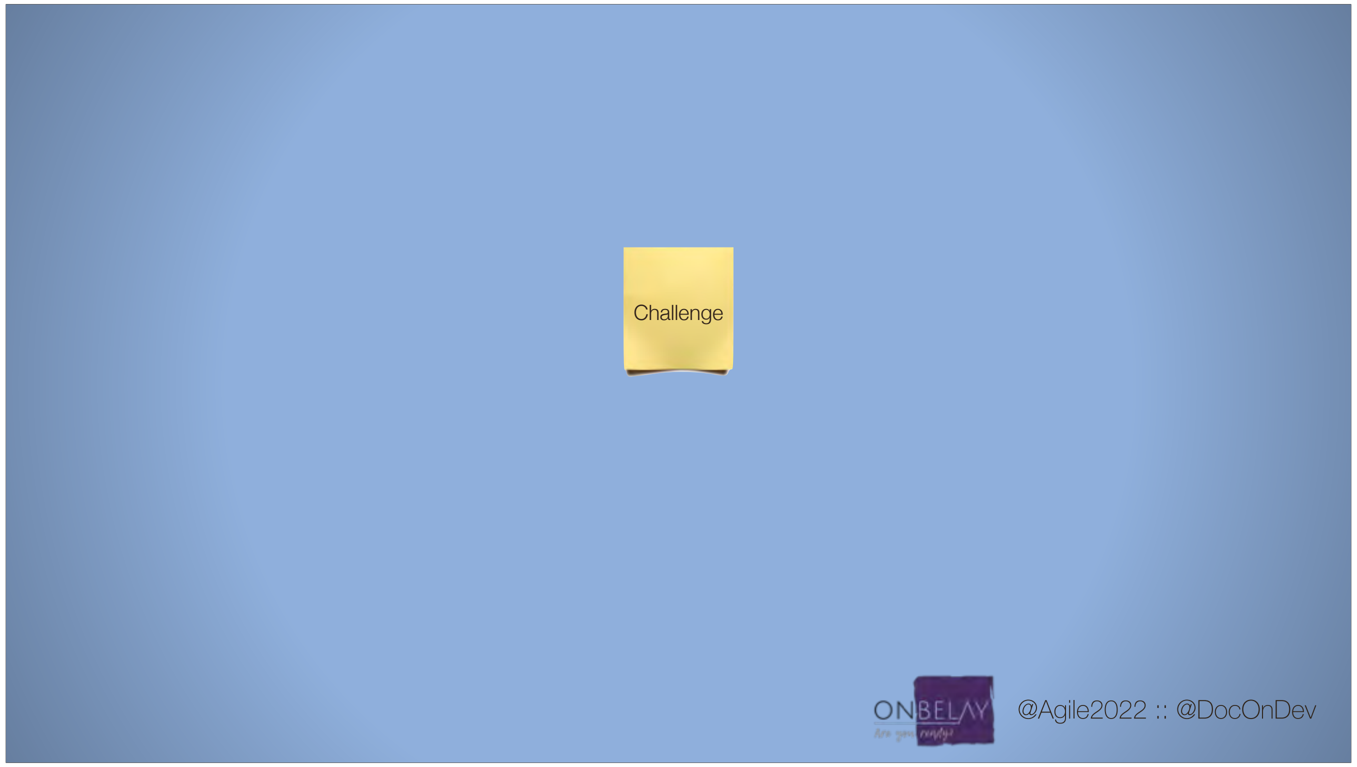
Too Many Defects

Too Many Interruptions

Unclear Business Value

Unclear Requirements

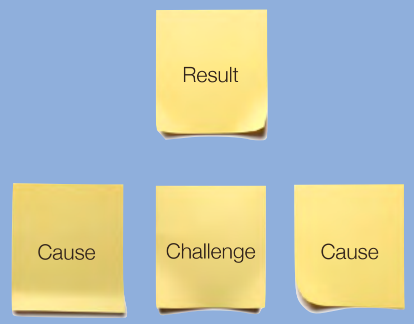
Waiting on Others



Challenge

Result

Challenge

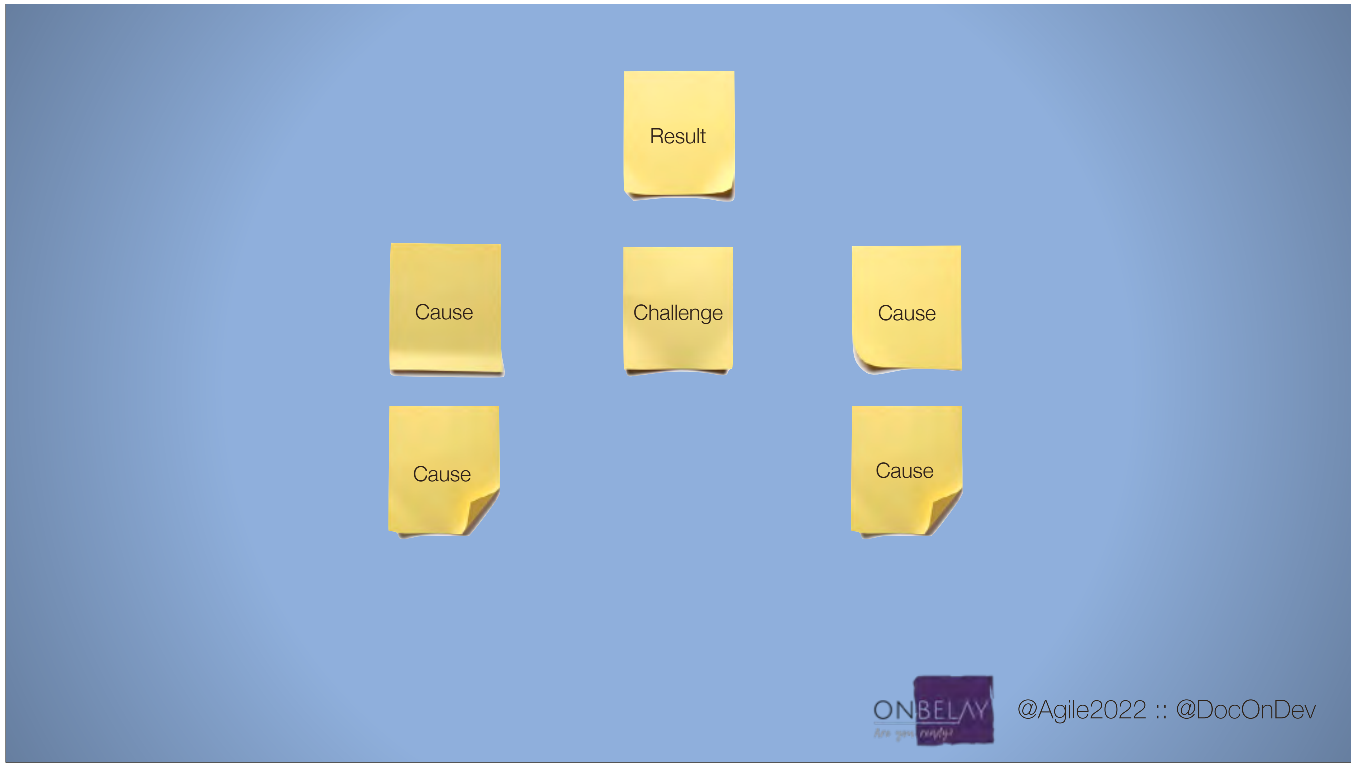


Result

Cause

Challenge

Cause



Result

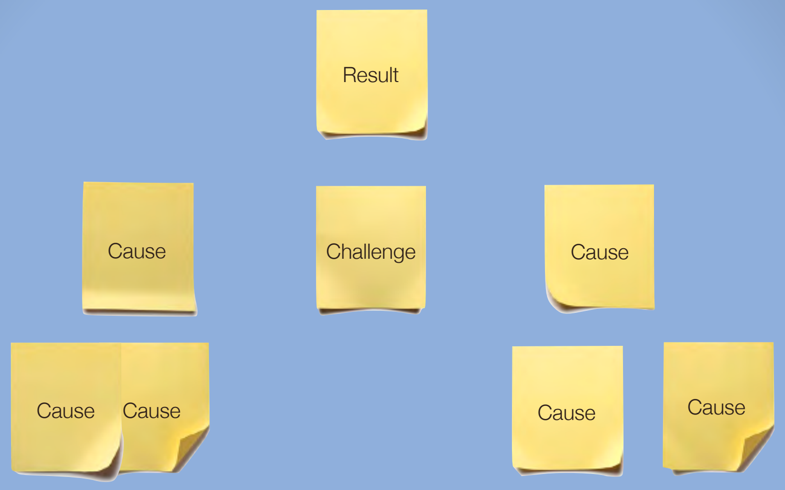
Cause

Challenge

Cause

Cause

Cause



Result

Cause

Challenge

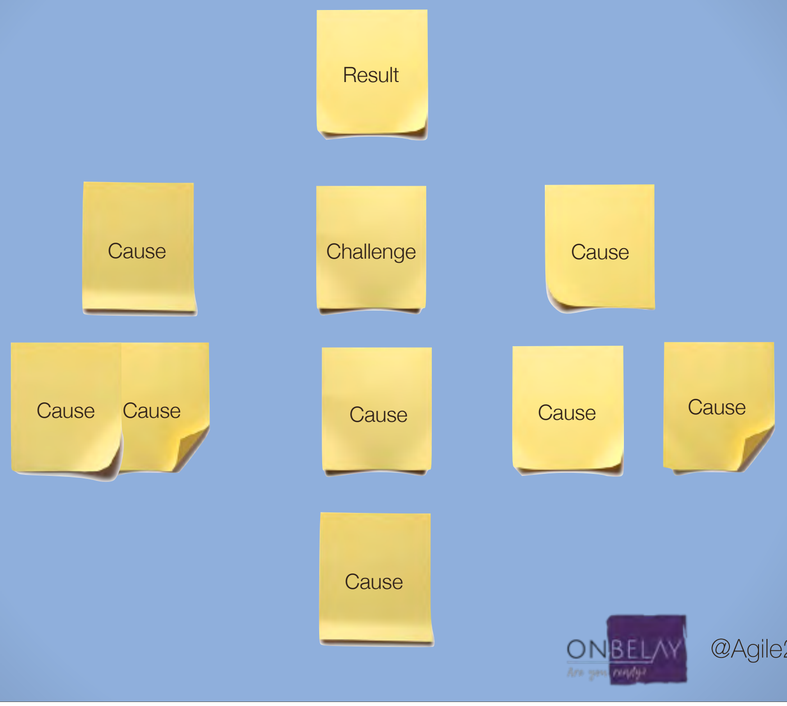
Cause

Cause

Cause

Cause

Cause



Result

Cause

Challenge

Cause

Cause

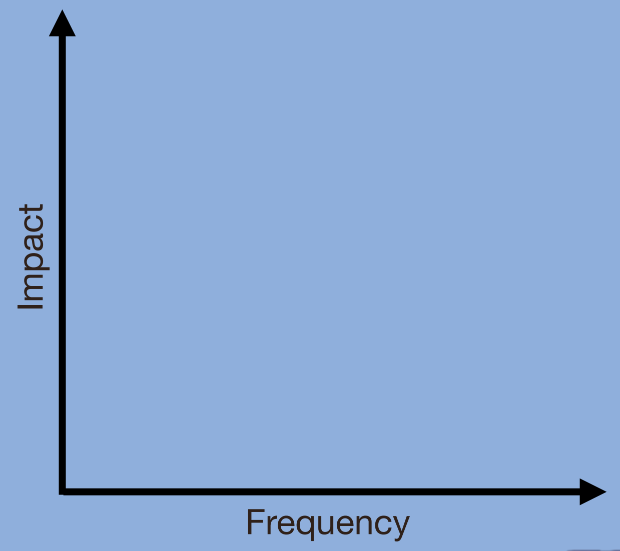
Cause

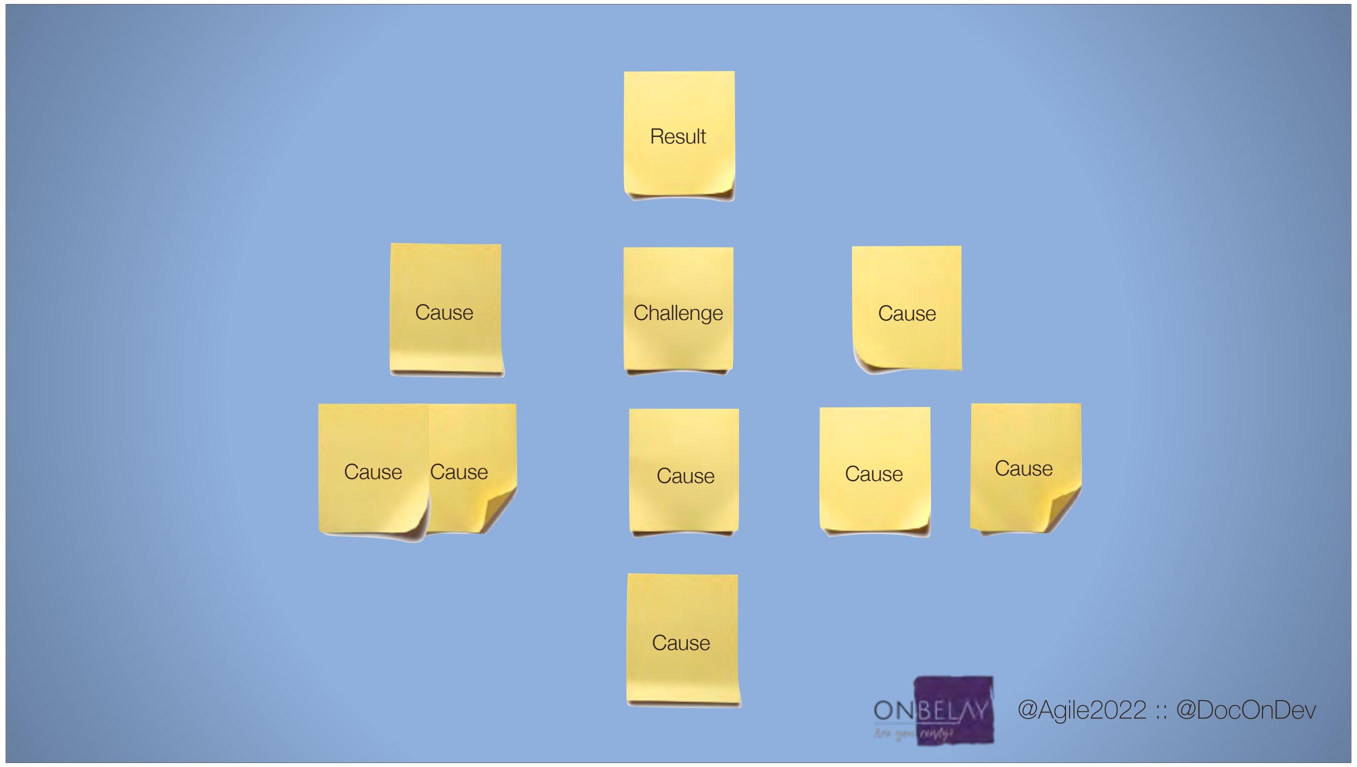
Cause

Cause

Cause

Cause


Impact
Frequency



Result

Cause

Challenge

Cause

Cause

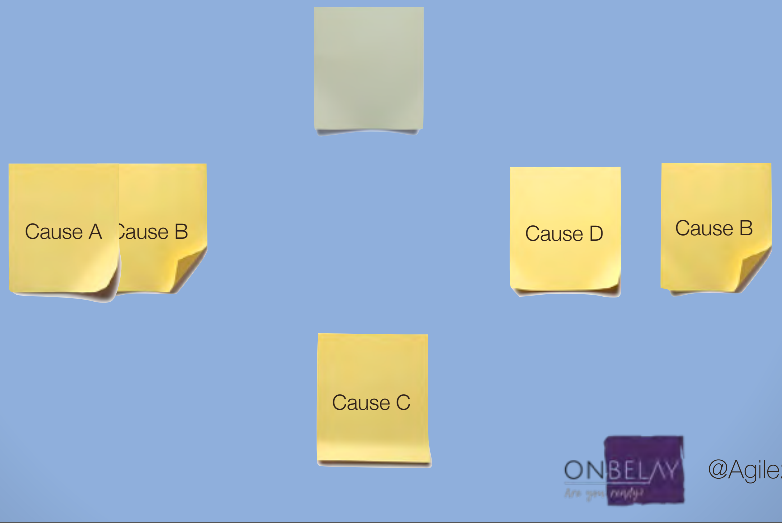
Cause

Cause

Cause

Cause

Cause



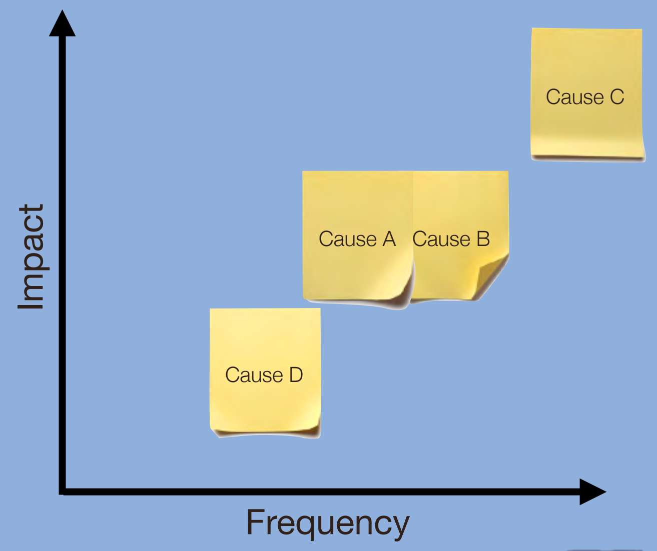
Cause A

Cause B

Cause D

Cause B

Cause C


Impact
Cause C
Cause A
Cause B
Cause D
Frequency

Hawthorne Effect

@Agile2022 :: @DocOnDev

The Hawthorne effect (also referred to as the observer-expectancy bias) is a type of reactivity in which individuals modify an aspect of their behavior in response to their awareness of being observed. This can undermine the integrity of research, particularly the relationships between variables.

The original research at the Hawthorne Works for telephone equipment in Cicero, Illinois, on lighting changes and work structure changes such as working hours and break times was originally interpreted by Elton Mayo and others to mean that paying attention to overall worker needs would improve productivity.

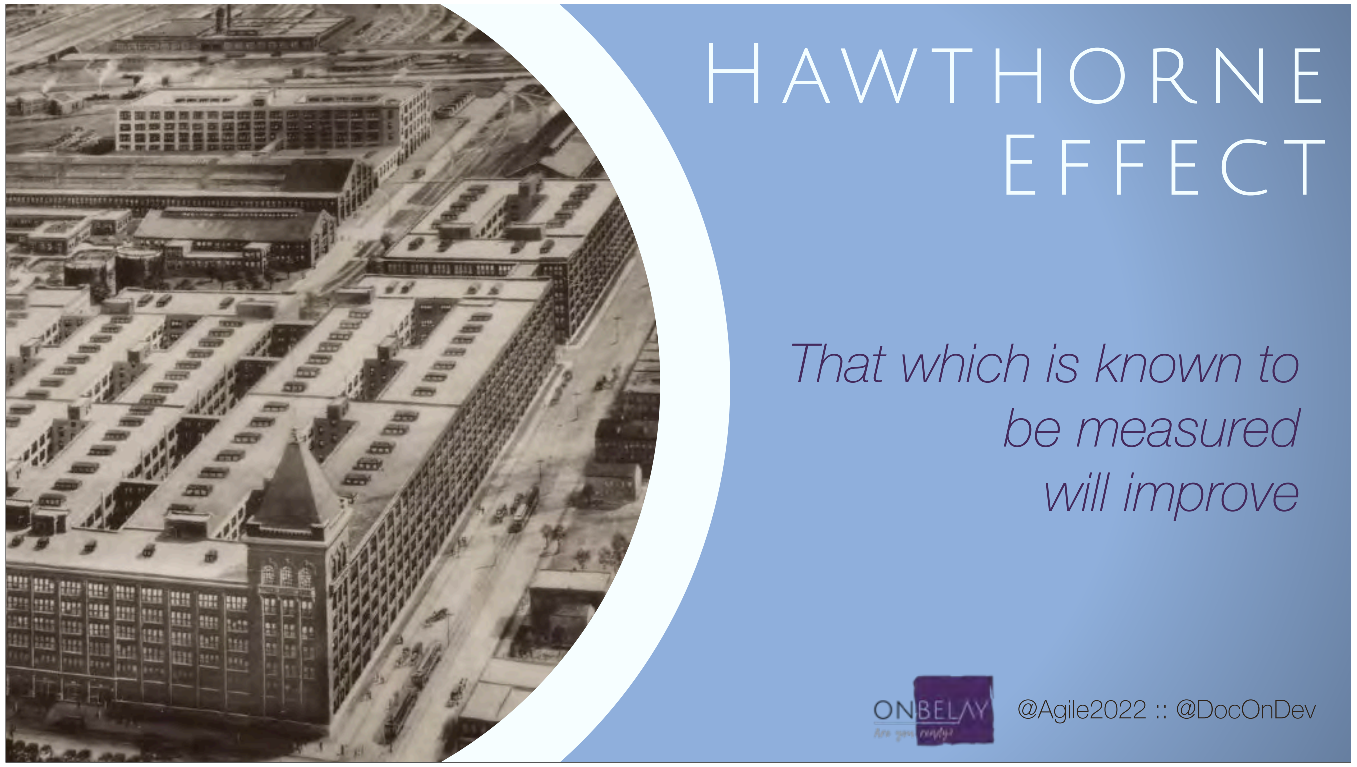
Later interpretations such as that done by Landsberger suggested that the novelty of being research subjects and the increased attention from such could lead to temporary increases in workers' productivity. This interpretation was dubbed "the Hawthorne effect". It is also similar to a phenomenon that is referred to as novelty/disruption effect.[6]

Hawthorne Effect

@Agile2022 :: @DocOnDev

The Hawthorne effect (also referred to as the observer-expectancy bias and closely related to novelty bias) is a type of reactivity in which individuals modify an aspect of their behavior in response to their awareness of being observed. This can undermine the integrity of research, particularly the relationships between variables.

The original research at the Hawthorne Works for telephone equipment in Cicero, Illinois, on lighting changes and work structure changes such as working hours and break times - the increased attention from being measured/monitored lead to temporary increases in workers' productivity.



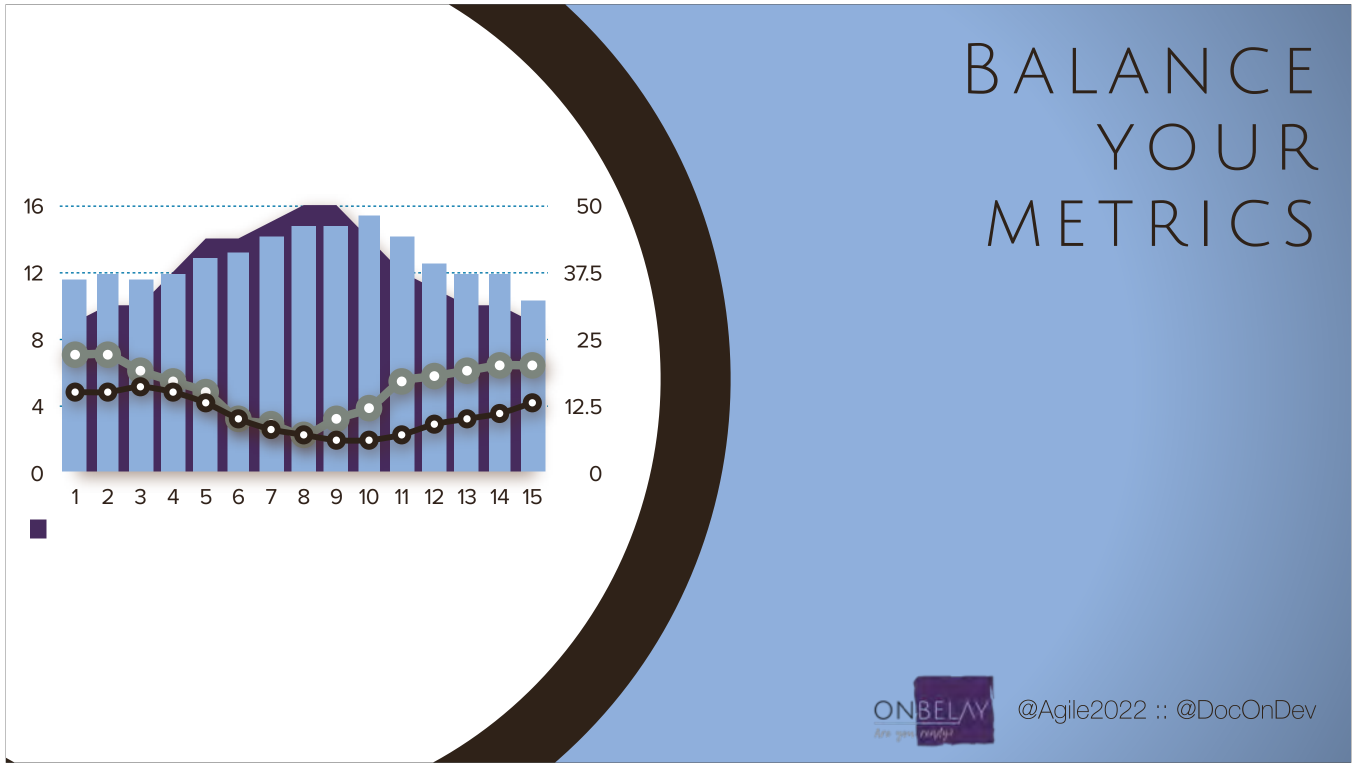
Hawthorne Effect

That which is known to be measured will improve

@Agile2022 :: @DocOnDev

The Hawthorne effect (also referred to as the observer-expectancy bias and closely related to novelty bias) is a type of reactivity in which individuals modify an aspect of their behavior in response to their awareness of being observed. This can undermine the integrity of research, particularly the relationships between variables.

