# Rotating DevOps role improves engineering service quality

As many high-performing agile software engineering teams embrace a DevOps culture, they're adding the role of Directly Responsible Individual (DRI). The role is also known by various other names, such as Google's "Sheriff" or Facebook's slightly different "Designated Response Individual." Rotating within an agile team, the DRI is responsible for service availability, service health, and incident management. The DRI advocates for the customer and drives positive changes to improve the customer experience with services.

In Microsoft IT, we're using a DRI to help us deliver better services faster and more cost effectively. The DRI actively looks at services in production, thereby helping our agile teams be proactive rather than reactive. This has helped us reduce—by up to 50 percent—the number of support tickets and bugs that we have to resolve. With the rest of the team free of this distraction, they have more time to deliver business value.

We used to only get four to five hours per day of productive work out of each software engineer. Since adding this this role to our teams, productive time has increased to six hours per day. This role also reduces risk because resolving issues doesn't interfere with our ability to deliver on a sprint. In addition, we're finding that the DRI reduces the number of engagements we have with support, so these costs also are going down.

## DRI process and expectations

In Microsoft IT, we have a primary DRI with a secondary DRI as a backup. The primary DRI is 100 percent allocated to this role and has no other team tasks. Each day, the primary DRI reviews incident logs, responds to critical incidents or patterns of incidents. They also log defects, and assign them to individuals based on root cause analysis. For visibility, the secondary DRI is looped into any issues. In the event the primary DRI is unavailable or busy, the secondary DRI steps in.

### DRI role rotation

The primary and secondary DRI role rotates across all team members. For a seamless transition, the secondary DRI becomes the primary DRI at the next rotation. The primary and secondary DRI don't overlap the Scrum Master role during the same sprint.

The rotation cadence is two weeks, which aligns with the ideal two-week sprint cadence. This ensures that the DRI can participate in service reviews and other service-line meetings that are held every other week. It also ensures that the DRI has ample impact during the sprint and the opportunity to spend time in preferred engineering activities. Rotations start on the first day of the sprint and last until the first day of the next sprint. It's up to the sprint team to track and manage their DRI schedule.

### Sprint capacity

DRI activities require effort, and effort doesn't come free. Effort correlates to capacity, and existing engineering efforts need to change or stop to free up this capacity. For this reason, the primary DRI is not accounted for in the current sprint capacity. We schedule the primary DRI time as "days off" in Visual Studio Team Services (VSTS). This keeps DRI work from having an impact on the sprint plan. In the event the secondary DRI becomes heavily engaged, we have to re-plan the sprint accordingly.

## Incident management

The DRI responds to incidents in two ways:

- **Pull.** During core hours, the DRI reviews incidents for critical issues or patterns of issues that require resolution.
- **Push.** Outside of core hours, the service engineering team engages the DRI when software engineering assistance is required to respond to an incident. The on-call schedule for the primary DRI is rotated so that no one has to be on call for more than one major holiday per year.

In both cases, the DRI isn't solely responsible for fixing the issue. The DRI creates a VSTS work item and links it to the incident when possible. We prefer to track the work in a single system, while ensuring the effort (time) is tracked in VSTS.

The DRI performs root cause analysis and engages the software engineer who's accountable for the feature area or component. The DRI isn't expected to be the hero and fix all issues; however, if the issue is easily fixed the DRI may take the fix forward independently while following up with the extended team for visibility.

## High-severity issues

When handling a high-severity live site production issue, the primary DRI should involve the secondary DRI, unless the primary DRI is confident that the issue can be resolved quickly. The DRI is also empowered to contact other team members who have knowledge that could be helpful. Reaching out to others, even if they're not on call, is the right thing to do. Multiple people working on critical issues can decrease the time to resolution and reduce the stress for the DRI, who would otherwise handle the issue alone. It also helps team members grow in understanding.

# Less need for supporting teams

As we mature the DRI role in our agile teams, we expect to reduce—and eventually eliminate—the need for supporting teams. This will free up capacity for creating more business value and quality within our agile teams.

## Sustaining engineering

The cheapest way to fix a bug is to catch it when it's introduced and have the individual who introduced the bug fix it. When the sustaining engineering team resolves defects that we introduce, it creates a culture of reduced accountability and deferred quality. Releasing the sustaining engineering team frees up capacity and changes our team mindset to rapidly fix forward.

