Testers in the deployment pipeline block or assets?

Fitting in software testers when you release multiple times per day

› Alan Parkinson  › CEO/CTO  › @alan_parkinson
My fellow engineers, please stop asking “Is this process good or bad?” and start asking “Is it well-suited to my situation?”

**Jocelyn Goldfein.**  MD at Zetta Venture Partners
Former engineering exec at Facebook and VMWare
High-performers were twice as likely to exceed organizational performance goals as low performers.
People use their prejudices, beliefs and worries to infer what the lack of information about testing in DevOps means.
It’s interesting to note that having automated tests primarily created and maintained either by QA or an outsourced party is not correlated with IT performance.
<table>
<thead>
<tr>
<th>Manual work</th>
<th>High Performers</th>
<th>Medium Performers</th>
<th>Low Performers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration management</td>
<td>28%</td>
<td>47%*</td>
<td>46%*</td>
</tr>
<tr>
<td>Testing</td>
<td>35%</td>
<td>51%*</td>
<td>49%*</td>
</tr>
<tr>
<td>Deployments</td>
<td>26%</td>
<td>47%</td>
<td>43%</td>
</tr>
<tr>
<td>Change approval process</td>
<td>48%</td>
<td>67%</td>
<td>59%</td>
</tr>
</tbody>
</table>

What does your pipeline look like?
A hypothetical Continuous Delivery pipeline

- Compile
- Small tests
- Medium tests
- Large tests
- Exploratory testing
- Deploy to production
What does your pipeline look like?
My Continuous Delivery pipeline

Compile → Small tests → Docker Container → Medium tests → Deploy to production

Lint

6 minutes 23 seconds
Measuring software delivery performance

**Lead time for change**
The lead time of code commit to deployment in production

**Deployment frequency**
How often is work deployed to production? This is proxy of batch size within Lean

**Mean time to recovery (MTTR)**
How quickly a team recovers from failures in production: Bugs, bad deployments, outages
Measuring software delivery performance

**Lead time for change**

The lead time of code commit to deployment in production

**Deployment frequency**

How often is work deployed to production? This is proxy of batch size within Lean

**Mean time to recovery (MTTR)**

How quickly a team recovers from failures in production: Bugs, bad deployments, outages
Measuring software delivery performance

**Lead time for change**
The lead time of code commit to deployment in production

**Deployment frequency**
How often is work deployed to production? This is proxy of batch size within Lean

**Mean time to recovery (MTTR)**
How quickly a team recovers from failures in production: Bugs, bad deployments, outages
<table>
<thead>
<tr>
<th>Year</th>
<th>High Performers</th>
<th>Medium Performers</th>
<th>Low Performers</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016</td>
<td>Deployment Frequency</td>
<td>On demand (multiple times per day)</td>
<td>Between once per week and once per month</td>
</tr>
<tr>
<td></td>
<td>Lead Time for Change</td>
<td>Less than 1 hour</td>
<td>Between 1 week and 1 month</td>
</tr>
<tr>
<td></td>
<td>MTTR</td>
<td>Less than 1 hour</td>
<td>Less than a day</td>
</tr>
<tr>
<td></td>
<td>Change Failure Rate</td>
<td>0-15%</td>
<td>31%-45%</td>
</tr>
</tbody>
</table>

* Table 2.2 from *Accelerate: The Science of Lean Software and DevOps.*
our system of make-and-inspect, which if applied to making toast would be expressed: “You burn, I’ll scrape.”

W. Edwards Deming

Quality comes to City Hall
Overconfidence in case-study judgments

http://sethrylan.org/bayesian/citations/Oskamp-Overconfidence_in_Case_Study_Judgements.pdf
“The odds of a meltdown are one in 10,000 years”

Vitali Skylarov. Minister of Power and Electrification in the Ukraine two months before the Chernobyl accident (cited in Rylsky, 1986, February)
Feature branching

STORY-121
MASTER
STORY-122
STORY-119
“people in this country have had enough of experts”

Michael Gove  Justice secretary, Her Majesty’s Government
Let’s play spot the difference
Spot the difference

HindsightSoftware.com/SpotTheDifference
Feature branching
Rebasing
Rebasing
Racing to merge
Racing to merge
Feature Branches are intuitive but harmful
We’ve used feature branches, and Pull Requests for code review and testing for 5 years. I was wrong, this is a faster way.
Trunk based development
Feature toggles
Feature Flags, Feature Bits, or Feature Flippers
Toggle implementation options

**Build time toggles**

Embedded in code as a simple Boolean value or compiler directives

**Boot / Configuration toggles**

Controlled by environment variables or configuration files. Any change require a restart

**Real-time toggles**

Stored in the live database or controlled using service discovery. No need to reboot
Toggle implementation options

Build time toggles
Embedded in code as a simple Boolean value or compiler directives

Boot / Configuration toggles
Controlled by environment variables or configuration files. Any change require a restart

Real-time toggles
Stored in the live database or controlled using service discovery. No need to reboot
Toggle implementation options

**Build time toggles**
Embedded in code as a simple Boolean value or compiler directives

**Boot / Configuration toggles**
Controlled by environment variables or configuration files. Any change require a restart

**Real-time toggles**
Stored in the live database or controlled using service discovery. No need to reboot
Segmenting with toggles

**Per user**

Embedded in code as a simple Boolean value or compiler directives

**Per tenant**

Controlled by environment variables or configuration files. Any change require a restart

**Per deployment group / region**

Stored in the live database or controlled using service discovery. No need to reboot
Segmenting with toggles

**Per user**
Embedded in code as a simple Boolean value or compiler directives

**Per tenant**
Controlled by environment variables or configuration files. Any change require a restart

**Per deployment group / region**
Stored in the live database or controlled using service discovery. No need to reboot
Segmenting with toggles

**Per user**
Embedded in code as a simple Boolean value or compiler directives

**Per tenant**
Controlled by environment variables or configuration files. Any change require a restart

**Per deployment group / region**
Stored in the live database or controlled using service discovery. No need to reboot
Remove activities from the critical path
Remove activities from the critical path

- Compile
- Small tests
- Medium tests
- Large tests
- Deploy to production
- Exploratory testing
Developers need to proactively reach out to testers to pair on risks when identified.
Monitoring, your safety net
Production monitoring can create an opportunity for exploratory testing.
What would happen if a Monkey got into your data center?

https://github.com/Netflix/SimianArmy
Functional tests (large)
End-to-end tests (large)
Story tests (large)
Examples/BDD
Prototypes

Exploratory testing
Usability testing
Accessibility
Acceptance testing
Alpha/Beta

Unit tests (small)
Components (medium)

Performance & load
Security

Lisa Crispin and Janet Gregory Agile Testing and More Agile Testing
Should it be in the pipeline or production?
I can’t do CD...

Applying these concepts in Scrum or when you don’t own the production system
Trunk based development in Scrum
https://danashby.co.uk/2016/10/19/continuous-testing-in-devops/
Small commits through the pipeline

- Compile
- Small tests
- Medium tests
- Large tests
- Deploy to production
- Exploratory testing
Testers in the deployment pipeline blockers or assets?

Don’t forget to submit feedback if you enjoyed the session
The problem with Pull Requests

Late integration of code
Subsequent commits depend on the first, late feedback for issues causes a cascade of rework.

Rework from merge conflicts
Discourages refactoring code through large batch size and conflicts.

Rework from re-testing
Performing testing and then discovering someone else has merged and have to rebase.