The original research at the Hawthorne Works for telephone equipment in Cicero, Illinois, on lighting changes and work structure changes such as working hours and break times - the increased attention from being measured/monitored lead to temporary increases in workers' productivity.

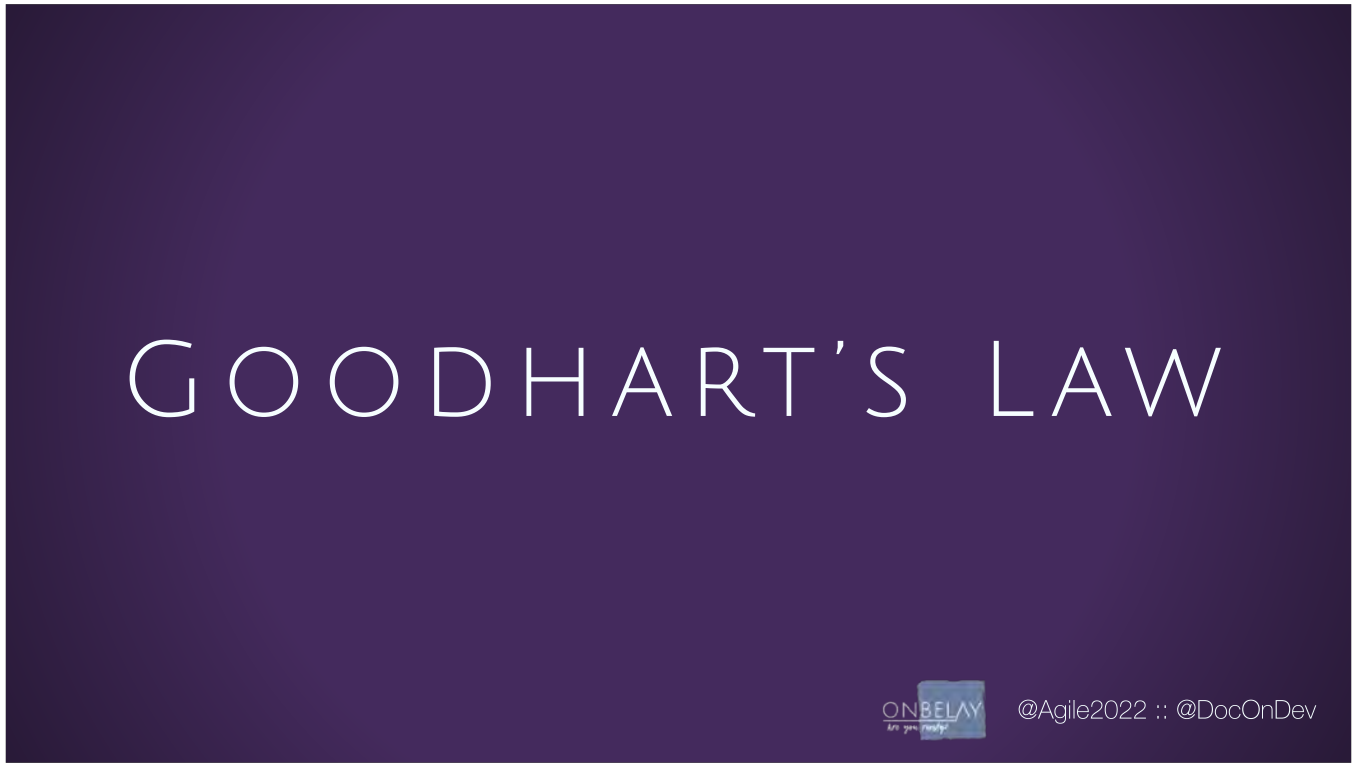


BALANCE YOUR METRICS

ONBELAY

@Agile2022 :: @DocOnDev

If you are going to measure Throughput, consider also measuring Code Quality as a potential countervailing measure



Goodhart's Law

GOODHART'S LAW

Photo By Jamesfranklingresham - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22189516

@Agile2022 :: @DocOnDev

An adage in economics known as Goodhart's Law:

Charles Goodhart was an economic advisor to the UK Government under Margaret Thatcher. Thatcher's approach to monetary policy included setting targets for specific financial indicators.

Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes.

In other words - when you set a target for a metric, the odds are the metric no longer means what it once did and therefore your target doesn't mean what you think it does.

Goodhart's Law

When a measure becomes a target, it ceases to be a good measure.

Photo By Jamesfranklingresham - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=22189516

ONBELAY @Agile2022 :: @DocOnDev

An adage in economics known as Goodhart’s Law:

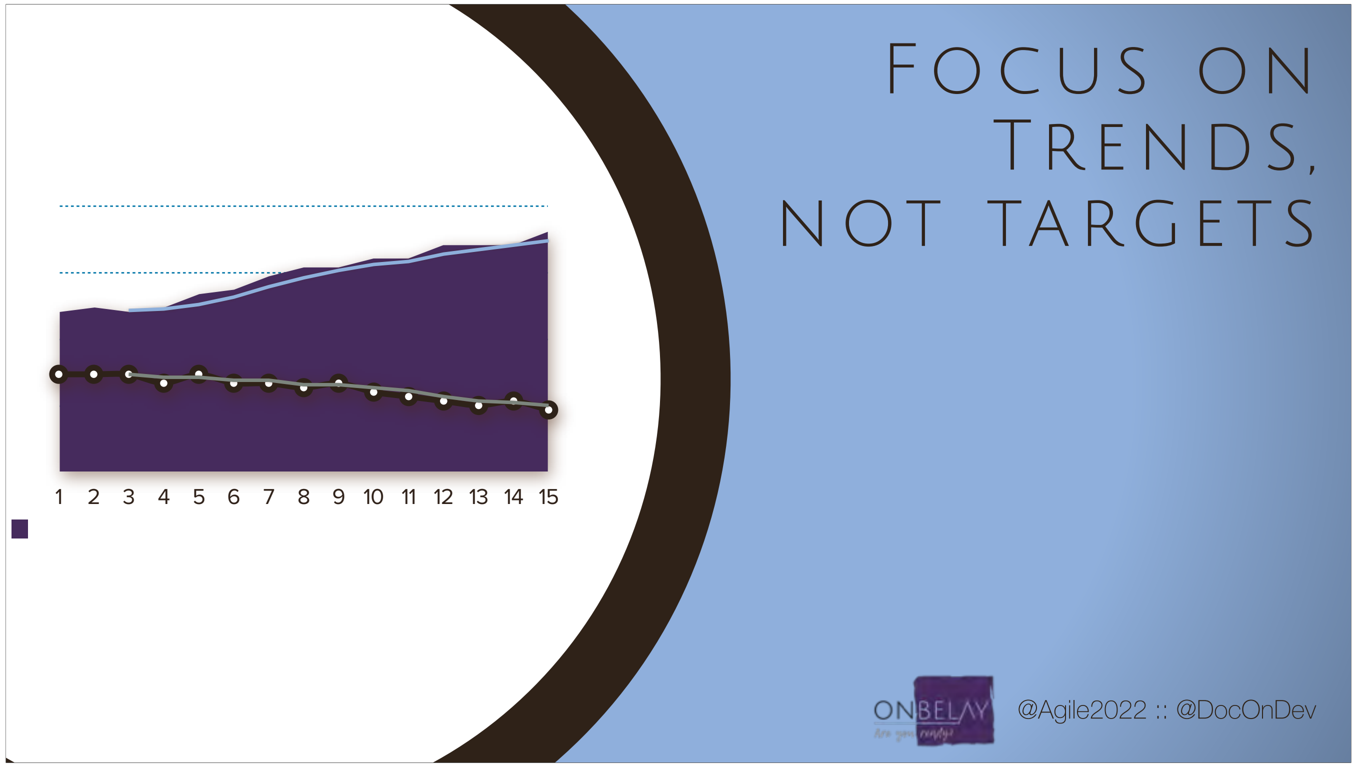
Charles Goodhart was an economic advisor to the UK Government under Margaret Thatcher. Thatcher’s approach to monetary policy included setting targets for specific financial indicators.

Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes.

In other words - when you set a target for a metric, the odds are the metric no longer means what it once did and therefore your target doesn’t mean what you think it does.

PEOPLE WITH TARGETS AND JOBS DEPENDENT UPON MEETING THEM WILL PROBABLY MEET THE TARGETS – EVEN IF THEY HAVE TO DESTROY THE ENTERPRISE TO DO IT.

- W. Edwards Deming

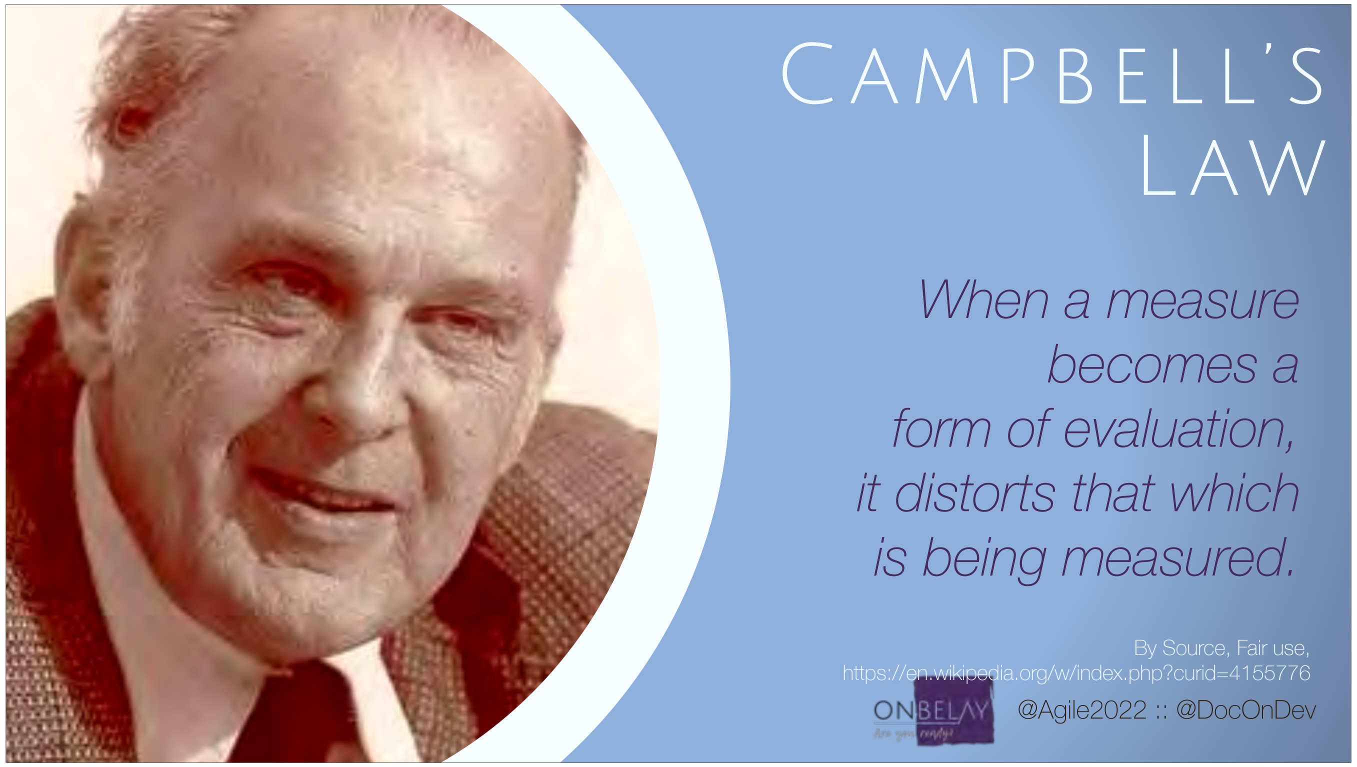


Focus on Trends, Not Targets

@Agile2022 :: @DocOnDev

Pay attention to how the values are trending, not what the values are. Are they trending consistent with the team’s strategy?

Campbell's Law



CAMPBELL'S LAW

When a measure becomes a form of evaluation, it distorts that which is being measured.

By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=4155776

ONBELAY @Agile2022 :: @DocOnDev

Social Sciences - Donald T. Campbell

"The more any quantitative social indicator is used for social decision-making, the more subject it will be to corruption pressures and the more apt it will be to distort and corrupt the social processes it is intended to monitor.”

By Source, Fair use, https://en.wikipedia.org/w/index.php?curid=4155776

Avoid turning metrics into incentives

@Agile2022 :: @DocOnDev

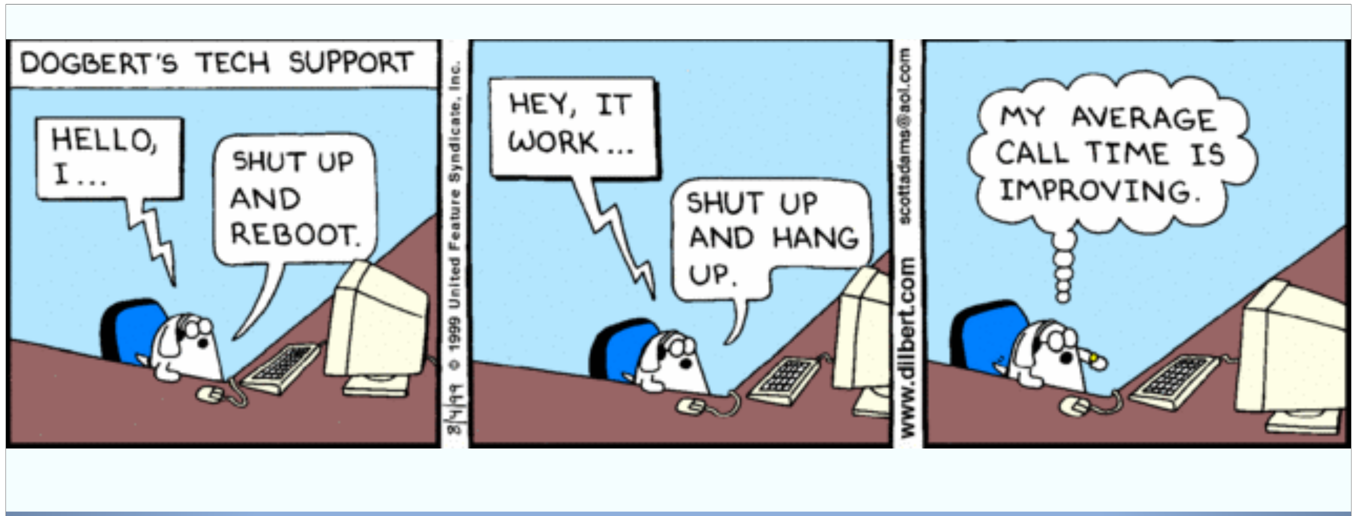
Comparing teams
Rewarding/recognizing teams that do “better” on their metrics
talking about measures more than the desired outcomes

Perverse Incentives

Perverse Incentives

An unintended result, contrary to the interests of the incentive makers

ONBELAY @Agile2022 :: @DocOnDev


DOGBERT'S TECH SUPPORT
HELLO, I...
SHUT UP AND REBOOT.
8/4/99 © 1999 United Feature Syndicate, Inc.
HEY, IT WORK...
SHUT UP AND HANG UP.
www.dilbert.com scottadams@aol.com
MY AVERAGE CALL TIME IS IMPROVING.

Perverse Incentives

An unintended result, contrary to the interests of the incentive makers

ONBELAY @Agile2022 :: @DocOnDev

I think it is interesting how often you'll hear a manager say, the employees gamed the system.

Let's get this straight right now.

How Do You Forecast?

@Agile2022 :: @DocOnDev

Ways they forecast now.

Ways they know that you’re going to make it to the deadline given some quantity of work and a target date.

FORECASTING

@Agile2022 :: @DocOnDev

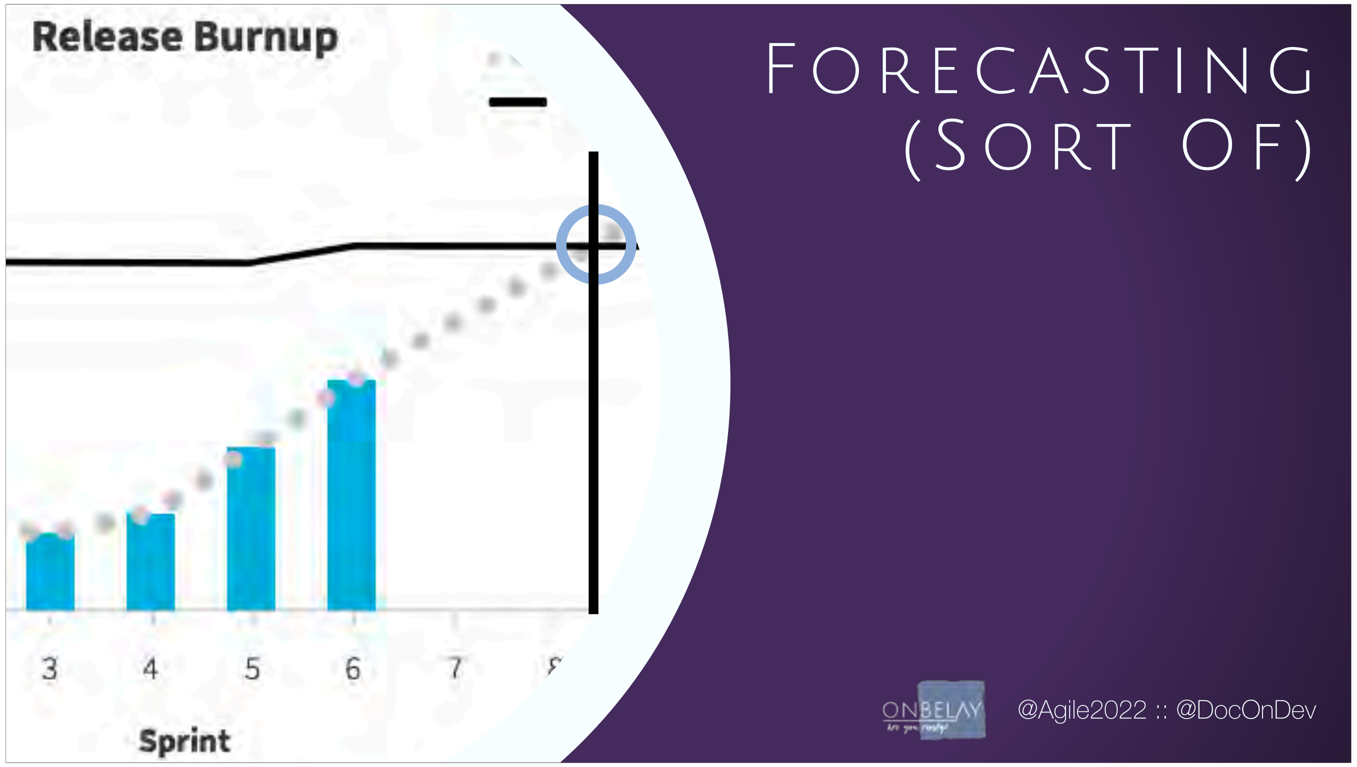
Velocity typically has three related uses:
A gauge when planning our iterations (sometimes referred to as making a commitment)
Determine if we are tracking to our iteration goals or release goals

Means of forecasting. Sort of.

Among these, forecasting is probably the most important and the hardest to do.

Forecasting (Sort of)

We sort of forecast



FORECASTING (SORT OF)

@Agile2022 :: @DocOnDev

We take our current velocity or the average velocity of the past few iterations and we divide it into our best current guess for the work that needs to be done.

Maybe we graph our best current guess for the work that need to be done and graph our burn with an extended trend line. Where the work line and trend line cross <next> is when we'll be done.

ARE THESE FORECASTS ACCURATE?

@Agile2022 :: @DocOnDev

No. They do not tend to be particularly accurate. Even if we might imagine they are precise, they are not accurate.

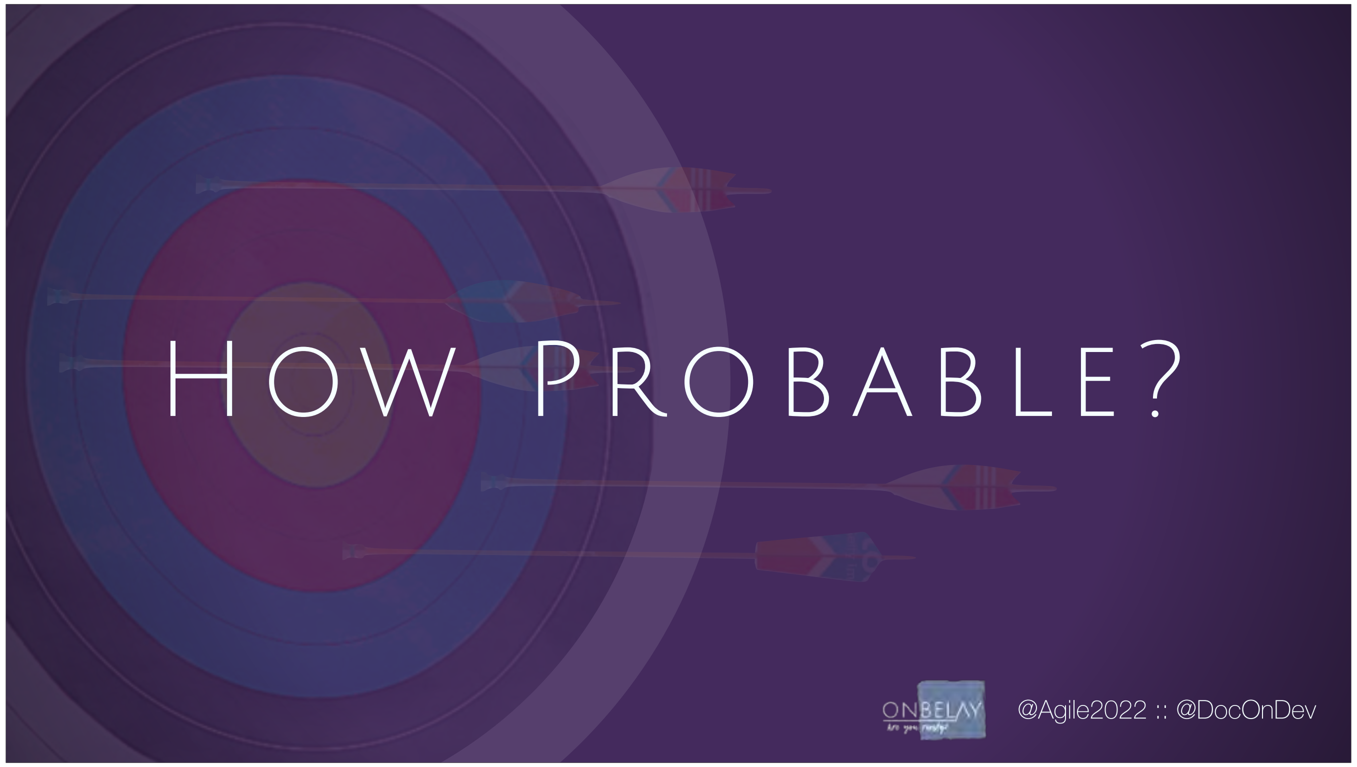
ARE THESE FORECASTS DEFINITE?