## Release management

Today we depend on a virtual team of release managers to deploy our software to production. Handoff from the agile team to this team results in a loss of context and requires a dedicated effort for knowledge transfer. Going forward, the primary DRI will take responsibility for deployment to production. The DRI will ensure there's proper deployment documentation, automation, and validation. After deployment, they'll review the results and service state. This practice will also reduce access to potentially sensitive information from a broad team to a single individual, which is a pattern that's in alignment with Sarbanes-Oxley (SOX) compliance.

# Key results

Since adopting the DRI role in our agile teams, we've experienced many benefits, including improved service quality and customer experiences, career growth for team members, and greater readiness for DevOps within our teams.

## Service quality

With a DRI proactively investigating internal exceptions and ticket trends, our teams have been resolving bugs during each sprint. This has improved our customers' experiences and reduced exception and ticket trends week over week. The following screenshots show ticket trends for our payee management team, which has a rotating DRI role. A recent period showed a 50 percent reduction. Year over year, we had a 30 percent reduction in tickets.

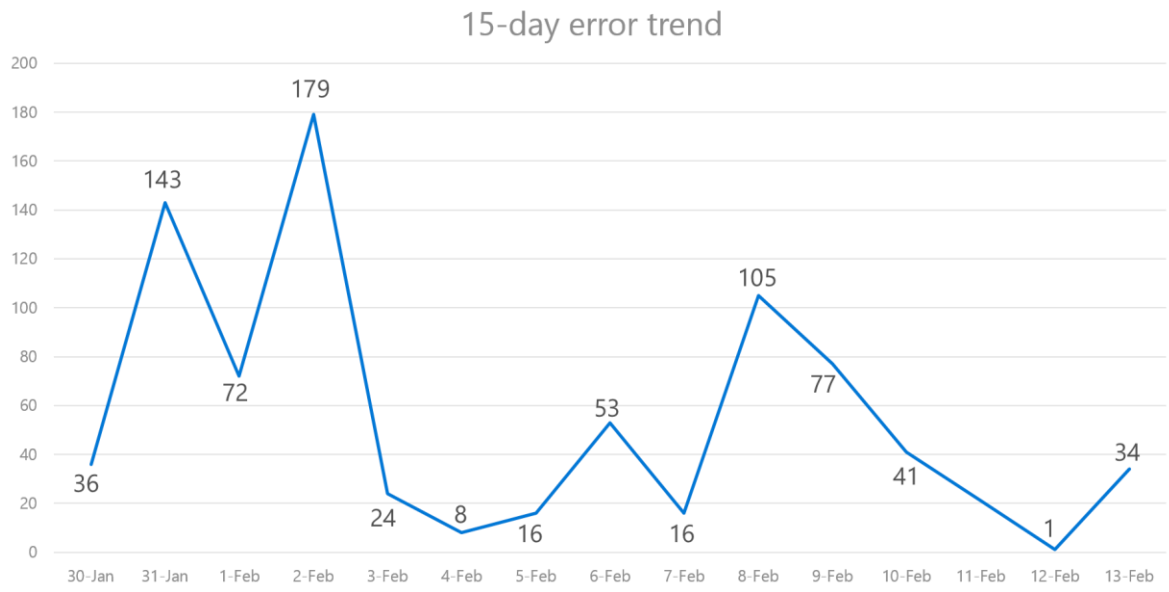


*Figure 1. Ticket trends over 15 days: over 50 percent reduction*

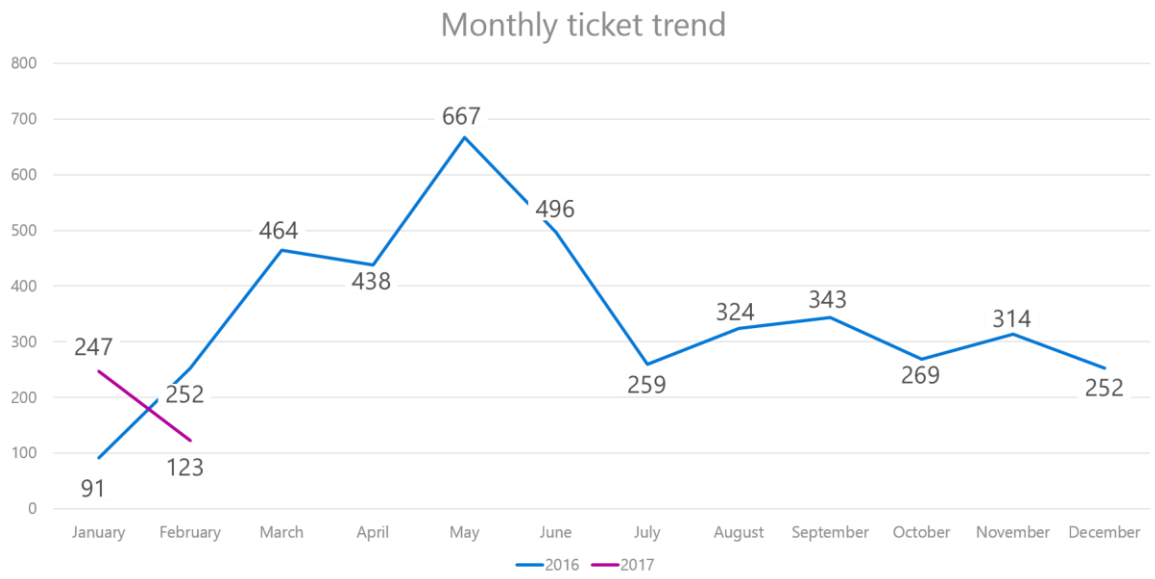


*Figure 2. Ticket trends over a year: over 30 percent reduction with a steady decline*

Addressing defects broadly across the team has put greater focus on quality and our bug backlog. Payee management is now experiencing a shallow bug backlog (less than 30 new/active bugs).

## Improved customer experiences

When team members participate as a DRI, they gain knowledge about the end-to-end service. The DRI is responsible for understanding the service in full, the customer experience, and how the service is enabling business outcomes. This increased broad focus makes team members more accountable to deliver high-quality customer experiences and is driving richer designs.

Each DRI brings a unique lens and different values to the role. This diversity in focus is helping the team improve the service in many areas. For example, one of our DRIs discovered that a service wasn't running in the same region as our data store. This pattern didn't exist in pre-production and may not have been noticed without the DRI function. The team now has a backlog item to redeploy the service to the same region as the customer, which will reduce latency and improve the customer experience.

Previously, when our customers encountered defects, they would retry their task or use known workarounds. Today, the DRI proactively identifies blocking issues and fixes defects before the customer escalates them. In some cases, the DRI logs support tickets before the customer is even aware that an issue exists. This is dramatically improving our mean time to detect (MTTD) and mean time to resolve (MTTR) metrics.

