DevOps Metrics 101:
What really matters when measuring performance from a DevOps angle
Utilization

\[ N = \frac{\rho^2}{1 - \rho^2} \]

Rho = Capacity Usage

WIP

MTTR

Deploy Frequency

LT, CFR, TP

NPS:
Learning Outcomes

1. Define the types of metrics used for DevOps transformations.

2. Show how these metrics are measured and interpreted.

3. Identify top three ways to begin capturing and using DevOps metrics.
Beware the Red Yellow Green (RYG) Report

<table>
<thead>
<tr>
<th>PROJECTS</th>
<th>MGR</th>
<th>SCHEDULE</th>
<th>BUDGET</th>
<th>SCOPE</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>John</td>
<td>Green</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Sarah</td>
<td>Green</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Jerry</td>
<td>Green</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Julia</td>
<td>Green</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>Adam</td>
<td>Green</td>
<td>Yellow</td>
<td>Red</td>
<td>Conflicting Priorities</td>
</tr>
<tr>
<td>E</td>
<td>Ann</td>
<td>Green</td>
<td>Yellow</td>
<td>Green</td>
<td></td>
</tr>
</tbody>
</table>

“If we have data, let’s look at data. If all we have are opinions, let’s go with mine.”
~ Jim Barksdale

Think about when you visit a badly designed website and how little you trust it.
Literature on DevOps performance metrics

1. Delivery lead time
2. Deploy frequency
3. Mean time to recover
4. Change failure rate
1. Lead Time: elapsed time (work time + wait time)

From code commit to code running successfully in prod

From customer request to code running successfully in prod
Why Delivery Lead Time matters

HURRICANE HARVEY

High volume of flooded cars

Business ask: optimize online claims process

IT response: collaborative design and implementation in 8 hours

Results: streamlined online claims process by 40%

Jim Grafmeyer & Cindy Payne  https://www.youtube.com/watch?v=9WAiFAgkO5g.
DevOps Handbook experiments in accelerating delivery at Nationwide
Delivery Lead Time con’t

Rework during build/test/deploy can increase when tech debt (code quality and/or architecture issues) is not addressed.

Measuring DLT over time helps us see trends and discover what needs to improve in build/test/deploy part of the value stream.

Damon Edwards @damonedwards
http://www.rundeck.comvops/
2. Deploy Frequency
Why Deployment Frequency matters

Code is hard to see and spoils quickly if not integrated into production.

The more frequent deployments are, the smaller the batch size is.

Small batches accelerates feedback and reduces WIP which improves lead times, quality, & efficiency.
Knowledge work is perishable

Transaction costs:
Low for a one-time 6 month supply
High for a one day supply.
While economies of scale can reduce costs in manufacturing, software is a different story.

Two things to consider:
- Transaction cost
- Holding cost
3. Mean Time to Recover (MTTR)

MTTR = downtime / # of incidents

2 incidents in Dec had combined downtime of 120 min. Dec MTTR is 60 min.
Why MTTR matters

Failure is inevitable. Hardware and software are going to fail. Hope is not a strategy. Need to be able to detect failure, isolate it, & recover quickly. Ability to adapt quickly matters.

But...
4. Change Failure Rate (CFR)

Answers the Q: What % of changes to prod fail?

CFR = # of failed items / total # of work items completed

Failure - a change resulting in an outage or degraded service where hotfix, rollback or patch required.

Ex: 60 items completed in Dec, 20 of them resulted in a failure. Dec CFR is 30%.
Why Change Failure Rate (CFR) matters

Need some sort of quality metric. Quality is hard to measure. Measure something to help you make better decisions.

As a ratio, CFR provides a simple and effective way to identify opportunities.
“When you focus solely on shallow data you give up the return on investments that can be realized by deeper and more elaborate analysis.”

~John Allspaw

Ex: why did it make sense for someone to do that at that time?

Learning from incidents requires psychological safety and then some.