No. They are not definite.

Are these forecasts probable?

@Agile2022 :: @DocOnDev

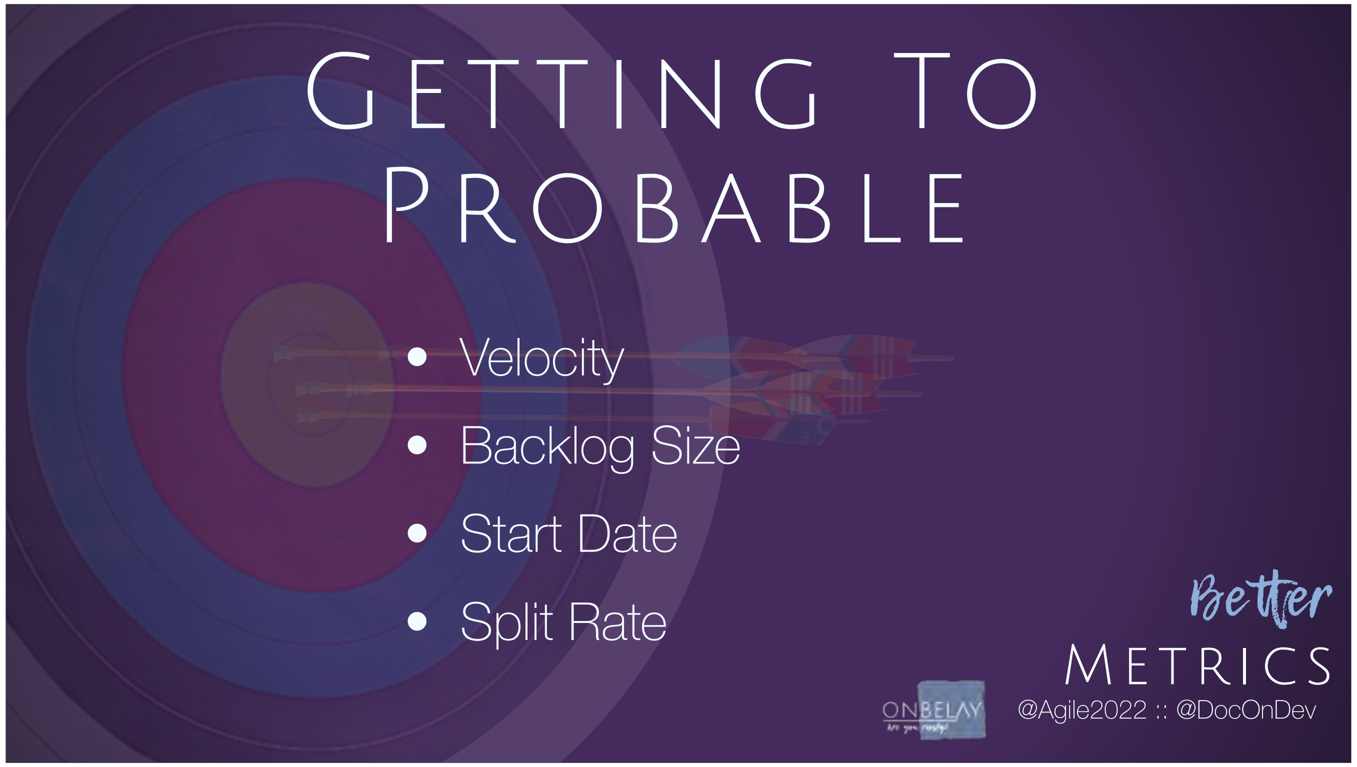
Maybe. They’re possible, for sure. But how probably are they? How much confidence do you have in your forecasts when using the common technique?



How Probable?

@Agile2022 :: @DocOnDev

The truth is, we don’t know this. We do not know the mathematical probability of hitting our dates.



Getting To Probable

- Velocity
- Backlog Size
- Start Date
- Split Rate

Better Metrics

ONBELAY

@Agile2022 :: @DocOnDev

But there is a way to get better at this. <next>

You’ll need to know your velocity and backlog size - so far this sounds familiar… <next>
You’ll also need your start date and split rate.

The start date is usually now or in the future. We are forecasting work remaining, not work already done.

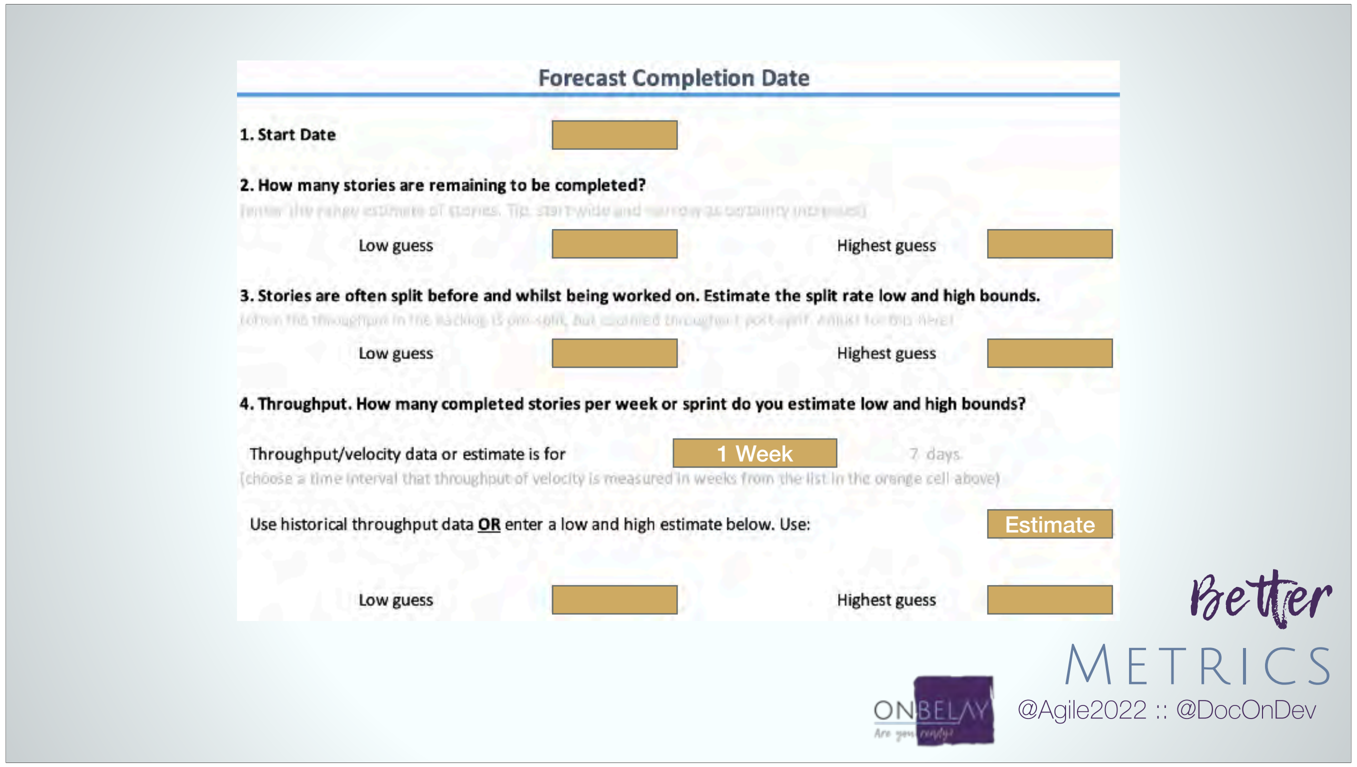
The split rate is the percentage of growth - say one story in the backlog ends up being two or more stories when you execute on it. This happens with progressive elaboration.

FOCUSED OBJECTIVE

Focused Objective
forecasting - risk - staff - cost of delay

http://focusedobjective.com/
free-tools-resources/

Go get the Forecasting tool from Focused Objective. Here’s the URL.



Forecast Completion Date

1. Start Date

2. How many stories are remaining to be completed?

(enter the range estimate of stories. Tip: start wide and narrow as certainty increases)

Low guess Highest guess

3. Stories are often split before and whilst being worked on. Estimate the split rate low and high bounds.

(when the throughput in the backlog is pre-split, but counted throughput post-split. Adjust for this here)

Low guess Highest guess

4. Throughput. How many completed stories per week or sprint do you estimate low and high bounds?

Throughput/velocity data or estimate is for 1 Week 7 days

(choose a time interval that throughput of velocity is measured in weeks from the list in the orange cell above)

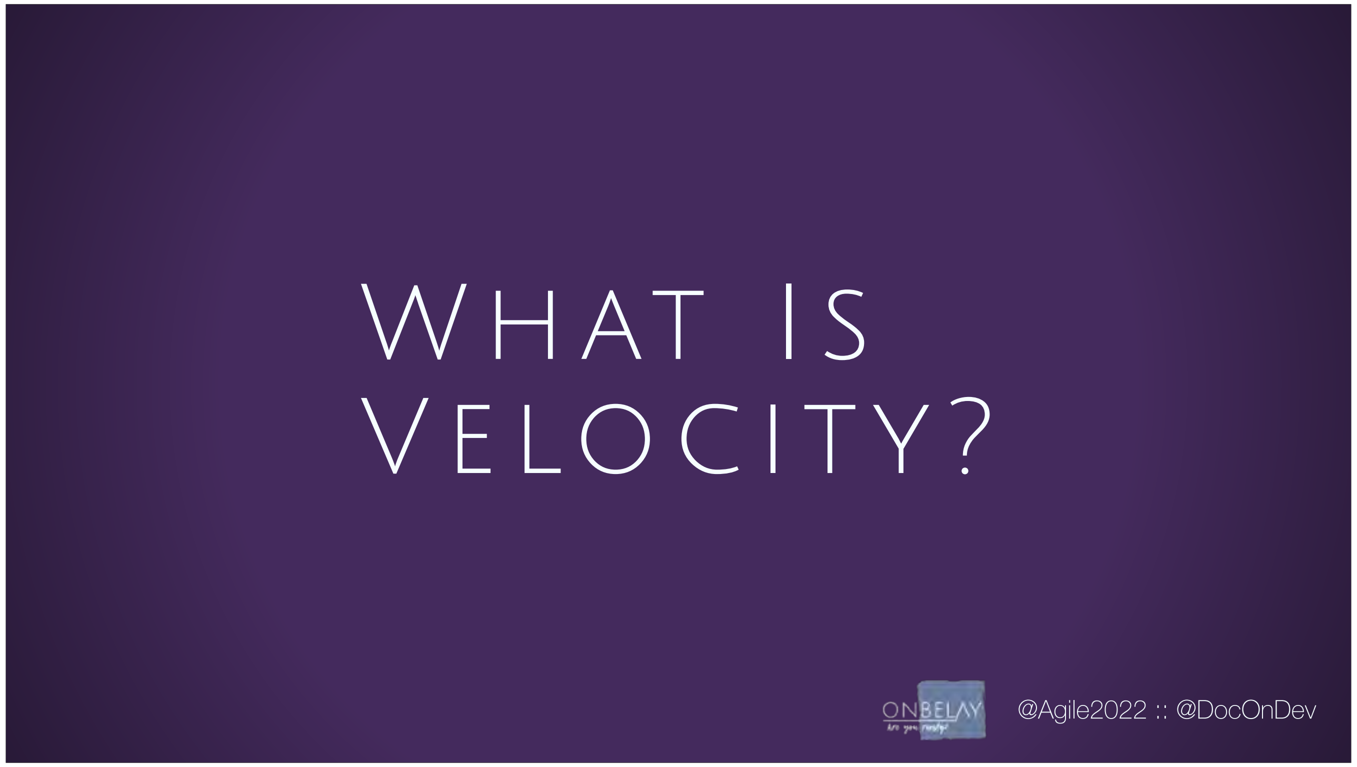
Use historical throughput data **OR** enter a low and high estimate below. Use: Estimate

Low guess Highest guess

You enter the start date
Your low and high guess for stories remaining - this are often the same number, but not always
Your split rate - if you have not been tracking split rate, you can look at the percentage of growth of your overall backlog over the past few iterations.
Your velocity increment - this is used to bucket into iterations
And your velocity or throughput history. I use the high and low from the past few iterations.

Be honest with the numbers. You can use this tool to create a forecast that looks how you want OR you can use this tool to create a realistic forecast based on probability. The latter is smart. The former is a waste of this tool.

EXIT TO EXCEL TO SHOW THIS LIVE



What Is Velocity?

@Agile2022 :: @DocOnDev

To figure out WHY this is, let’s start with WHAT it is…

What Is Velocity?

#AgileMetrics :: @DocOnDev

What Is Velocity?

ONBELAY @Agile2022 :: @DocOnDev

What Is Velocity?

Work Units Over Time

@Agile2022 :: @DocOnDev

Work Units over Time - technically, that would be speed. Velocity requires a vector.
Work Units over Time toward delivery of value - Velocity...?

What Is Velocity?

Lagging Indicator

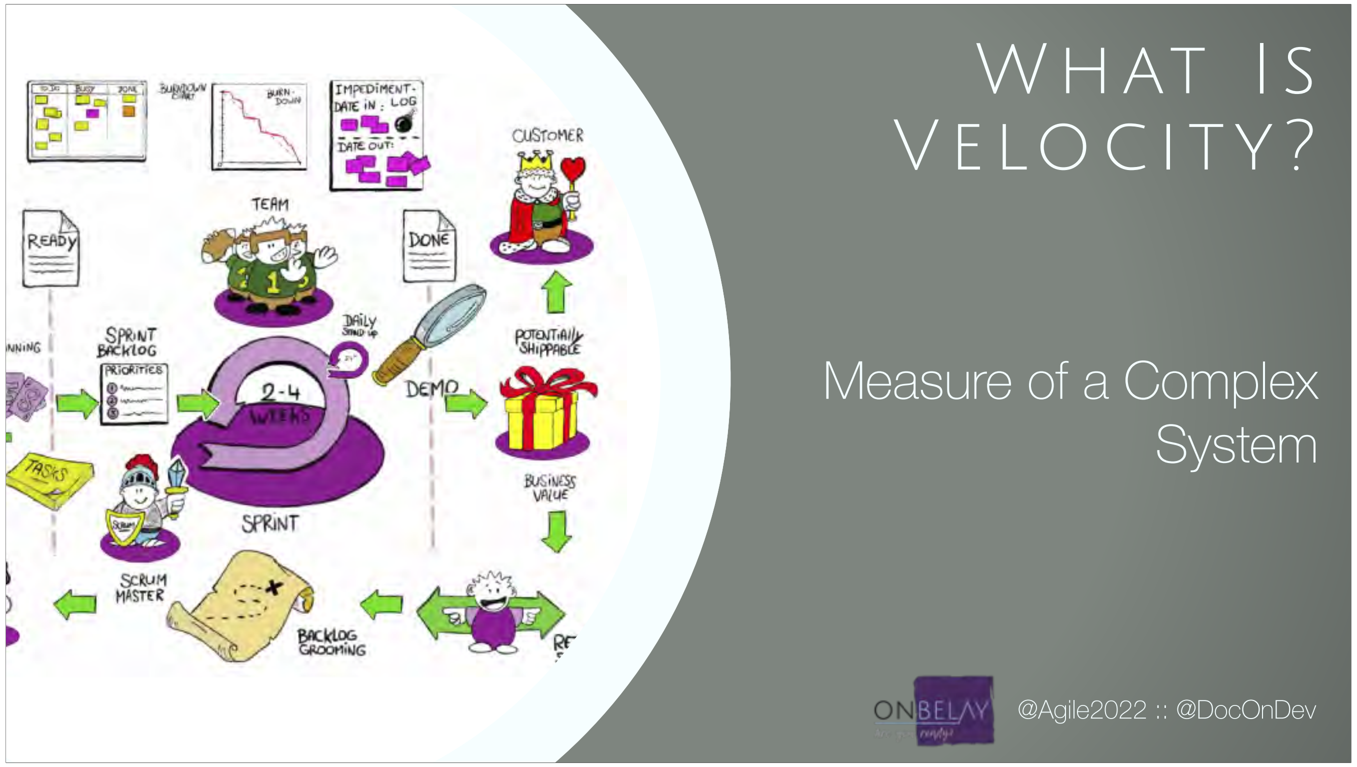
@Agile2022 :: @DocOnDev

Lagging Indicator
Tells us what happened.
Lagging Indicators confirm long-term trends, but are not good predictors.

I overheard a discussion the other day where someone said, “meteorologists/climatologists cannot be trusted. They claim they know climate trends, but they can’t even tell you what the weather will be tomorrow.”

If the unemployment rate is rising, the economy has been doing poorly.
We know how it is trending, but we don’t know specifically where it will go next.

Lagging indicators are good for trends over broad cycles. You know that sales will be higher around Christmas every year. But you don’t know what sales will be.

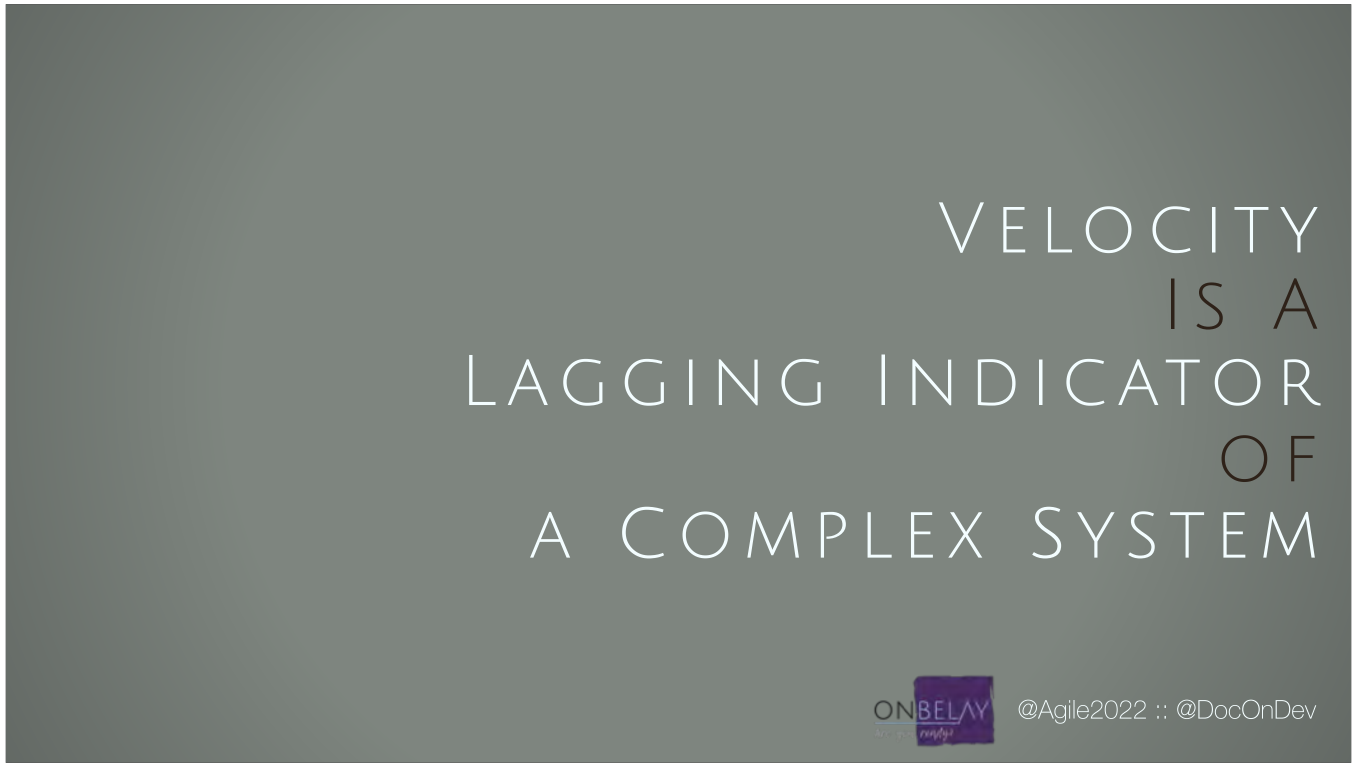


What Is Velocity?

Measure of a Complex System

@Agile2022 :: @DocOnDev

Tells us about the end result (sort of), but nothing about the process by which that result was attained

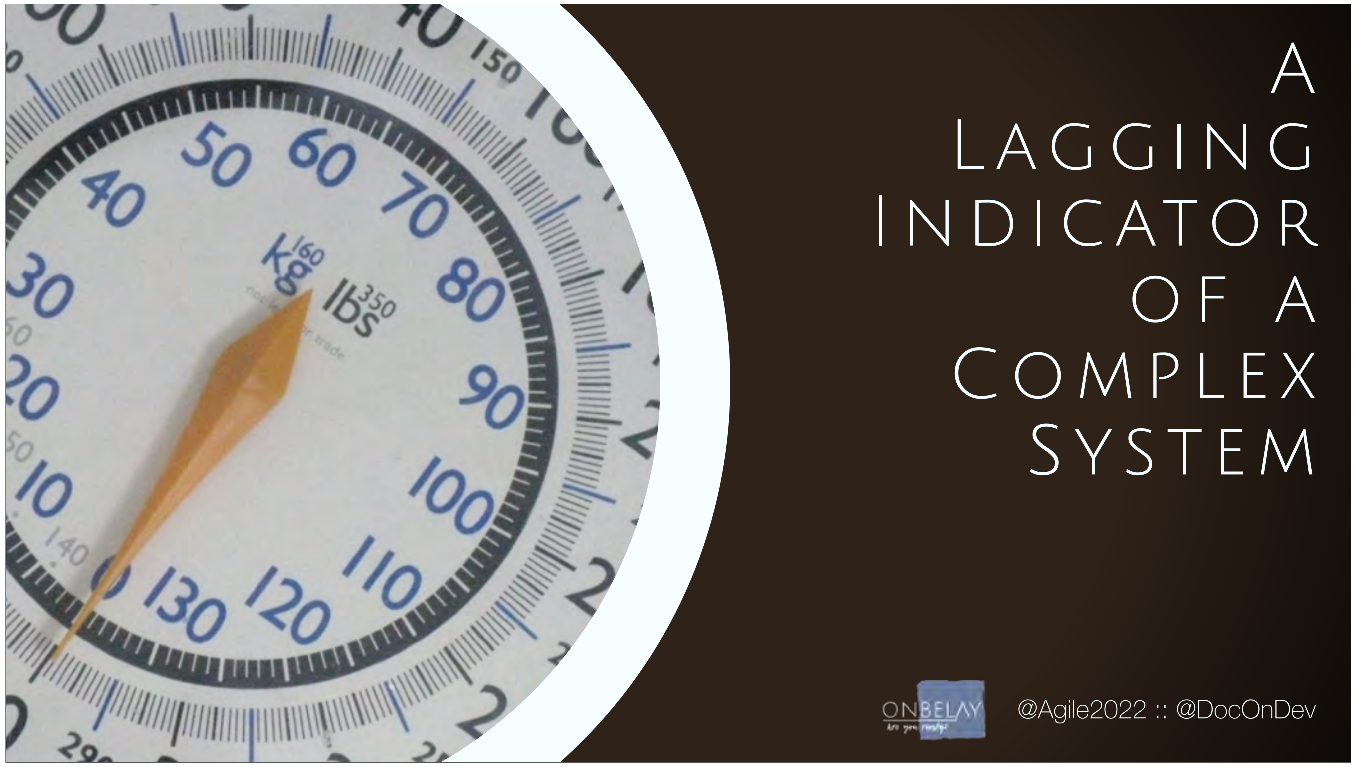


Velocity
Is a
Lagging Indicator
of
a Complex System

ONBELAY @Agile2022 :: @DocOnDev

That’s interesting, but it doesn’t help me reason about it much.

So I got to thinking about it - what’s something that is a lagging indicator for a complex system? Something that might help me think about this more clearly…



A Lagging Indicator of a Complex System

@Agile2022 :: @DocOnDev

Your body-weight is a lagging indicator.
What are things that affect your body weight?
Diet, Exercise, Genetics, Physical Health, Mental Health, Environment, Work, Stress, Social Network
Does any given body weight mean you are healthy?
Does any given velocity mean your project is healthy?

WELL, WELL - IT SEEMS YOUR WEIGHT IS PERFECT. YOU JUST HAPPEN TO BE ELEVEN FEET TOO SHORT.

HOW MIGHT SOMEONE LOSE WEIGHT?