## Career growth and autonomy

Our software engineers find that working within the DRI role is a rewarding experience. They're developing new skills and forming new patterns of working that are increasing their impact and relevance, with the following benefits:

- **Career growth.** Engineers are developing DevOps full-cycle software engineering skills and relevant industry experience.
- **Leadership skills.** Engineers are participating in service health with live site reviews and ongoing engagement opportunities and gaining leadership experience across organizations.
- **Engineering autonomy.** When assigned DRIs aren't performing DRI tasks, they can work on projects that they're most passionate about—such as training, bug fixing, vNext design, or other work the DRI is most passionate about.

## DevOps readiness

The DRI rotation is building DevOps basics in our teams: from telemetry analysis and instrumentation to deployment into production. After each team member has been the DRI for a few rotations, they're better suited for aggressive DevOps responsibilities and patterns of working.

# Testimonials

Here are some testimonials about the benefits of adding the DRI role to our agile teams.

## Service engineer testimonial

*I now have a greater partnership with software engineering where I think we are focused on more meaningful work that has greater impact. While the work is highly technical, we are seeing success as we apply focus to these challenges.*
—Kyle Kardong

## Business partner testimonial

*I strongly believe the list of production backlog items which are being proactively resolved is primarily due to the DRI role. While we are yet to exploit the role to the maximum possible extent, we have started seeing some tangible benefits due to this new role rollout. Business also started getting involved in prioritizing work items and would get maximum bang for the buck.*
—Binu Surendranath

## Product owner testimonial

*Introduction of DRI in Payee Management has put a renewed focus on the health of production and built a stronger sense of ownership across the team. The DRI has been able to proactively identify issues before support tickets get created and identify issues from error log analysis that might not have otherwise been identified. The speed at which the team can go from issue identification to fix available is much quicker than in the past and is pushing the right process changes with our business partners to ensure we can get fixes quicker to production.*
—Katie Lencioni

## Scrum Master testimonial

*DRI lifts the burden of managing existing features and configurations with the development currently taking place for future sprints. Having a dedicated person to monitor and resolve production issues is very impactful to the workload and responsibility associated with the Scrum Master role.*
—Ronald Klemz

# For more information

## Microsoft IT

microsoft.com/ITShowcase

Adopting modern engineering processes and tools at Microsoft IT

Creating a culture of modern engineering within Microsoft IT

Moving to modern engineering

Transforming IT for modern engineering