5. A Culture metric to gage team safety

Examples:
• On my team, failure causes inquiry and not blame.
• Our leadership is open to hearing bad news
• In my org, failures are learning opportunities and messengers are not punished.

@nicolefv  https://www.youtube.com/watch?v=avauW5FAWCw
## Typology of Organizational Culture

Westrum

<table>
<thead>
<tr>
<th>Pathological Power-oriented</th>
<th>Bureaucratic Rule-oriented</th>
<th>Generative Performance-oriented</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low cooperation</td>
<td>Modest cooperation</td>
<td>High cooperation</td>
</tr>
<tr>
<td>Messengers shot</td>
<td>Messengers neglected</td>
<td>Messengers trained</td>
</tr>
<tr>
<td>Responsibility shirked</td>
<td>Narrow responsibilities</td>
<td>Risks are shared</td>
</tr>
<tr>
<td>Bridging discouraged</td>
<td>Bridging tolerated</td>
<td>Bridging encouraged</td>
</tr>
<tr>
<td>Failure leads to scapegoating</td>
<td>Failure leads to justice</td>
<td>Failure leads to inquiry</td>
</tr>
<tr>
<td>Novelty crushed</td>
<td>Novelty leads to problems</td>
<td>Novelty implemented</td>
</tr>
</tbody>
</table>

Adding a culture metric to previous 4 metrics (Speed, batch size, quality, ability to adapt) and you are off to a better start on your DevOps journey.

But…. (there’s always a but)
The reason people like Patrick Dubois and Andrew Shafer started the DevOps movement in the first place was to address local optimization
It doesn’t matter how fast one piece of the value stream moves when other parts of the system lag.

@jonsmart  The PMO is Dead, Long Live the PMO – Barclays  https://www.youtube.com/watch?v=R-fol1vkPIM.
Improve your decision making even more with The five best metrics you’ve never met.

1. Flow time
2. Flow efficiency
3. The WIP report
4. The Aging report
5. Work type distribution
1. Flow Time

Yes, let's do this!
Why Flow time matters

Understanding the elapsed time it takes a request to go from, “Yes, let’s do this”, to working in production, helps you be more predictable.
Upstream Discovery Transparency included in team space

Specialists supporting multiple teams are pulled in different directions resulting in conflicting priorities.

Dependencies on specialists mean that people aren’t available when needed.

2. Flow Efficiency

\[
\frac{\text{WORK}}{\text{WAIT + WORK}} \times 100\% = \text{FLOW EFFICIENCY}
\]
Dev & Ops are more reliant upon Product Owners/Product Mgrs, who prioritize the work tech does.

We need their help to ensure that non-functional requirements get prioritized.
Why Flow Efficiency matters.

When the Business Partners with Tech and they do a Dojo

IT needs Leadership to conquer tech debt, especially for the 1st time

- Dojo is Japanese for “Place of the way”
- Whole team learning model pioneered by USA retail giant Target
- Connects learning to real business problems
- Psychologically safe environment where teams learn new skills, practices and technologies faster than they would on their own
- Expertise and resources brought to team
- ~ 6-weeks in duration

One 6 week Dojo session reduced testing from 3 weeks to 3 hours.

“No Fear Change”

The Dojo is the fastest and most effective way to adapt a team to an engineering culture and accelerate expertise in Agile, DevOps and the Cloud
3. The WIP Report
Why WIP matters

People have a finite amount of capacity

https://itrevolution.com/book/the-cornerstone-for-winning/
https://www.youtube.com/watch?v=qav1y7G15JQ
Why WIP matters

People have a finite amount of capacity

https://itrevolution.com/book/the-cornerstone-for-winning/
https://www.youtube.com/watch?v=qav1y7G15JQ
High WIP means that other items sit waiting for service longer.

The single most important factor that affects queue size is capacity utilization.
Queuing Theory: Applied statistics that studies waiting lines

Queuing Theory allows us to quantify the relationship between wait times and capacity utilization.

Wait times increase exponentially as utilization approaches 100%.