ONBELAY @Agile2022 :: @DocOnDev

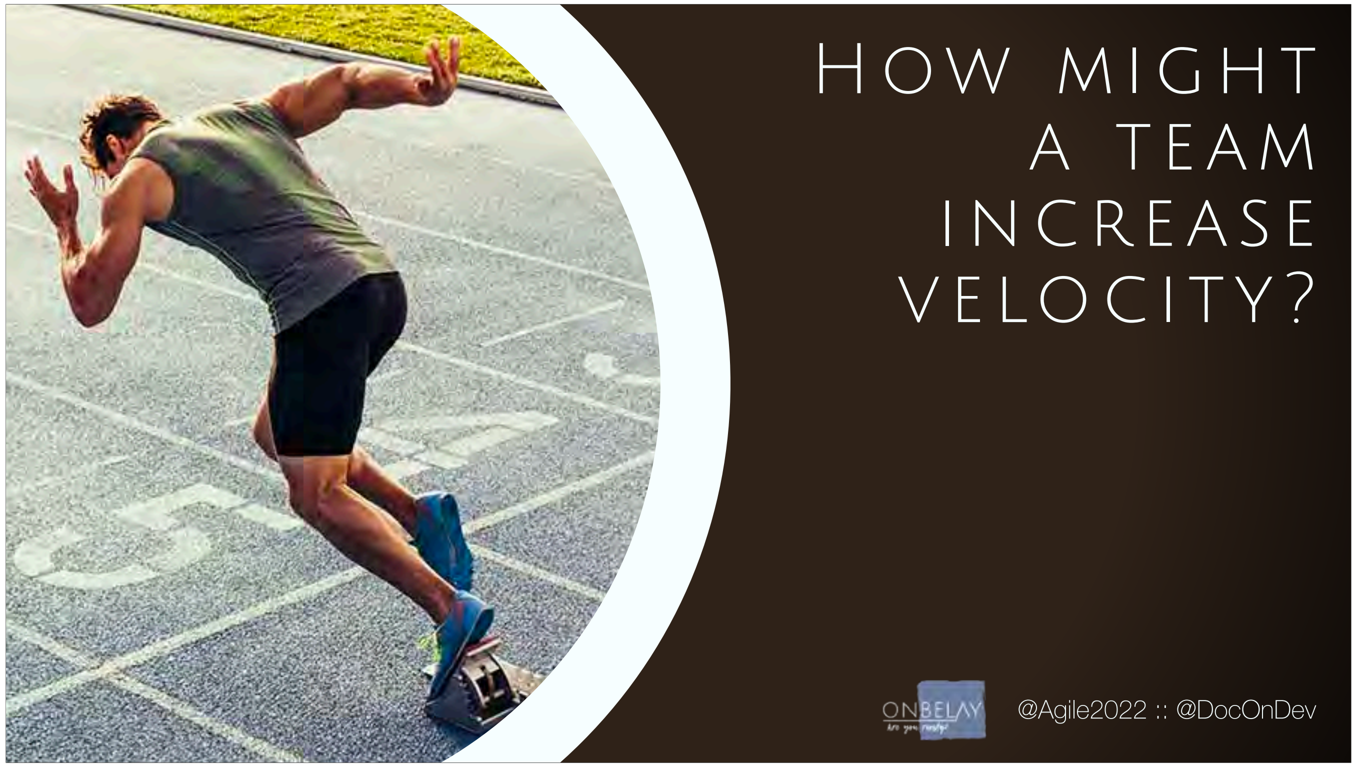
Let’s say you want to lose
  10 pounds
  5 kilograms

Reduce calories, change in exercise, more sleep, cut back on certain foods

Consume nothing but amphetamines
Stop drinking all liquids
Smoke crack
Cut off your forearm

The point here is simple:

Moving the metric in the right direction doesn’t necessarily improve the health of the overall system and MAY even hurt it.



HOW MIGHT A TEAM INCREASE VELOCITY?

@Agile2022 :: @DocOnDev

Limit WIP, reduce batch size, reduce dependencies

Skip testing
Cut corners on internal quality
Work more hours per week
Increase all estimates

The point here is simple:

Moving the metric in the right direction doesn’t necessarily improve the health of the overall system and MAY even hurt it.

YEAH…
SO I'M
GONNA NEED
YOU TO GIVE
ME MORE
VELOCITIES,
OKAY?

@Agile2022 :: @DocOnDev

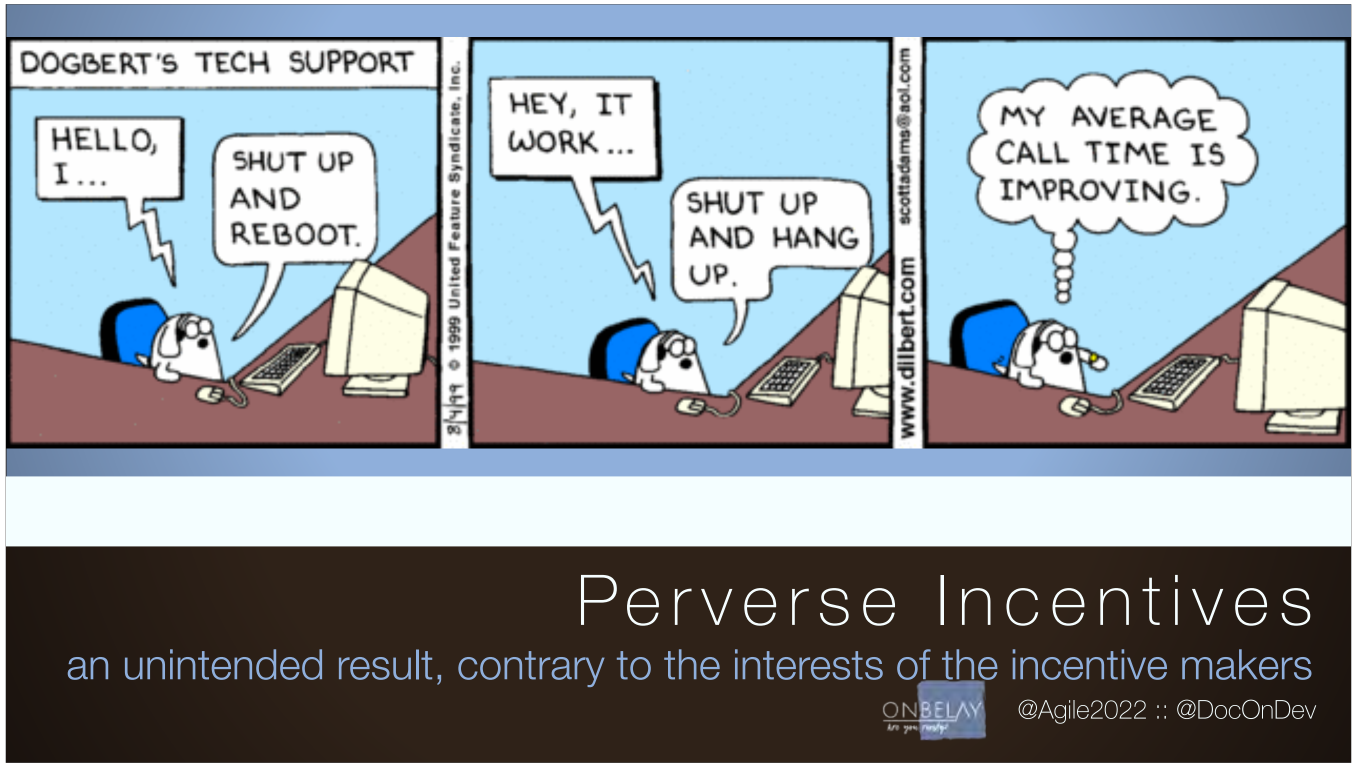
With good intentions or not, when bosses set goals for or ask for improvements in indicators like this, they are almost guaranteed to create problems rather than benefits.

First off, there is a law in economics known as Goodhart's Law which states:

Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes.

In other words - when you set a target for a metric, the odds are the metric no longer means what it once did and therefore your target doesn't mean what you think it does.

Furthermore, properly incentivized, people will hit the target by whatever means necessary.


DOGBERT'S TECH SUPPORT
HELLO, I...
SHUT UP AND REBOOT.
8/4/99 © 1999 United Feature Syndicate, Inc.
HEY, IT WORK...
SHUT UP AND HANG UP.
www.dilbert.com scottadams@aol.com
MY AVERAGE CALL TIME IS IMPROVING.

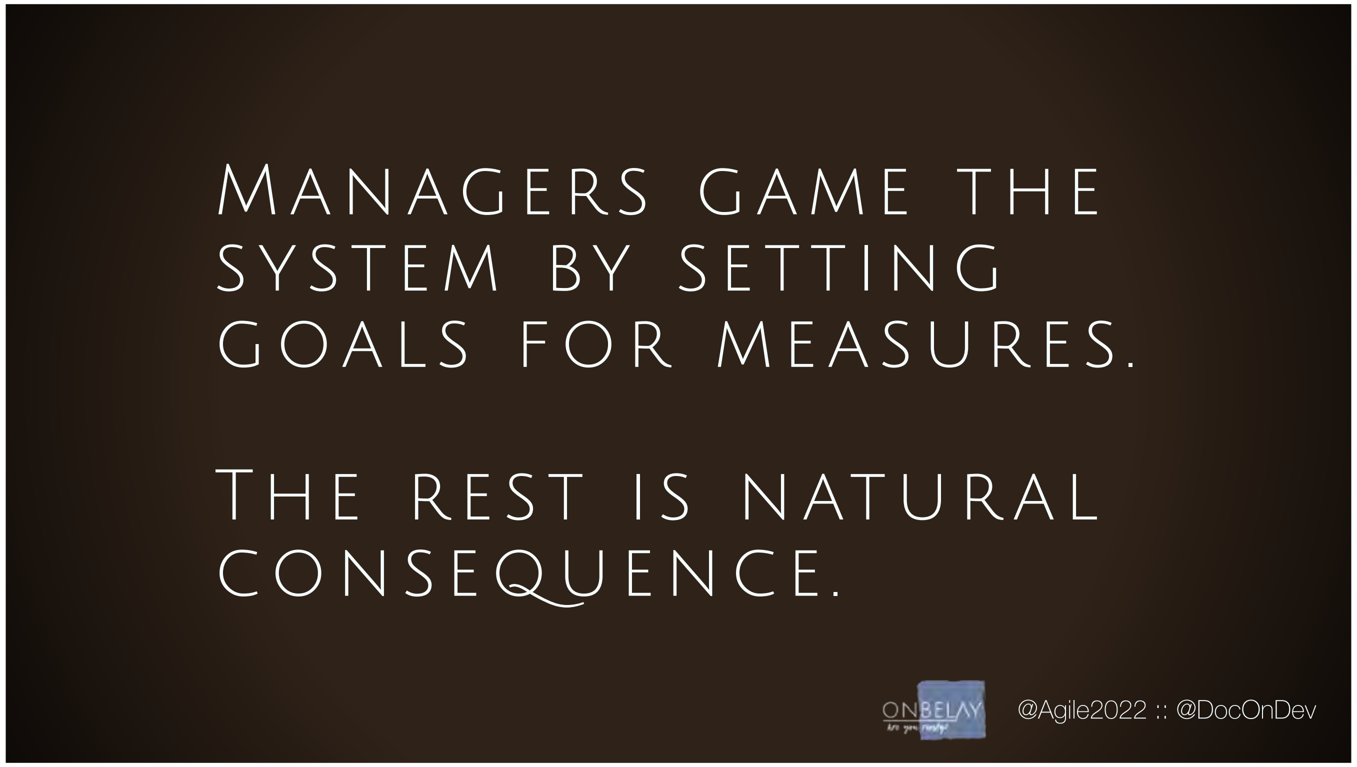
Perverse Incentives

an unintended result, contrary to the interests of the incentive makers

ONBELAY @Agile2022 :: @DocOnDev

I think it is interesting how often you'll hear a manager say, the employees gamed the system.

Let's get this straight right now.

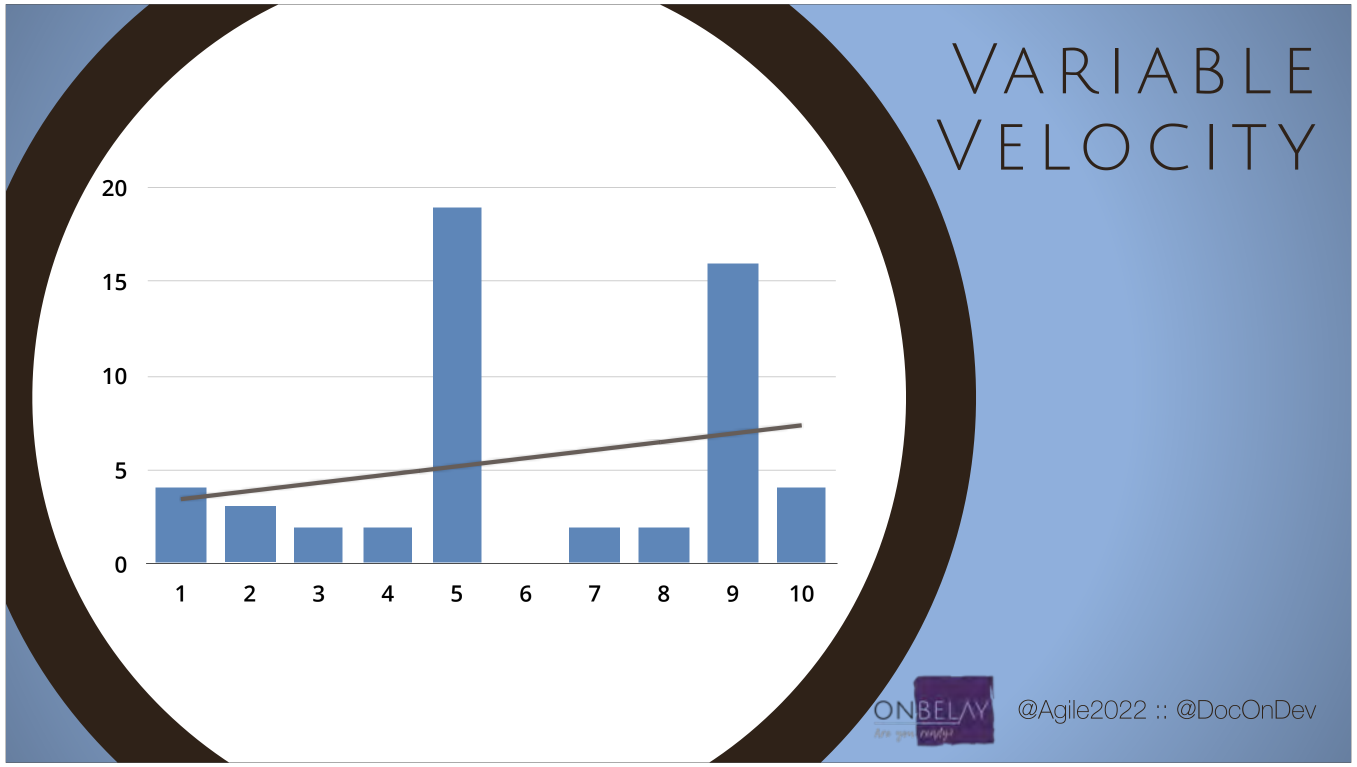


Managers game the system by setting goals for measures.

The rest is natural consequence.

@Agile2022 :: @DocOnDev

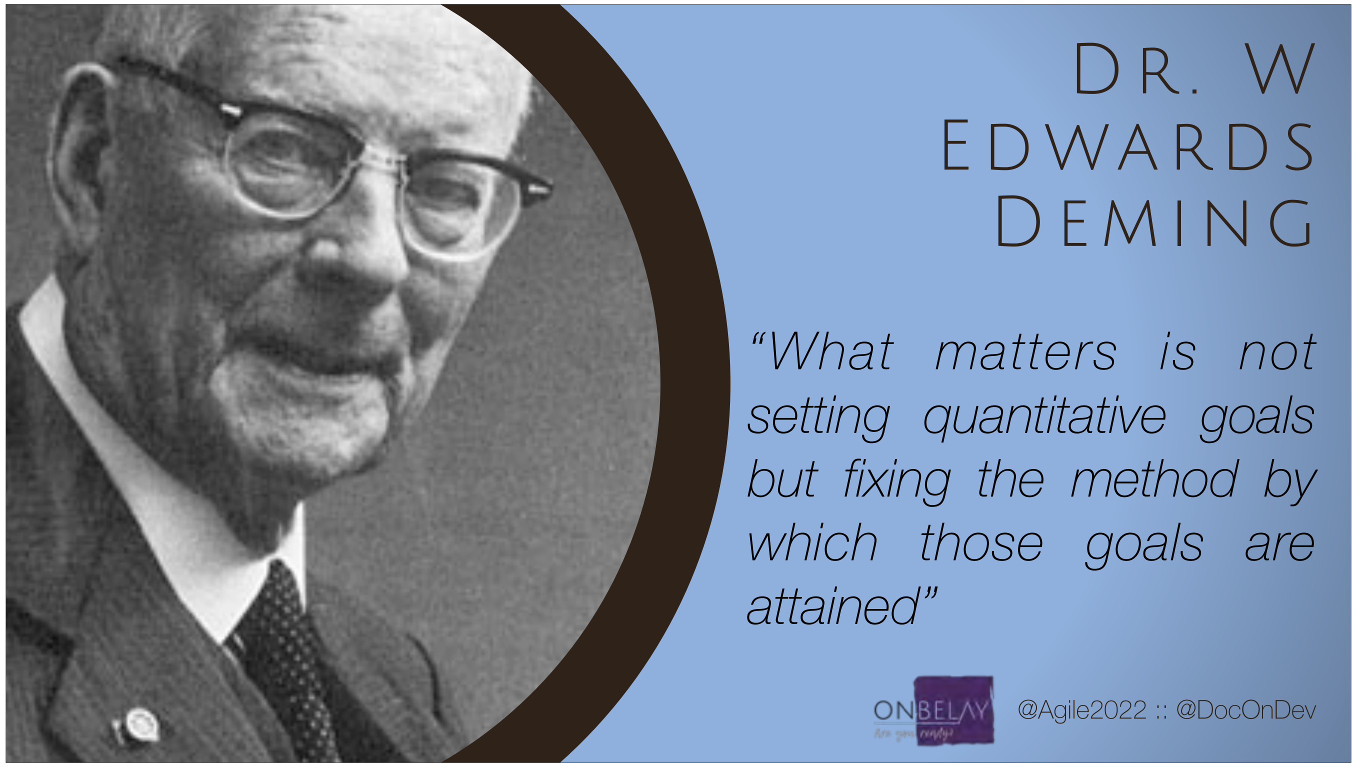
Rewards for # of bugs found - lower importance, duplicate bugs reported
Reward for code coverage - increased coverage/decreased test quality
Reward for higher velocity - more brittle code, lower test coverage, more bugs



Variable Velocity

@Agile2022 :: @DocOnDev

This is a velocity chart for a team over the course of 10 iterations



Dr. W Edwards Deming

“What matters is not setting quantitative goals but fixing the method by which those goals are attained”

@Agile2022 :: @DocOnDev

Deming ~ “Setting quantitative goals doesn’t matter. Fix the methods underlying those goals.”

So let’s look at some common causes of variable velocity.

Variable Velocity

Deferred Maintenance

@Agile2022 :: @DocOnDev

No time to clean the code
No time to refactor
No time to upgrade the infrastructure

These all gum up the system, causing it to move slower and slower in the name of speed

Deferred Maintenance Gums up the system and causes it to move slower.

@Agile2022 :: @DocOnDev

No time to clean the code
No time to refactor
No time to upgrade the infrastructure

These all gum up the system, causing it to move slower and slower in the name of speed

Variable Velocity

Batch Size

This is variable story sizes or consistently large stories
This is stories that don't get delivered until the entire feature is ready
This is features that don't get delivered until the release is ready

Inconsistent batch sizes cause variability in delivery, which results in lower predictability and may result in bottlenecks.

The bigger the Batch Size, the greater the risk.

@Agile2022 :: @DocOnDev

This is variable story sizes or consistently large stories
This is stories that don't get delivered until the entire feature is ready
This is features that don't get delivered until the release is ready

The bigger the batch, the greater the risk. The greater the risk, the more planning required. The more planning required, the more time it takes. The more time it takes, the bigger the batch.

This is a vicious cycle

Variable Velocity

Work In Progress

@Agile2022 :: @DocOnDev

The more items in flight at any one time, the longer each individual item takes to get to completion.
This creates the illusion of progress through business, but impedes progress in terms of actual value delivered sooner.

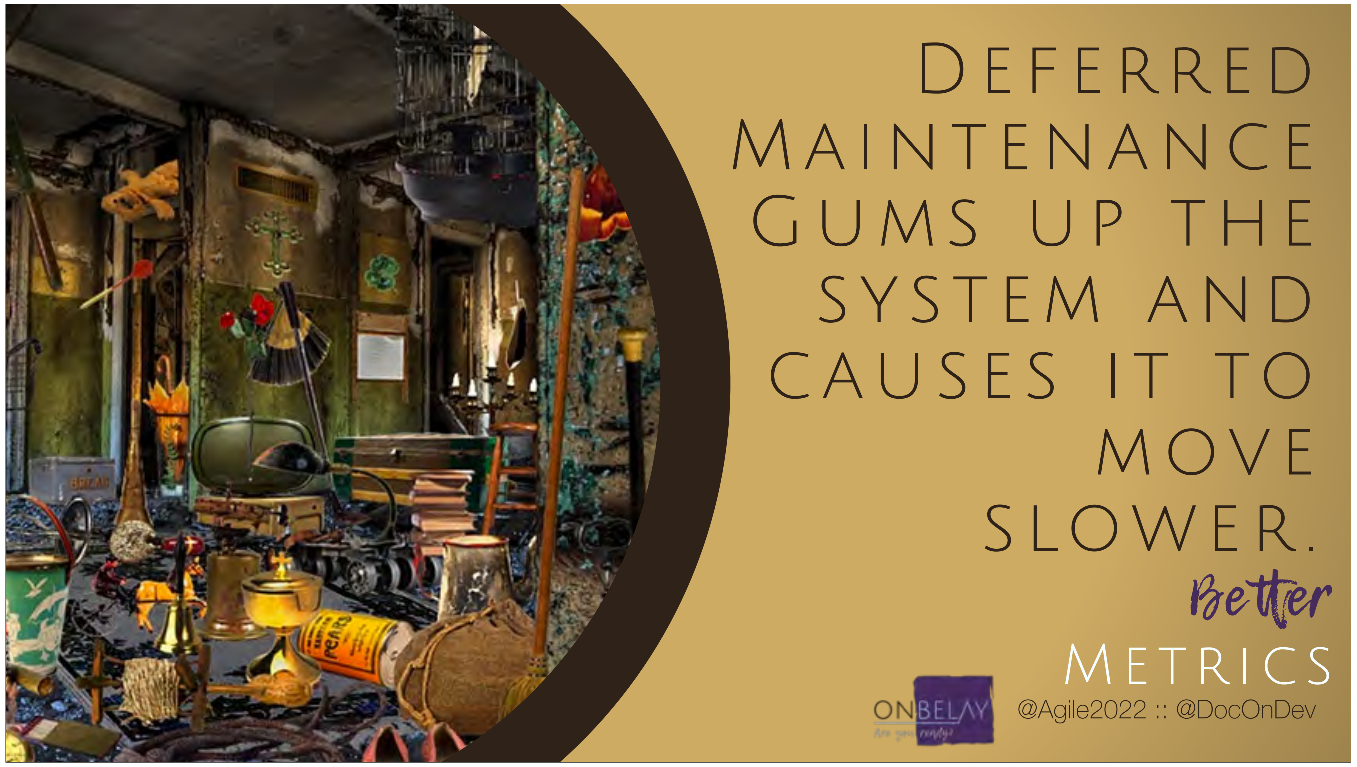
Limit Work In Progress

Stop Starting.
Start Finishing.

@Agile2022 :: @DocOnDev

Don’t start more work. Focus on finishing the work you’ve already started.

Better
METRICS



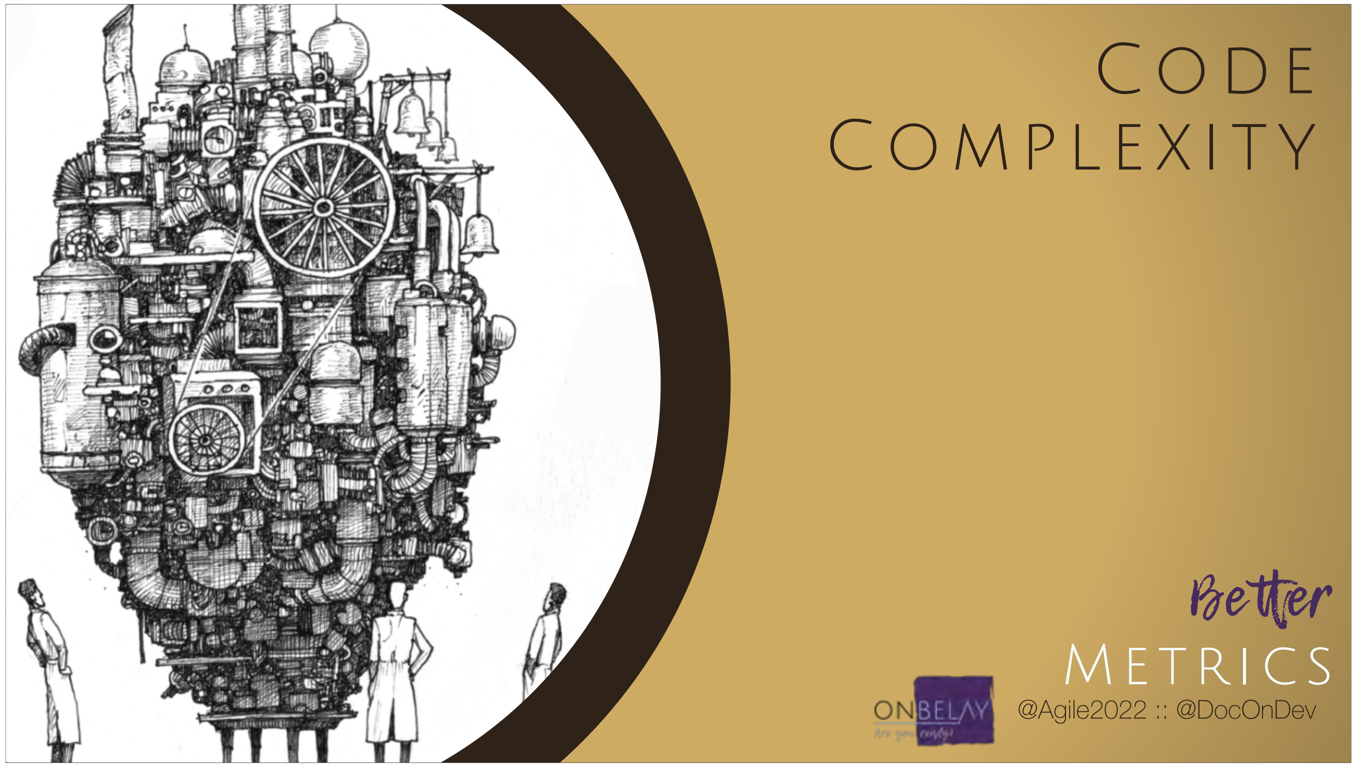
Deferred Maintenance Gums up the system and causes it to move slower.

Better Metrics

@Agile2022 :: @DocOnDev

No time to clean the code
No time to refactor
No time to upgrade the infrastructure

These all gum up the system, causing it to move slower and slower in the name of speed



Code Complexity

Better

Metrics

ONBELAY

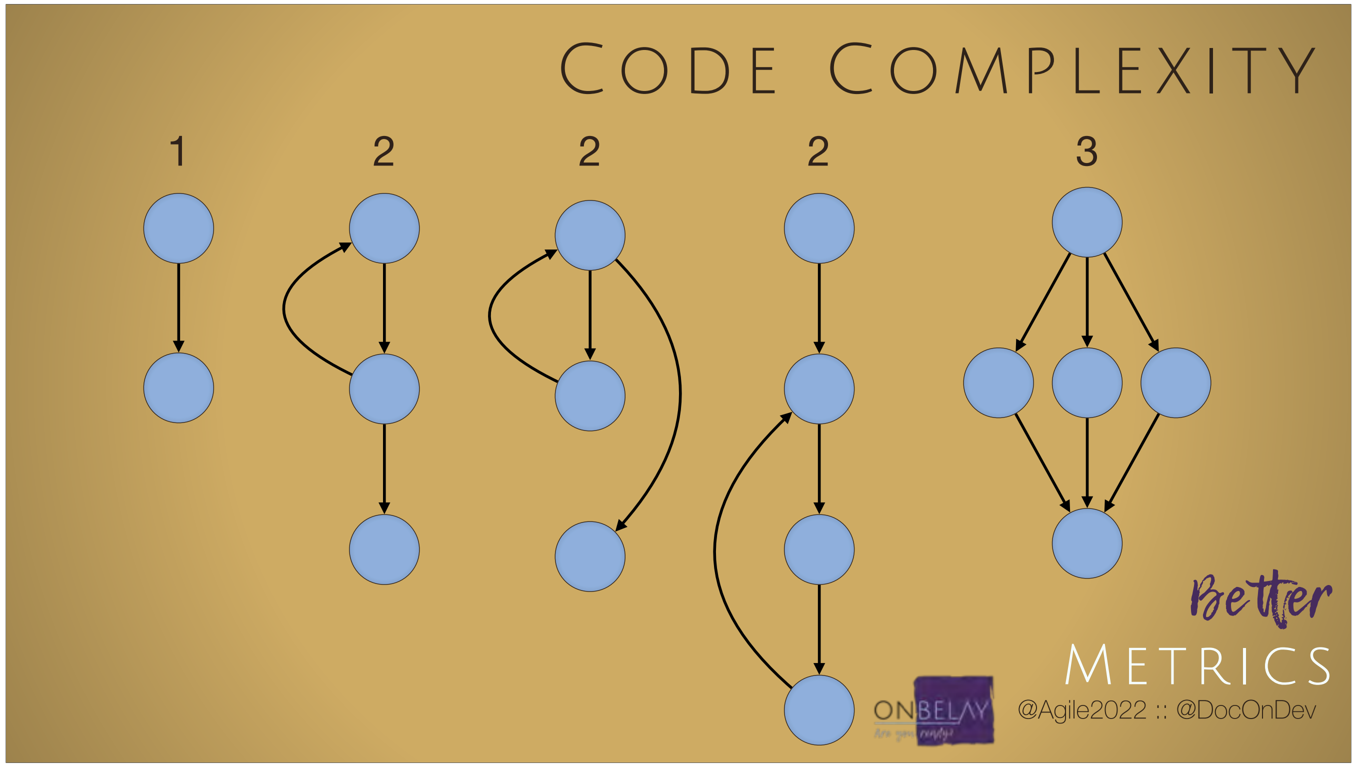
@Agile2022 :: @DocOnDev

You can use Cyclomatic or ABC Complexity.

Cyclomatic Complexity, also known as McCabe's number is based on nodes and edges in the code tree. It is essentially a count of linearly independent paths in the code. It was designed as a means of determining the number of tests you need for a piece of code.

ABC is similar to Cyclomatic Complexity, but is based on Assignments, Branches, and Conditionals, so it is a bit more robust. It was originally intended to be used as a means of forecasting.

Today, both of these measures are used as a proxy for code quality.



CODE COMPLEXITY

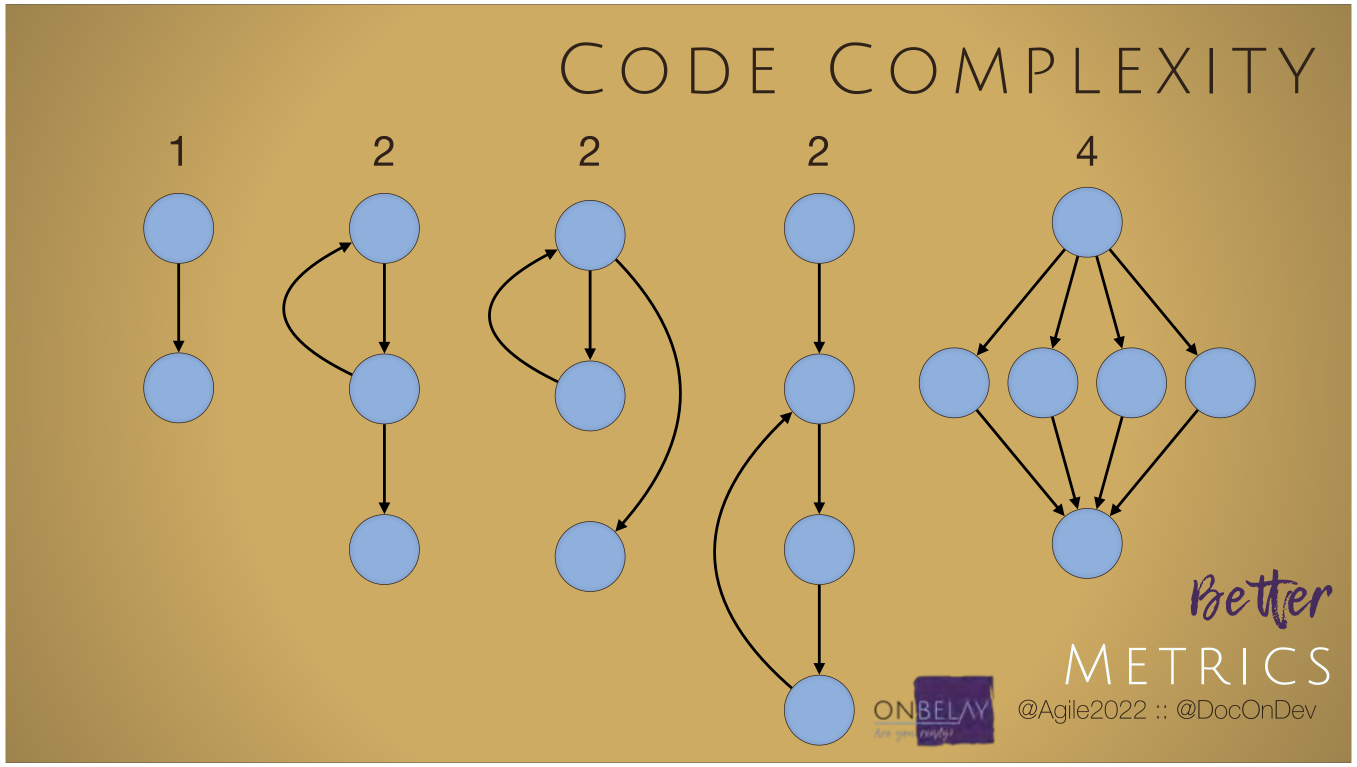
Better METRICS

@Agile2022 :: @DocOnDev

A strictly linear program has a cyclomatic complexity of 1 <next>

Whereas a Do Until, While, and For all have <next> a complexity of 2

A case statement <next> grows in complexity by one for every option<next>



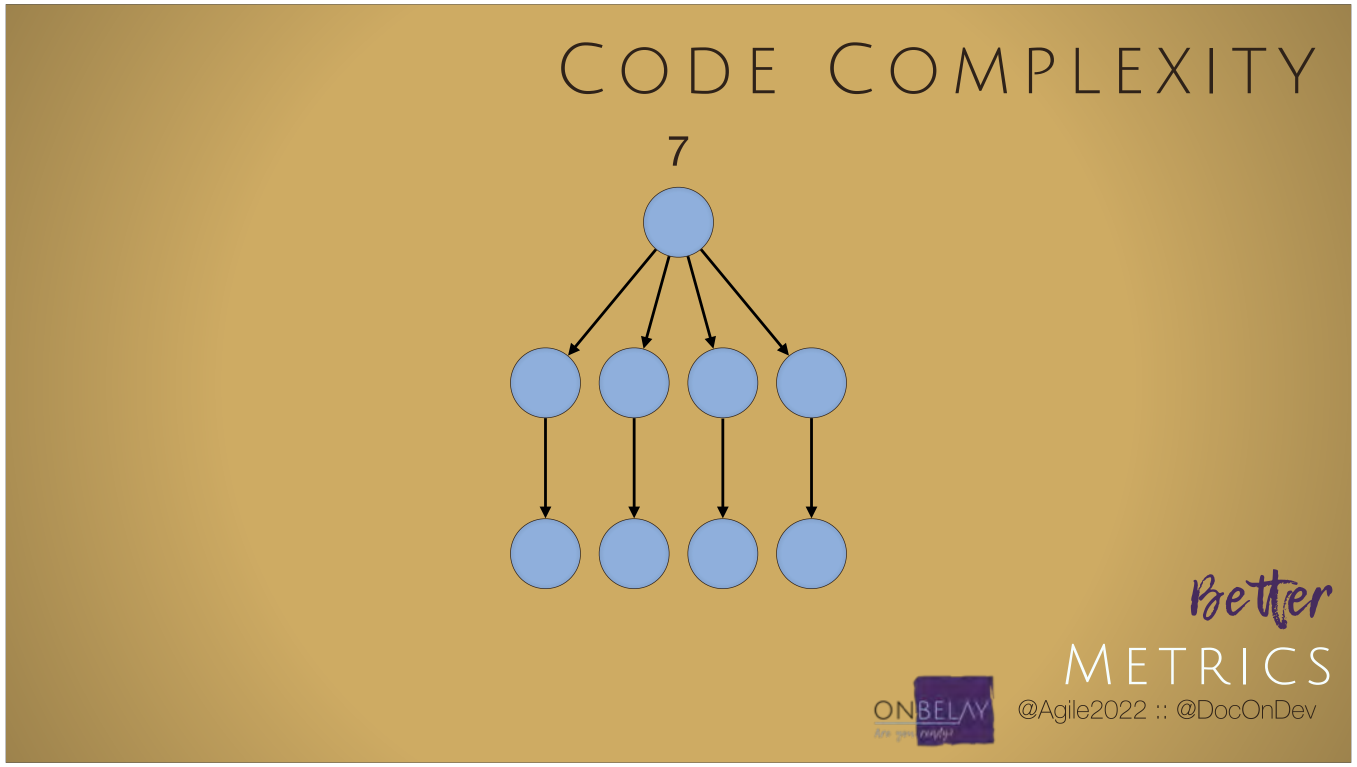
Code Complexity

Better Metrics

ONBELAY

@Agile2022 :: @DocOnDev

And if those case statements each exit the program, <next>



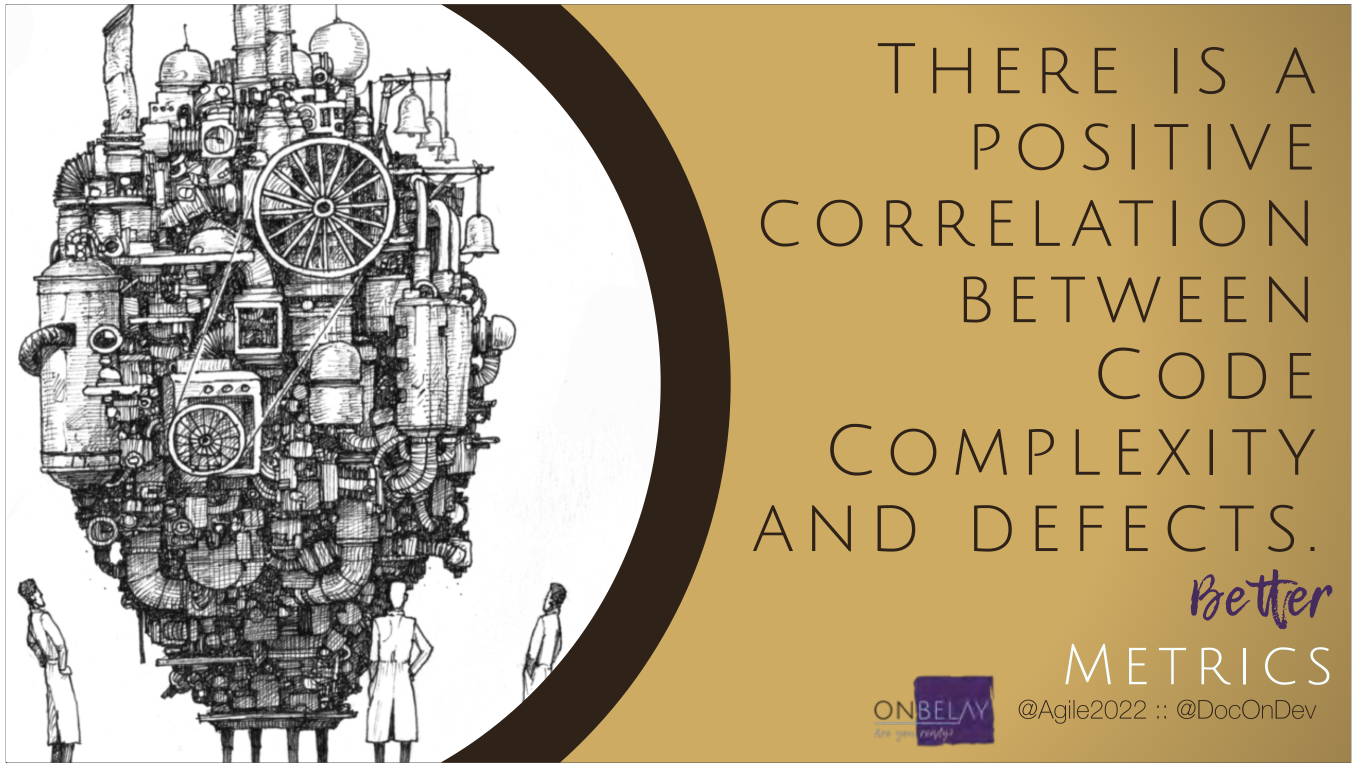
Code Complexity

Better Metrics

@Agile2022 :: @DocOnDev

the complexity increases by 2 for every new option

Why do we care about this?
Because <next>



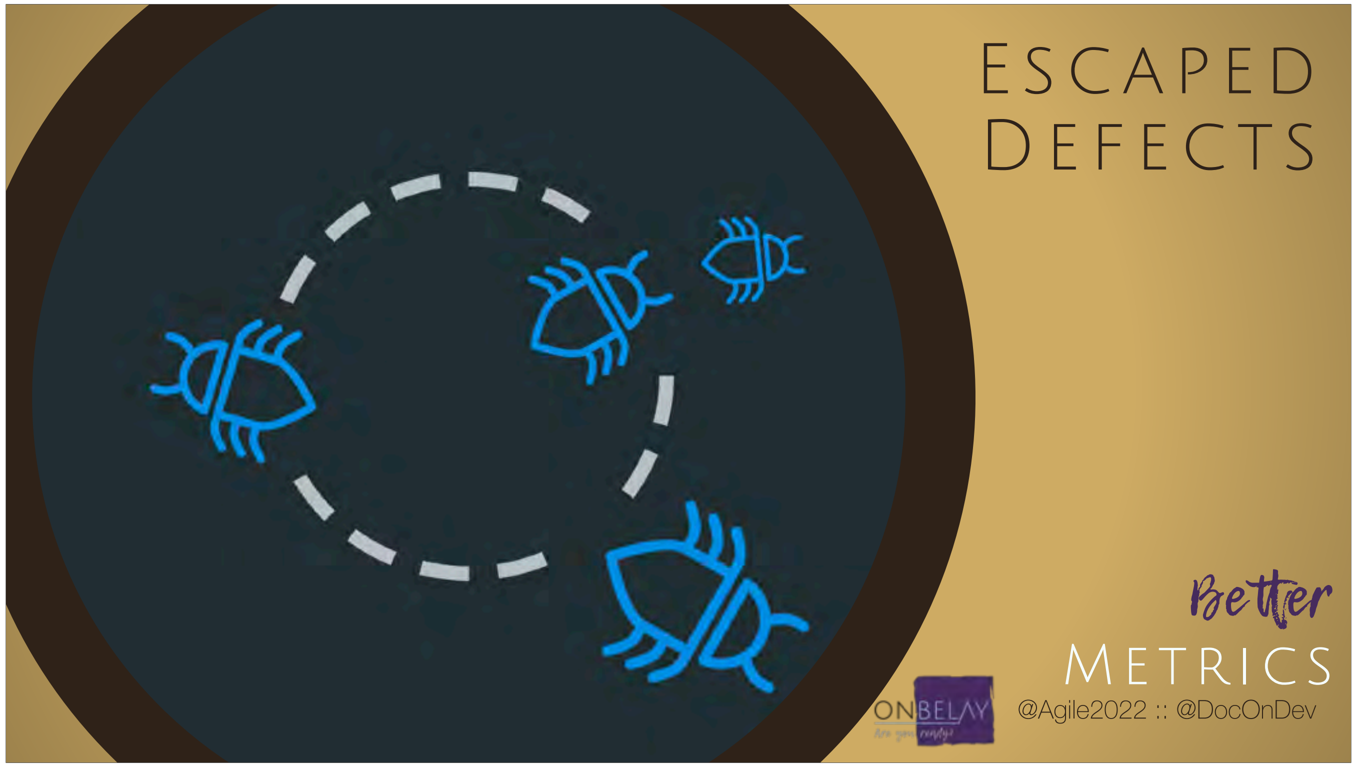
THERE IS A POSITIVE CORRELATION BETWEEN CODE COMPLEXITY AND DEFECTS.

Better Metrics

@Agile2022 :: @DocOnDev

The higher the complexity, the more likely a method is doing too much or has low cohesion.
The more responsibility a method has or the lower the cohesion, the higher the likelihood there are defects lurking therein.

Many conditionals can be collapsed with some refactoring - You can take look at basic inheritance and the factory pattern for more on this.



Escaped Defects

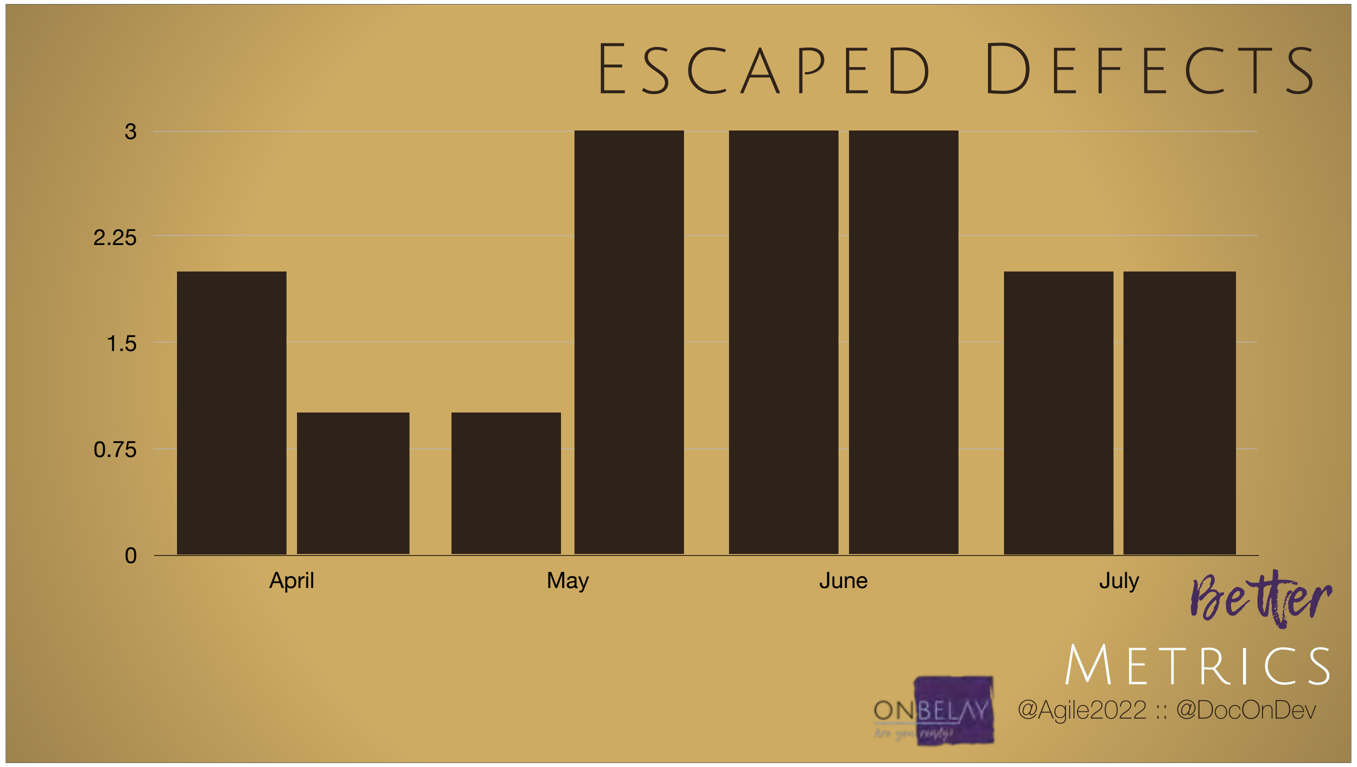
Better Metrics

@Agile2022 :: @DocOnDev

Count of defects found in production.