If the goal is speed, consider managing work by queues.

\[
N = \frac{\rho^2}{1 - \rho^2}
\]

\[
\rho = \text{Capacity Utilization}
\]

http://reinertsenassociates.com/books
WIP is a leading indicator
The WIP Report
## 4. The Aging Report

### Aging Report

<table>
<thead>
<tr>
<th>Title</th>
<th>Pri</th>
<th>Last Touched</th>
<th>State</th>
<th>Duration/Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fix IP Filter</td>
<td>1</td>
<td>1/10/17</td>
<td>Approved</td>
<td></td>
</tr>
<tr>
<td>Update HAProxy</td>
<td>2</td>
<td>1/14/17</td>
<td>Doing</td>
<td></td>
</tr>
<tr>
<td>Replace Host Name</td>
<td>1</td>
<td>1/22/17</td>
<td>Ready to Deploy</td>
<td></td>
</tr>
<tr>
<td>Install Tableau 10</td>
<td>1</td>
<td>1/20/17</td>
<td>In Review</td>
<td></td>
</tr>
</tbody>
</table>

- **Average Duration**
- **Duration**
- **Longer than Avg**

The number of days with no activity is 10.
Why Age of work items matter
5. Work Type Distribution

- Features
- Defects
- Risks
- Debts
## How to capture Work Type Distribution

<table>
<thead>
<tr>
<th></th>
<th>Next</th>
<th>Design</th>
<th>Build</th>
<th>Feedback</th>
<th>Deliver</th>
<th>Verify</th>
<th>Value</th>
<th>Delivered</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
<td>🟦</td>
</tr>
<tr>
<td><strong>Defects</strong></td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
<td>🟤</td>
</tr>
<tr>
<td><strong>Risks</strong></td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
<td>🟣</td>
</tr>
<tr>
<td><strong>Debts</strong></td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
<td>🟢</td>
</tr>
</tbody>
</table>

ddegrandis.com  @dominicad
Three ways to begin capturing and using DevOps metrics

1. Safe to fail experiments
2. Make work visible
3. Automatically capture data with tools
1. Safe to fail experiments

A complex system has no repeating relationships between cause and effect. When dealing with complex systems there is the need for experimentation.

Dave Snowden: http://cognitive-edge.com/methods/safe-to-fail-probes/
2. Make work & metrics visible
3. Automate – let your workflow mgmt tools automatically capture flow data.
A metrics learning experiment

1 metric trend in 4 areas:
- Speed
- Productivity
- Quality
- Predictability

See impacts of change in 1 metric by showing all 4 metrics

Inspired by Troy Magennis & Larry Maccherone, “Doing Team Metrics Right,”
http://focusedobjective.com/team-metrics-right/
Unplanned work delays
Planned work

Influence others using the power of visualization

1/4 How fast? Flow Time

Unplanned work delays
Planned work

Influence others using the power of visualization
What we’re measuring impacts people b/c people value what is measured.
### 3/4 How good? Quality

<table>
<thead>
<tr>
<th>Change Failure Rate</th>
<th># FD done items</th>
<th># of total done items</th>
</tr>
</thead>
<tbody>
<tr>
<td>⅓ = 25%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>⅔ = 28.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>⅔ = 33.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>⅔ = 0%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Oh - ok – I see what you mean!!!**
When people complain that things take too long, measure actuals. It’s useful to test opinions against data.

Percentiles answers Q: “What’s the probability of completing work in x days?”

90th percentile filtered on business requests
Three Takeaways

1. Capture & present metrics to help others see the problems & risks in order to provoke necessary conversations for change.

2. Implement change using experiments and a humble approach to get the buy-in you need for change.

3. Shift left – visualize upstream work along with your work to see the value stream to optimize the whole vs. individual teams/siloes.
Email: dominica@SendYourSlides.com
Subject: flow
To receive:

- copy of this presentation deck
- excerpts of Making Work Visible
- Tasktop video on TFS/SN tool integration
- Forrester article: Agile-Plus-DevOps With Value Stream Management