If at all possible log the defect against the release during which it was introduced. For some defects, this will be easy, but others might not rear their heads for weeks or even months.

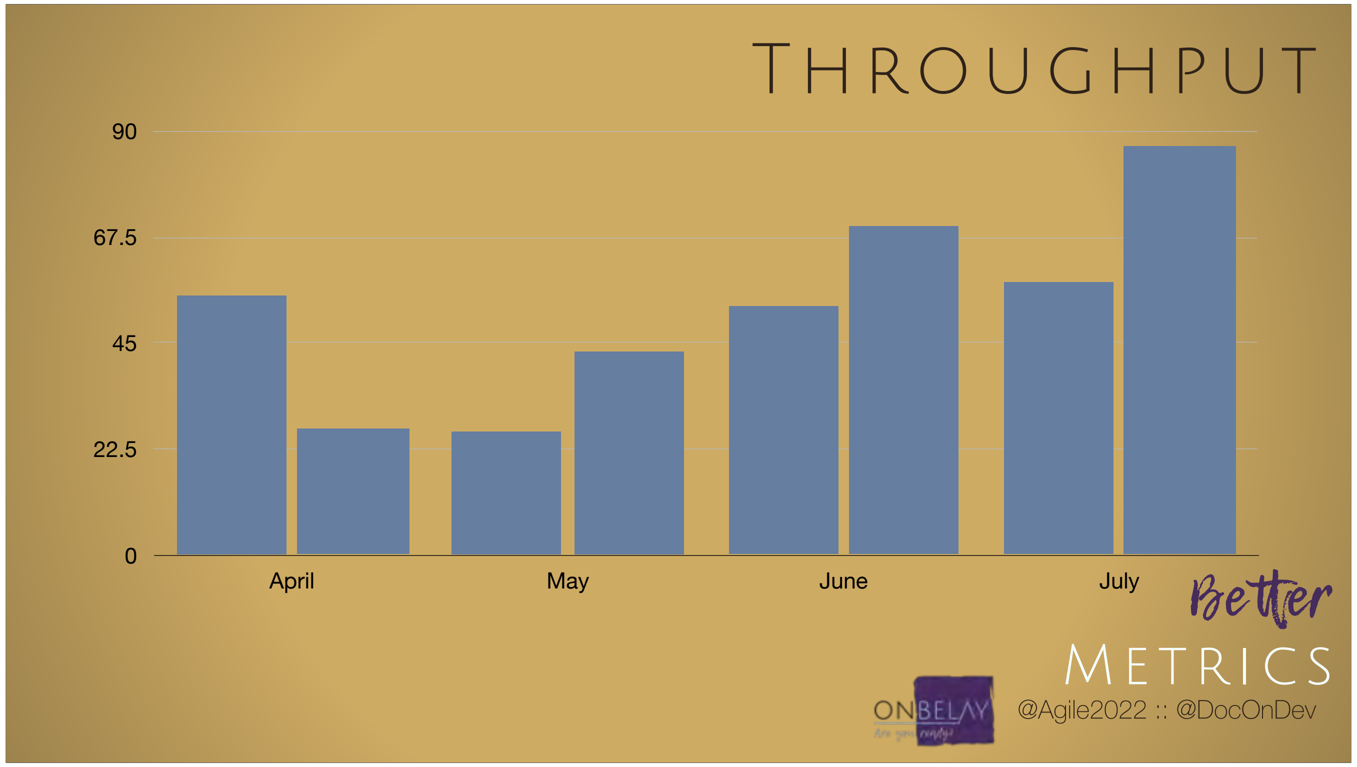
If it is in Development or UAT, it is not an escaped defect.



This is informative, but can be slightly misleading.

The number of defects escaped is devoid of any other information.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare team to team a bit better.



THROUGHPUT

Better METRICS

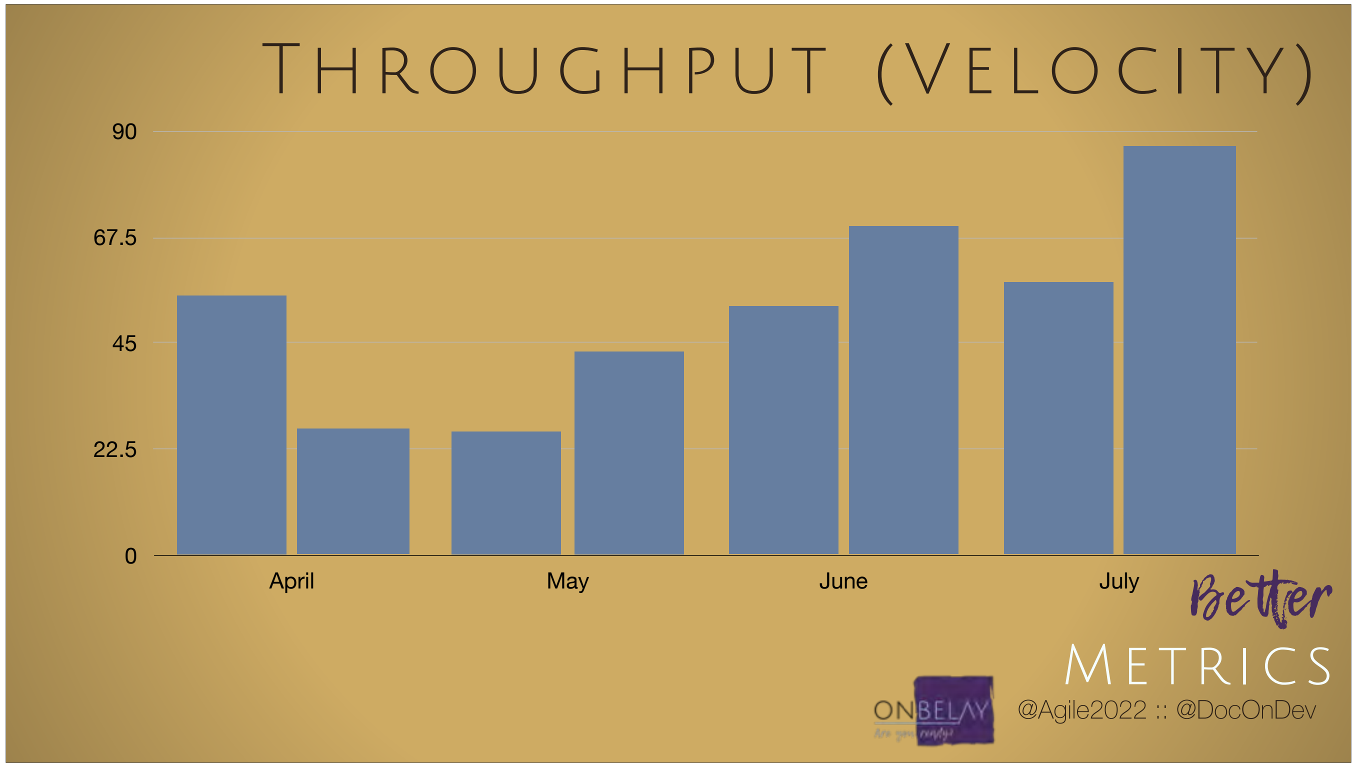
ONBELAY

@Agile2022 :: @DocOnDev

This is informative, but can be slightly misleading.

The number of defects escaped is devoid of any other information.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare team to team a bit better.



THROUGHPUT (VELOCITY)

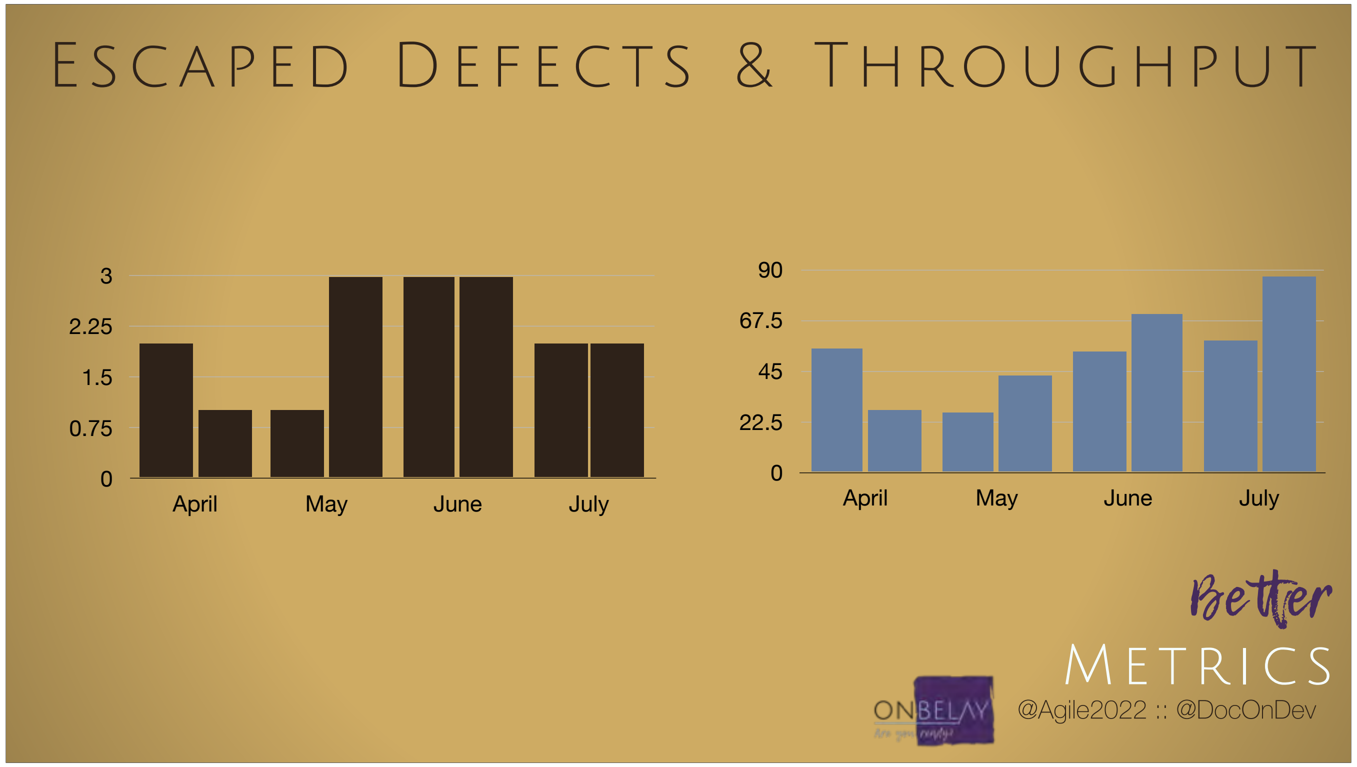
Better METRICS

@Agile2022 :: @DocOnDev

This is informative, but can be slightly misleading.

The number of defects escaped is devoid of any other information.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare team to team a bit better.



Escaped Defects & Throughput

Better
METRICS

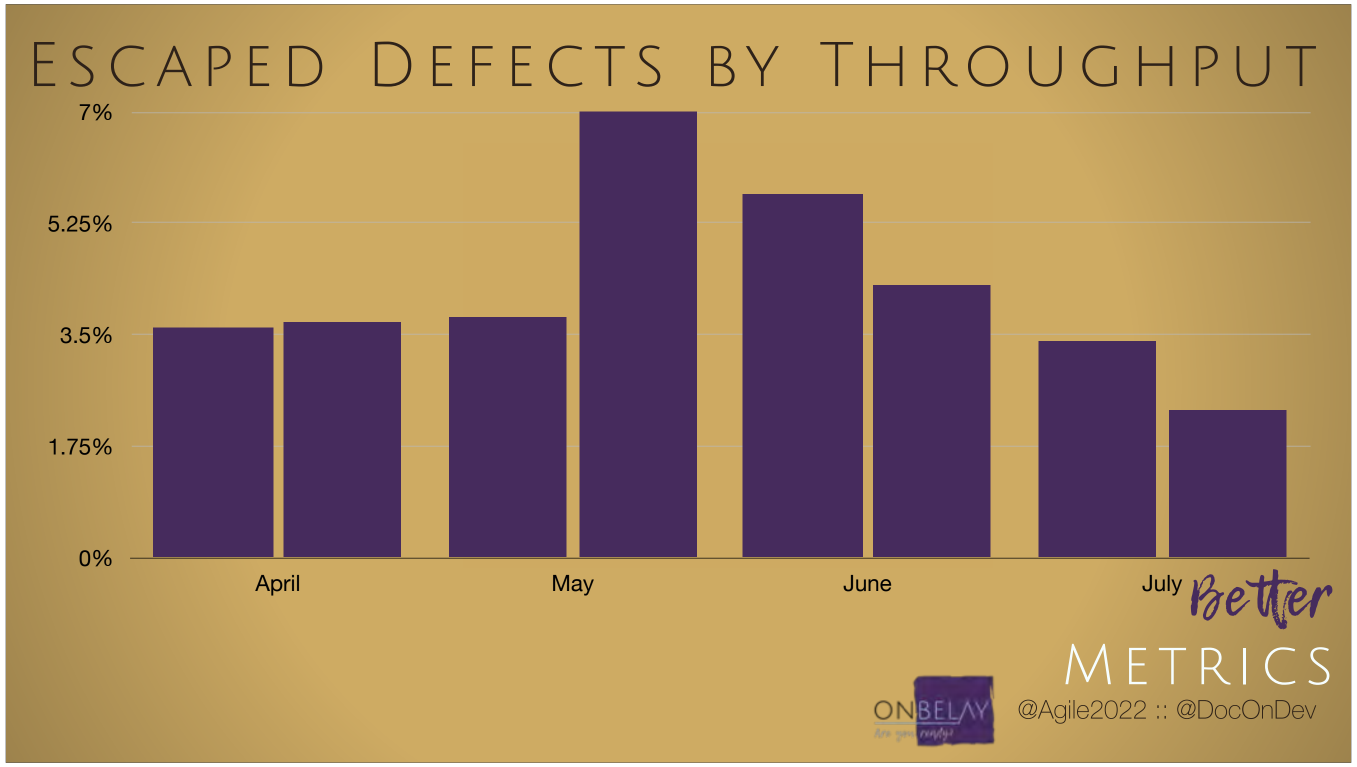
ONBELAY

@Agile2022 :: @DocOnDev

This is informative, but can be slightly misleading.

The number of defects escaped is devoid of any other information.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare team to team a bit better.



This is informative, but can be slightly misleading.

The number of defects escaped is devoid of any other information.

If, however, we look at defects as a percentage of throughput, we get a view into defect density and we can actually compare team to team a bit better.

Better
Metrics

The bigger the Batch Size, the greater the risk.

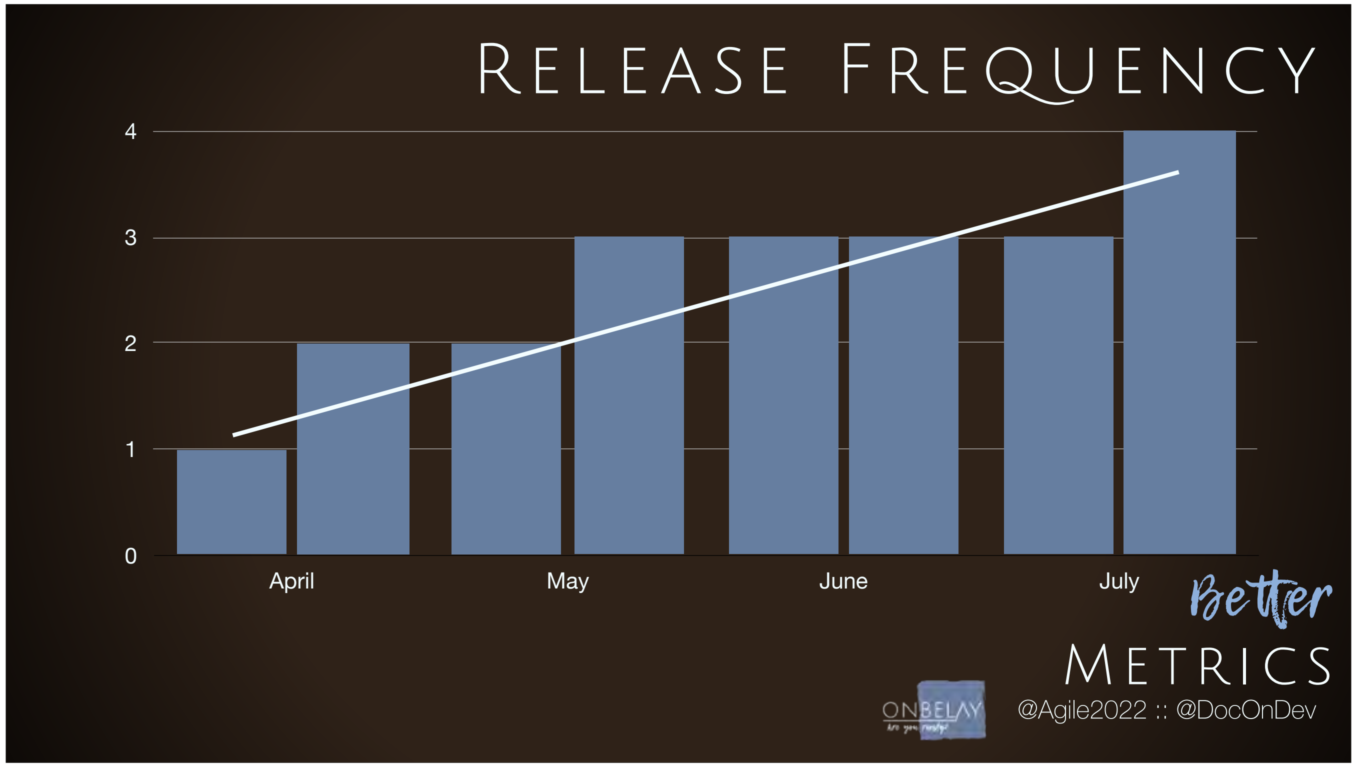
Better Metrics

@Agile2022 :: @DocOnDev

The bigger the batch size, the greater the risk.

One seemingly logical approach to this is to use story size as a proxy for batch size. And that can work, but I’d like to suggest some alternatives.

Release Frequency



Release Frequency

Better Metrics

@Agile2022 :: @DocOnDev

Assuming all other things equal, the more often we release, the necessarily smaller the release.

So we can effectively use release frequency as a proxy for release size. Our release size IS our batch size.

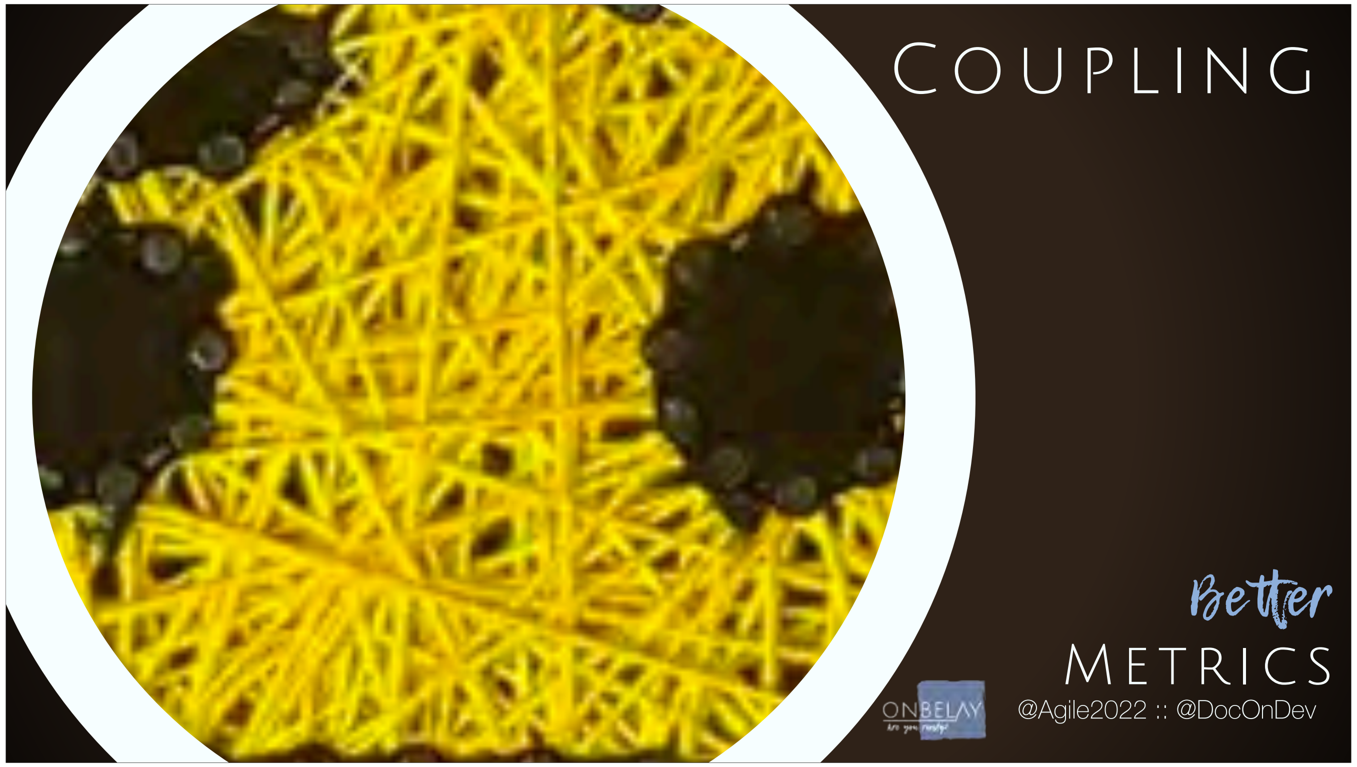
Small Releases Beat Big Releases.

Better Metrics

ONBELAY @Agile2022 :: @DocOnDev

small releases provide numerous advantages over big releases

Optionality, flexibility, learning, and safety (yes safety. Small releases have a smaller surface area and a smaller impact. They provide less risk.) -



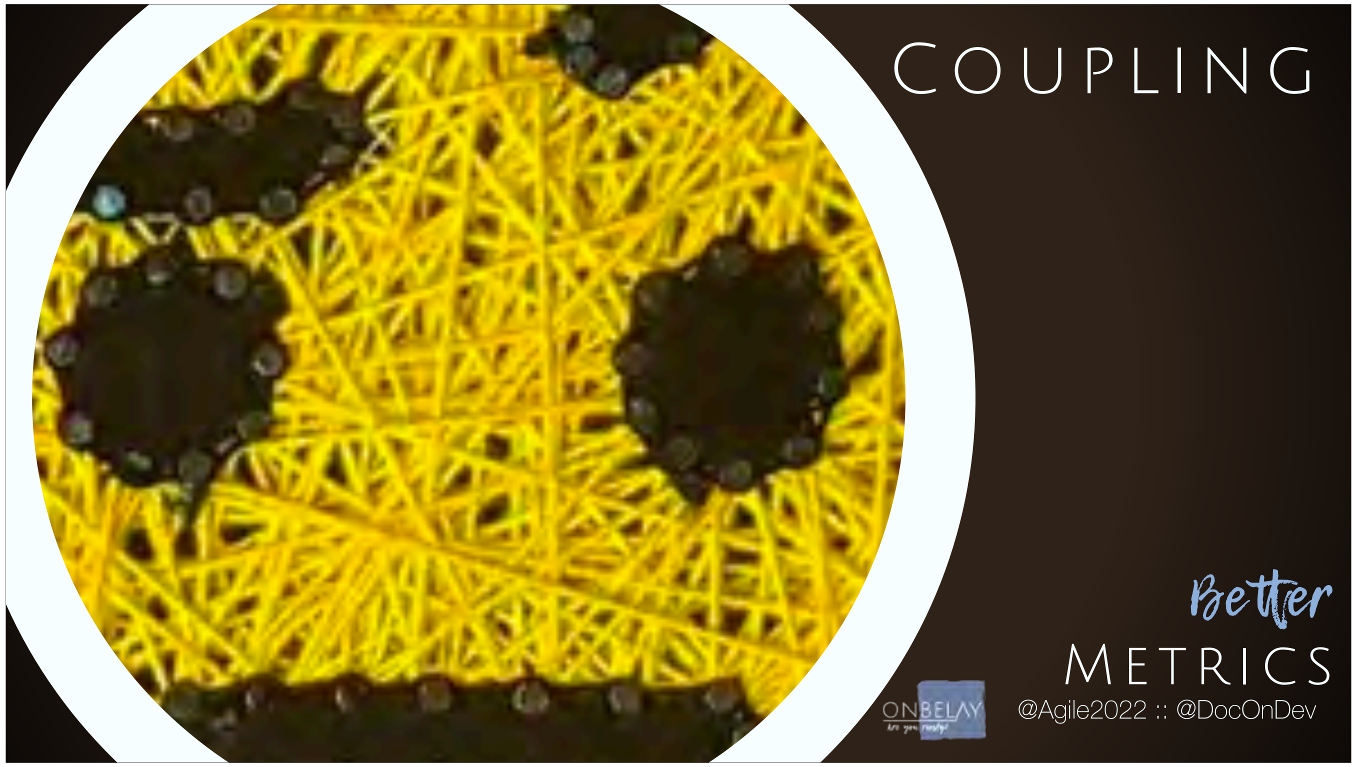
Coupling

Better Metrics

@Agile2022 :: @DocOnDev

Coupling refers to the dependencies associated with a piece of code.

Afferent coupling is the number of inbound dependencies
Efferent coupling is the number of outbound dependencies



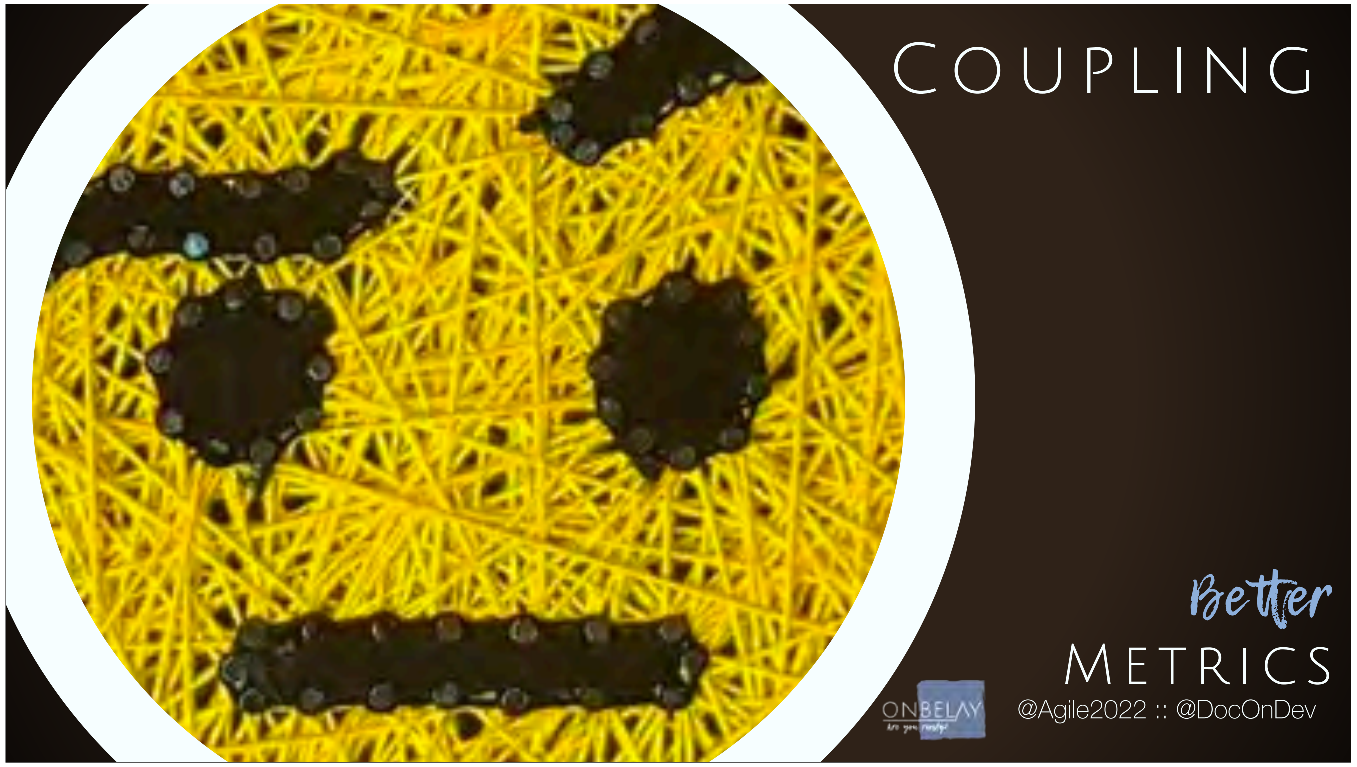
Coupling

Better Metrics

@Agile2022 :: @DocOnDev

Coupling refers to the dependencies associated with a piece of code.

Afferent coupling is the number of inbound dependencies
Efferent coupling is the number of outbound dependencies



COUPLING

Better METRICS

@Agile2022 :: @DocOnDev

Coupling refers to the dependencies associated with a piece of code.

Afferent coupling is the number of inbound dependencies
Efferent coupling is the number of outbound dependencies

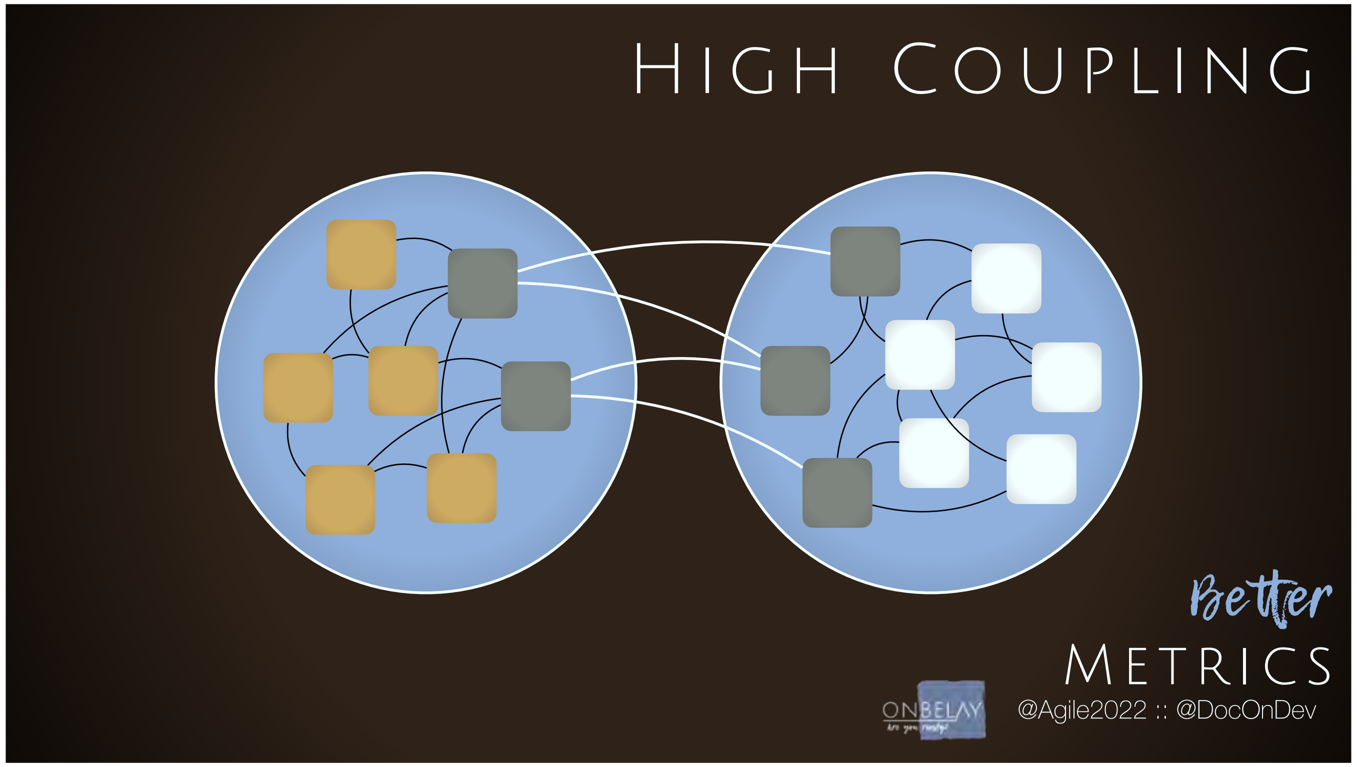
Coupling

Better Metrics

@Agile2022 :: @DocOnDev

The more tightly coupled the code, the more places you need to make changes when new functionality is introduced. The more places you need to make changes, the larger your batch size.

Tools like NDepend or JDepend will graph and quantify your coupling. The lower, the better.



HIGH COUPLING

Better METRICS

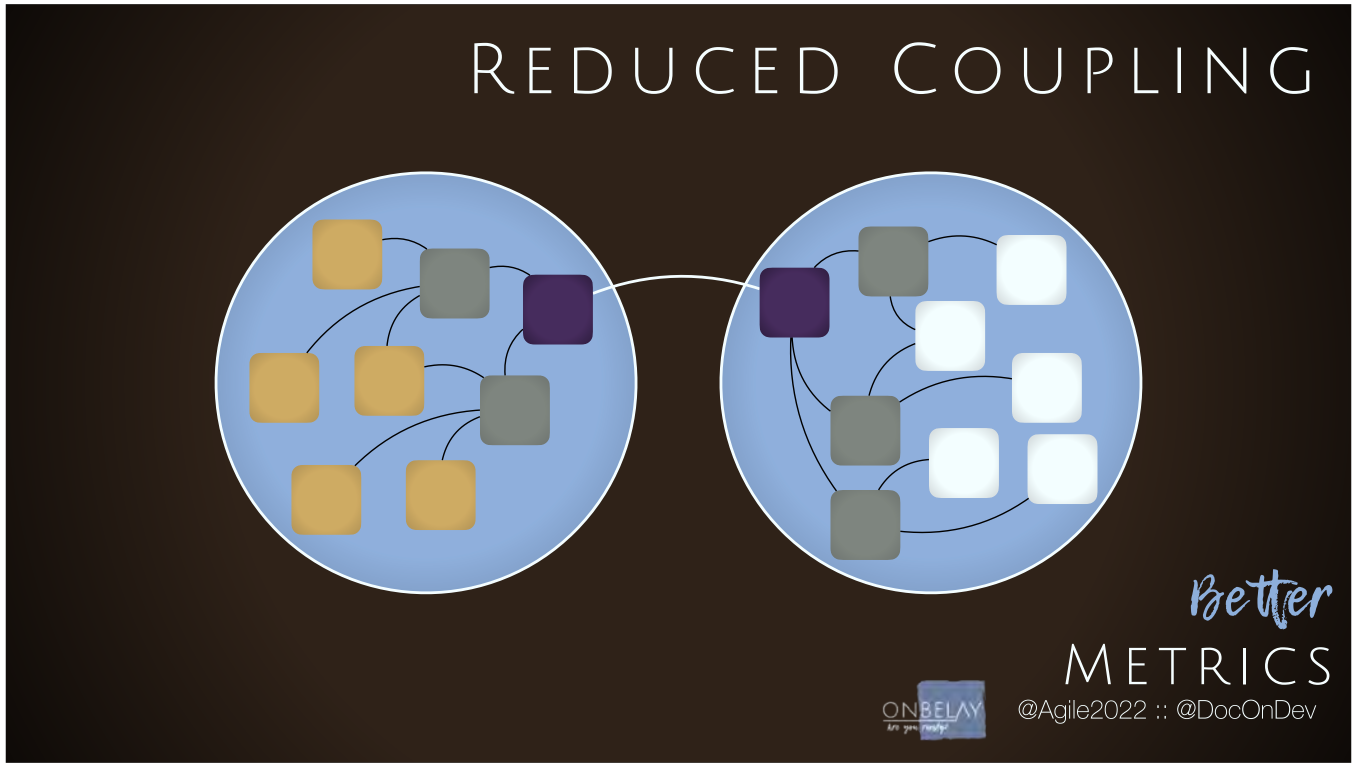
@Agile2022 :: @DocOnDev

These could be any number of things in a code-base

Shopping Cart and Cash Register
Employee Management and Payroll System

They seem to know a lot about one another.
Changes to Cash Register would potentially break Shopping Cart. Does that really make sense?
Changes to the payroll system could potentially break the Employee. Does that really make sense?

In addition, they seem to have a lot of connection paths internally.



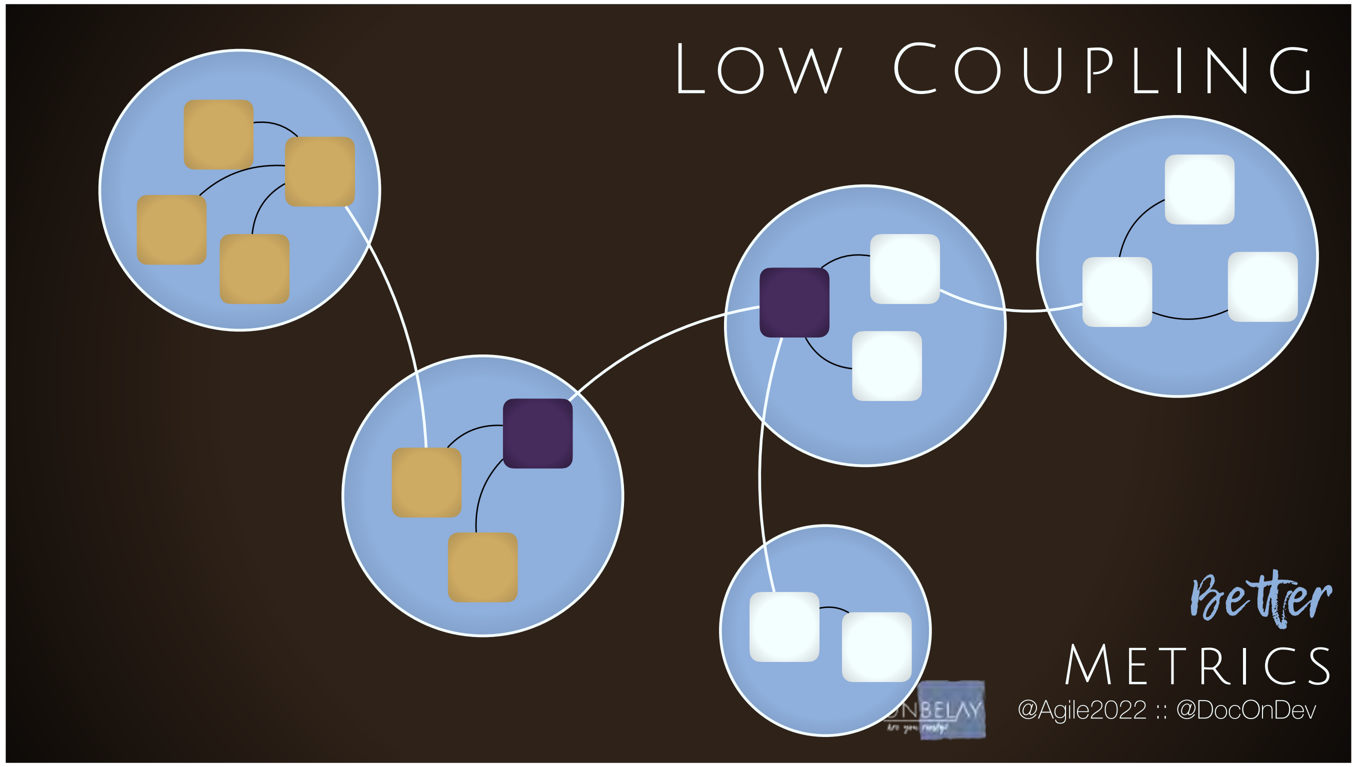
Reduced Coupling

Better Metrics

@Agile2022 :: @DocOnDev

We can start by separating these large logical items and adding an adapter or API that insulates the external components from internal changes.

We can usually reorganize the internal code as well to reduce the connections.



Low Coupling

Better
Metrics
@Agile2022 :: @DocOnDev

Polymorphism
Composition
Various Strategies or Patterns

Code that has low coupling tends to have higher cohesiveness as well. The code in the class is logically related and serves a single purpose.

Better
Metrics

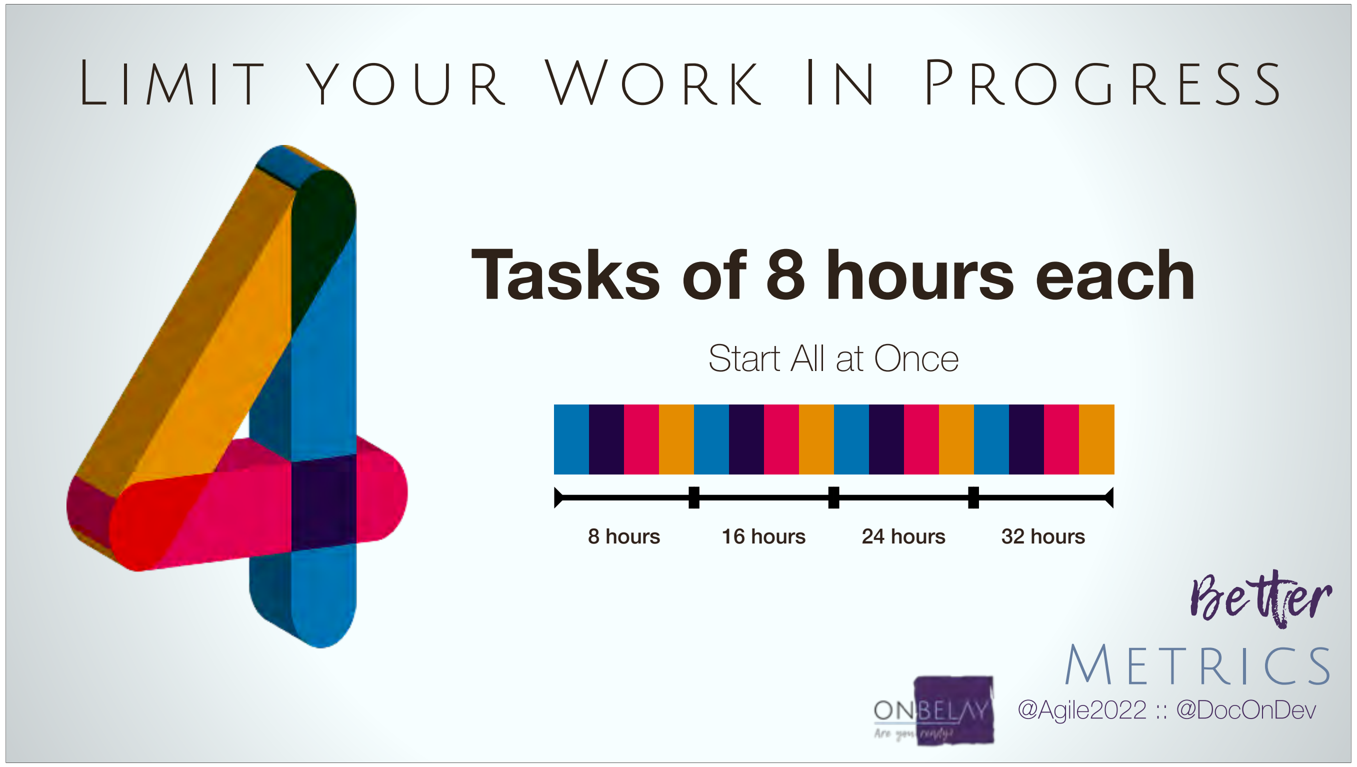
Limit Work In Progress

Stop Starting. Start Finishing.

Better Metrics

@Agile2022 :: @DocOnDev

Don’t start more work. Focus on finishing the work you’ve already started.



Limit your Work In Progress

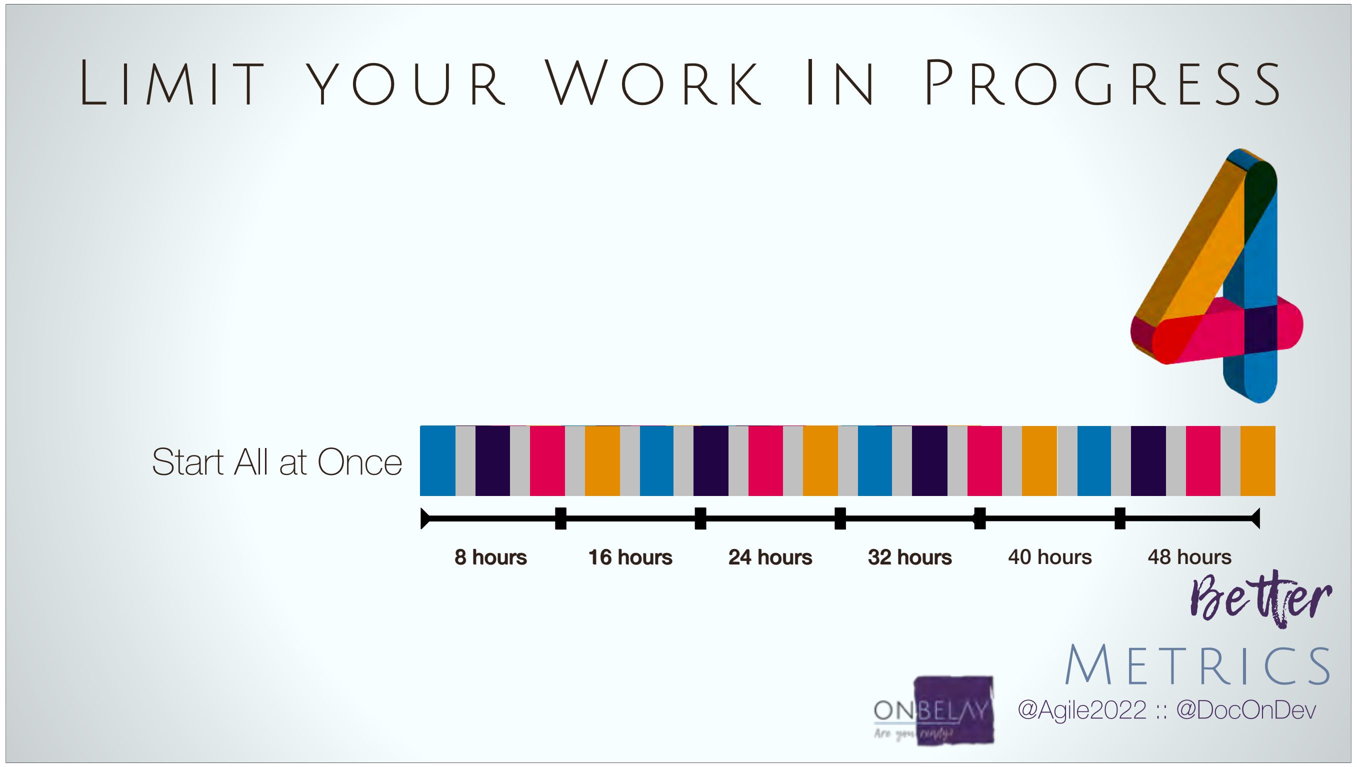
Tasks of 8 hours each

Start All at Once

Better METRICS

@Agile2022 :: @DocOnDev

Complete them one at a time, the first item will be done 8 hours after starting and all will be complete in 32 hours



Limit your Work In Progress

Start All at Once

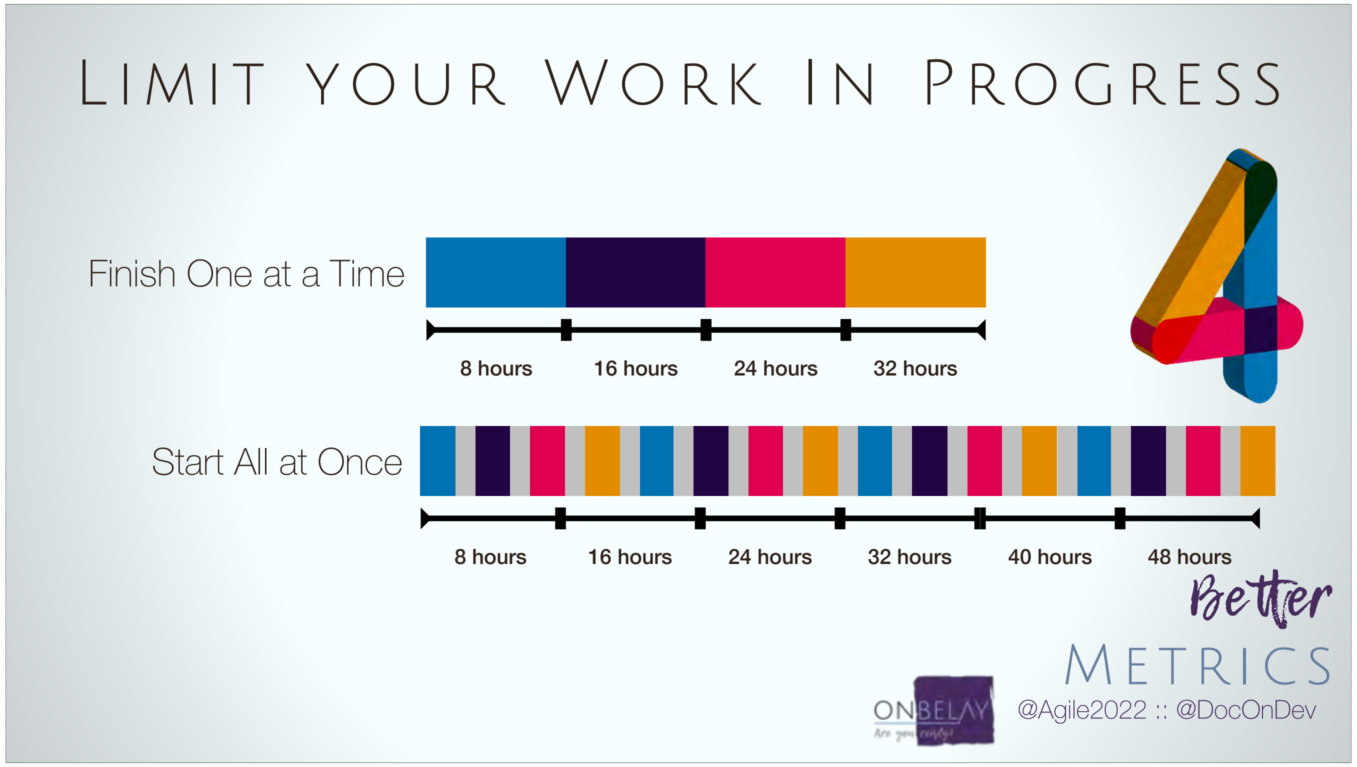
Better METRICS

ONBELAY Are you ready?

@Agile2022 :: @DocOnDev

Complete them in parallel, working on each one for 2 hours and we'd hope that the first one is done in 26 hours with all complete in 32 hours. <next>

But there is a tax for context switching. Studies have shown this tax is usually about 20% for each new item — compounded. So, in reality, it will take approximately 48.75 hours to complete all four tasks.



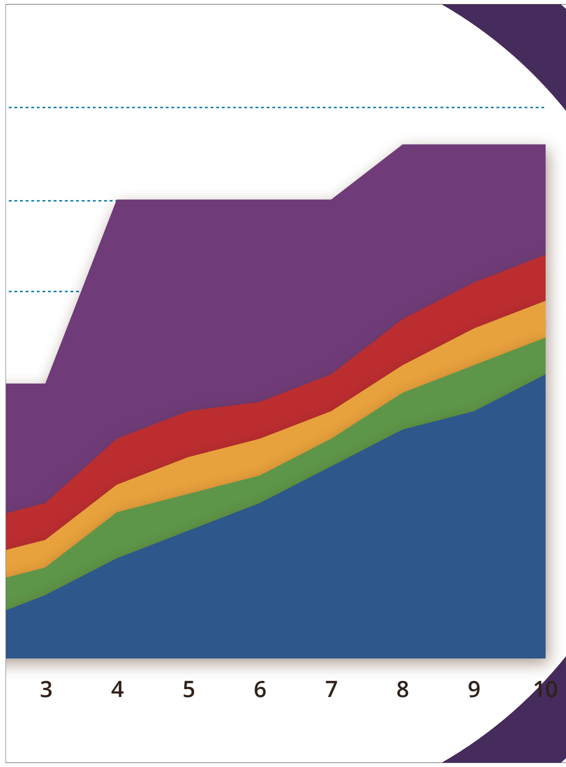
Limit your Work In Progress

Better METRICS

@Agile2022 :: @DocOnDev

Complete them in parallel, working on each one for 2 hours and we'd hope that the first one is done in 26 hours with all complete in 32 hours. <next>

But there is a tax for context switching. Studies have shown this tax is usually about 20% for each new item — compounded. So, in reality, it will take approximately 48.75 hours to complete all four tasks.

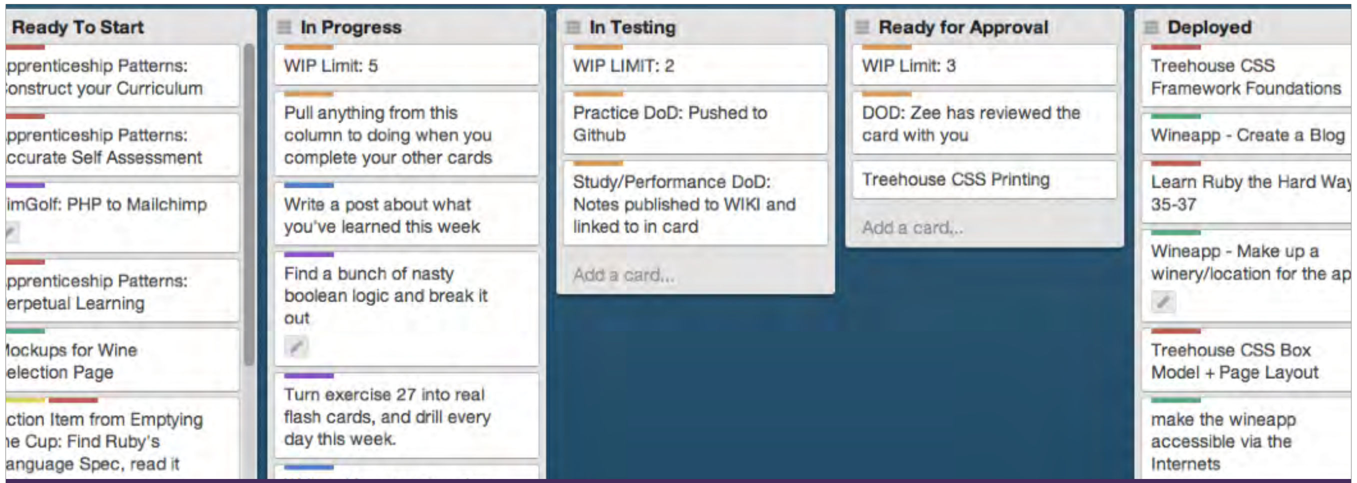


Cumulative Flow Diagram

Better Metrics

ONBELAY

@Agile2022 :: @DocOnDev



Ready To Start

- pprenticeship Patterns: onstruct your Curriculum
- pprenticeship Patterns: ccurate Self Assessment
- imGolf: PHP to Mailchimp
- pprenticeship Patterns: erpetual Learning
- lockups for Wine election Page
- ction Item from Emptying ne Cup: Find Ruby's anguage Spec, read it

In Progress

- WIP Limit: 5
- Pull anything from this column to doing when you complete your other cards
- Write a post about what you've learned this week
- Find a bunch of nasty boolean logic and break it out
- Turn exercise 27 into real flash cards, and drill every day this week.

In Testing

- WIP LIMIT: 2
- Practice DoD: Pushed to Github
- Study/Performance DoD: Notes published to WIKI and linked to in card
- Add a card...

Ready for Approval

- WIP Limit: 3
- DOD: Zee has reviewed the card with you
- Treehouse CSS Printing
- Add a card...

Deployed

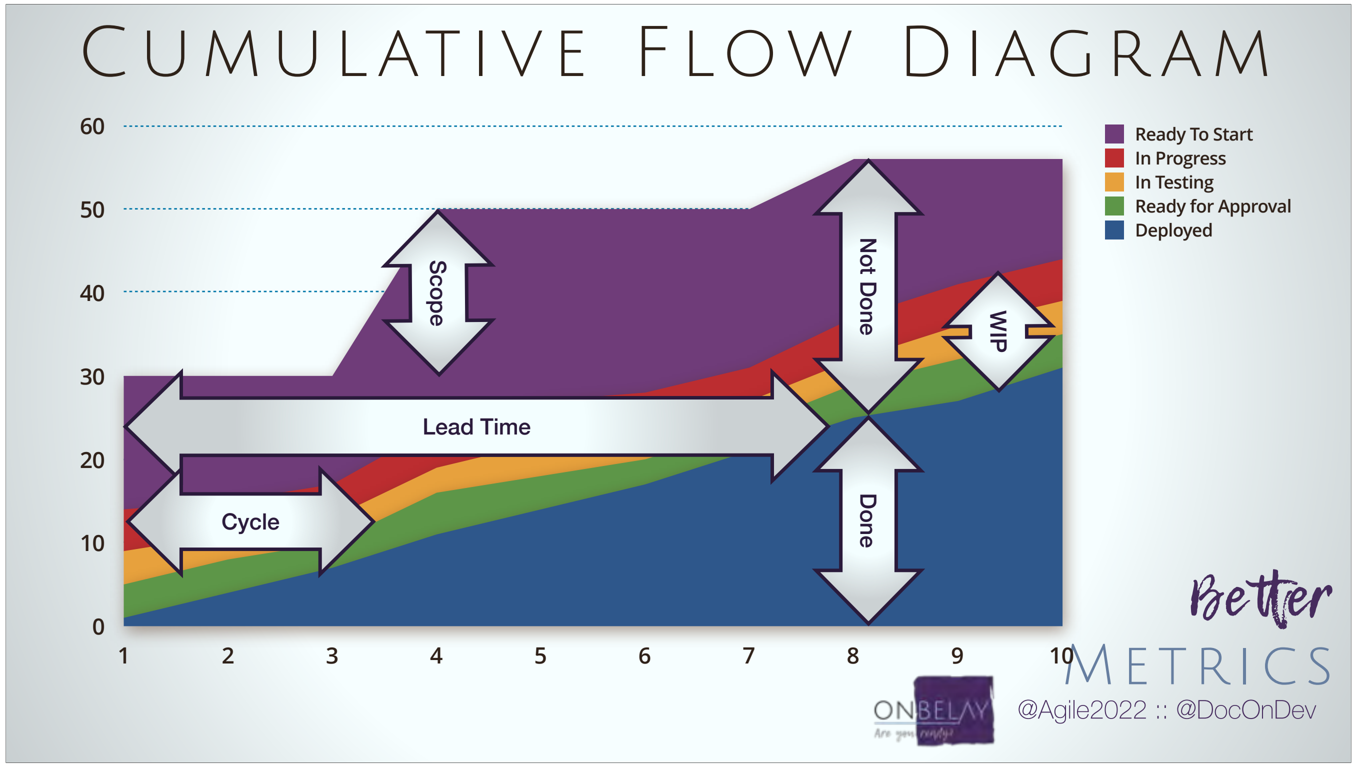
- Treehouse CSS Framework Foundations
- Wineapp - Create a Blog
- Learn Ruby the Hard Way 35-37
- Wineapp - Make up a winery/location for the ap
- Treehouse CSS Box Model + Page Layout
- make the wineapp accessible via the Internets

Cumulative Flow Diagram

Sample Backlog

ONBELAY Are you ready?

@Agile2022 :: @DocOnDev



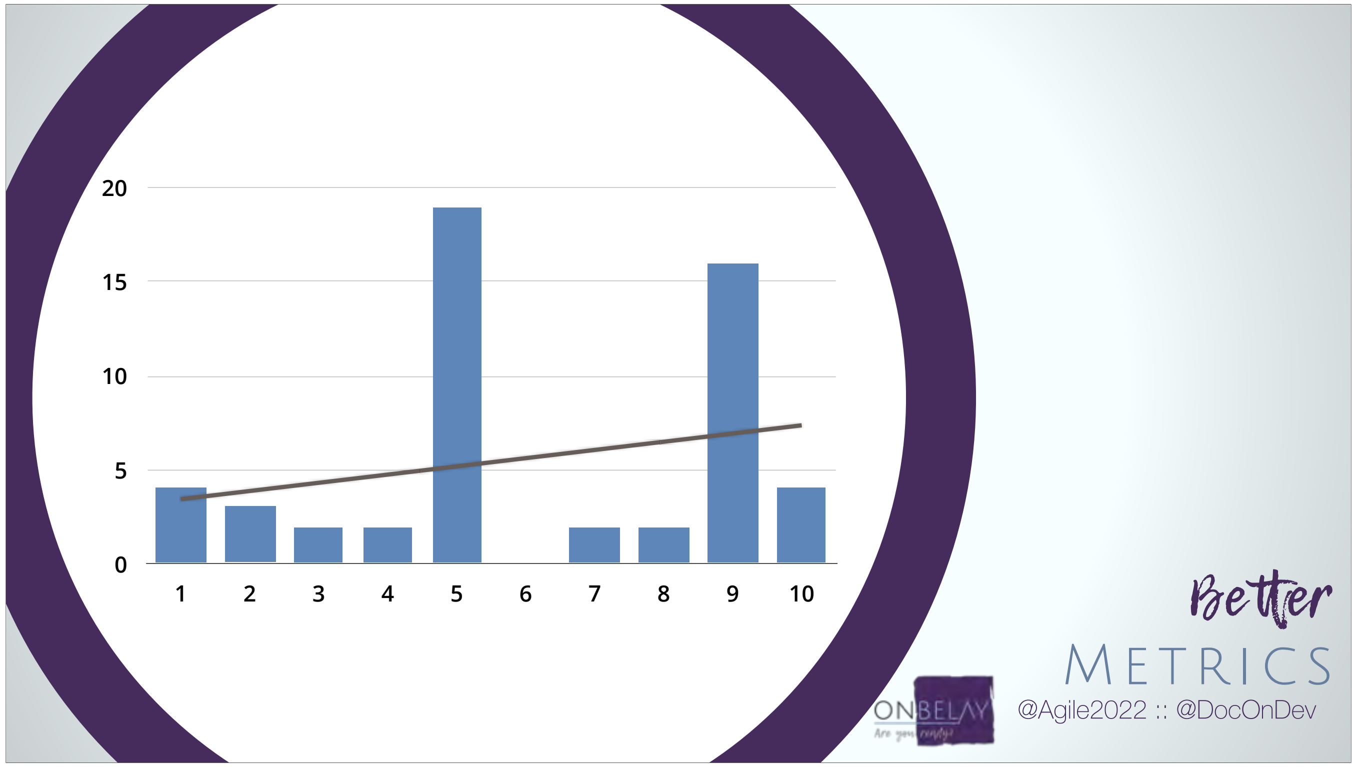
CUMULATIVE FLOW DIAGRAM

@Agile2022 :: @DocOnDev

Looking at this diagram, we can see <next>
work done and work not done <next>
The amount of Work in Progress at any given time <next>
The lead time - hey wouldn’t it be nice to hay, isn’t that nice <next>
And the cycle time - the amount of time an item is actually being worked on <next>

Finally, we can see changes in scope whenever our top line moves.

For a team that is operating well, this graph has relatively smooth lines that move together up and to the right



Better

METRICS

@Agile2022 :: @DocOnDev

Remember this chart from earlier?

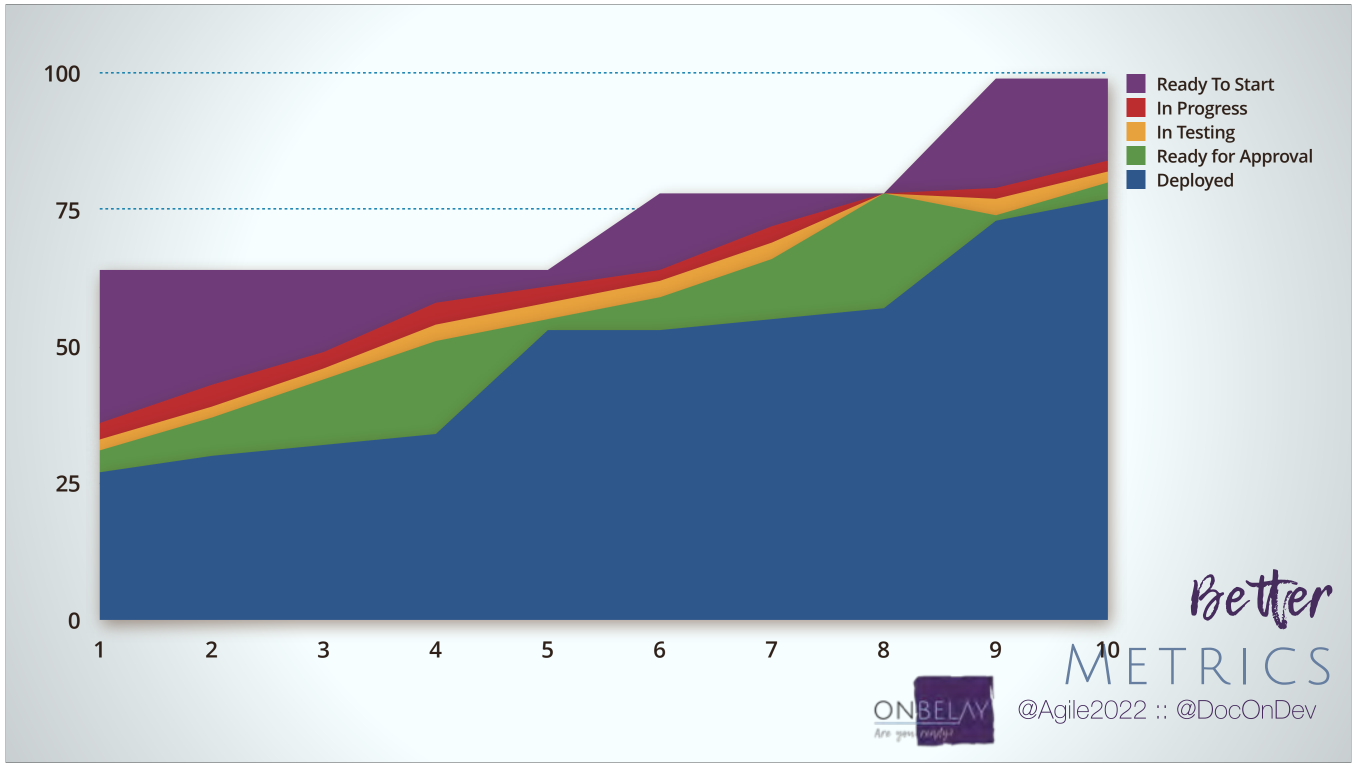
We couldn't say for sure what was the issue for this team.

Same Team
Same Data
Different Perspective

Better Metrics

ONBELAY

@Agile2022 :: @DocOnDev



Can anyone tell me what is “wrong” with this team?

Product owner is a traveling salesperson. On the road, doesn’t have time. Comes back and approves and then adds to the backlog.

Metrics are not for managers.

Metrics are not just for managers.

Metrics are not>Just for managers.

Metrics are for Teams.

Metrics are for Teams.

THANK YOU!

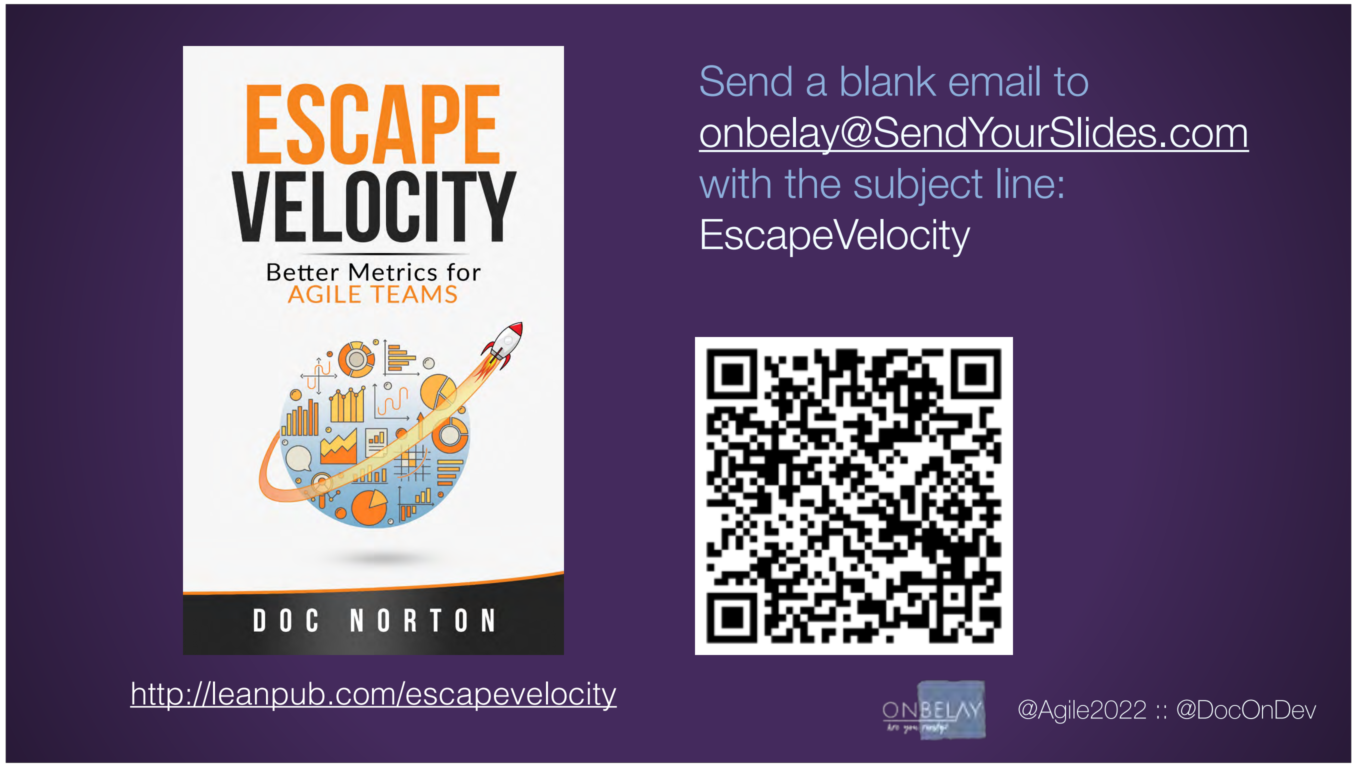
@Agile2022 :: @DocOnDev

Everybody stand up.

I'll let you know when to sit down. I won't make you stand long, I promise.

We're here today to talk about better metrics for agile teams.

Let's start with Velocity <next>



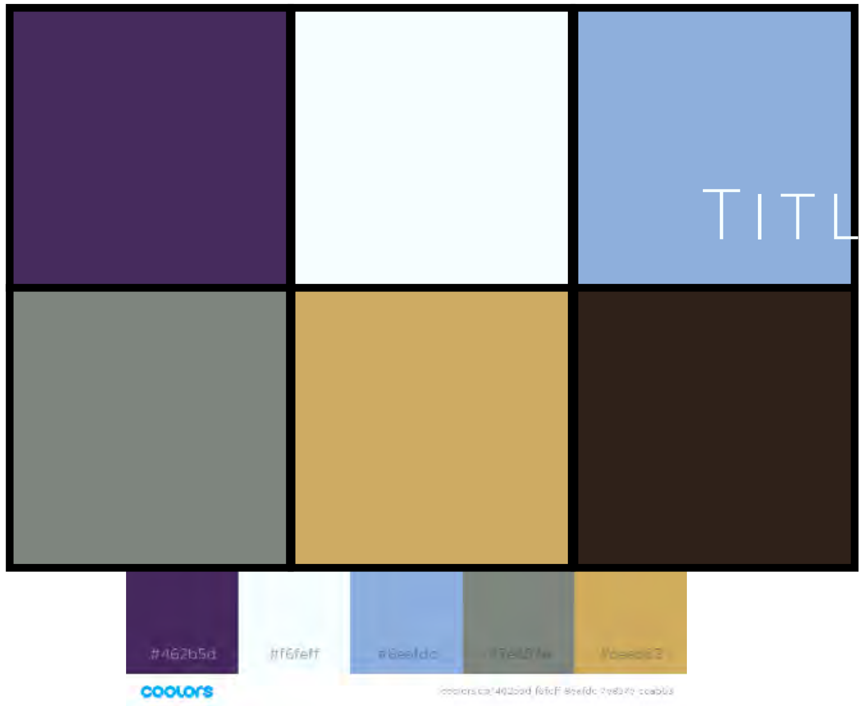
Send a blank email to
onbelay@SendYourSlides.com
with the subject line:
EscapeVelocity

http://leanpub.com/escapevelocity

ONBELAY

@Agile2022 :: @DocOnDev

To get all of the supporting material, send a blank email to onbelay@sendyourslides.com with the subject line EscapeVelocity (ALL ONE WORD)



TITLE

Helvetica Neue Regular

Helvetica Neue Bold

Helvetica Neue Thin

Garamond

Title Small

Subtitle

Body

Body Small

Caption

Quote

– *Attribution*

http://fontawesome.io/cheatsheet/