Shifting Other Than Left or Right

Jon Bach

jbtestpilot@gmail.com

Abstract

You've likely heard of "Shift-Left" -- the DevOps principle of moving testing closer to Development with programmatic approaches like Build/Code/Plan/Test pipelines.

The complementary DevOps dimension is "Shift-Right" -- moving testing closer to Production with programmatic approaches like Release/Deploy/Operate/Monitor.

In my work as a Program Manager who works with Developers, Testers, Business Intelligence, Legal, Operations, Compliance, Localization, and every adjacency you can think of, I've noticed a third and fourth dimension we don't talk enough about -- Shifting IN to the details and OUT to the big picture -- getting perspective so you know how best to marshal your Left and Right resources.

Biography

Jon's been in Tech since 1995, starting in Customer Support for a commercial real estate dial-up service. He's currently a Senior Program Manager Quality Manager and consultant for a small software development studio called ProphetTown. His longest role was at eBay where he worked for 13 years a Quality Director and Program Manager. He's also been a full-time employee at Microsoft during Y2K, and served on contracts at HP, Adobe, AT&T, WebMD, Getty Images, Alaska Airlines, McGraw-Hill, and more. He's the co-inventor (with his brother James) of Session-Based Test Management – a way to manage and measure Rapid Software Testing. Jon loves turning rhetoric and philosophy about software development into action (baking half-baked ideas), and frequently posts on LinkedIn about software quality and program management.

Introduction

I frame this paper with a model of thinking about how to examine aspects of a software risks and problems from what I learned in over 13 years at eBay as a Quality Program Manager and customer advocate

If shifting left is being able to find problems earlier, and shifting right is about learning from Production, it's time there was a paper about how we can see not just the Left or the Right, but zoom In and OUT to see the significance of one tree or the beauty of the WHOLE forest.

The Problem

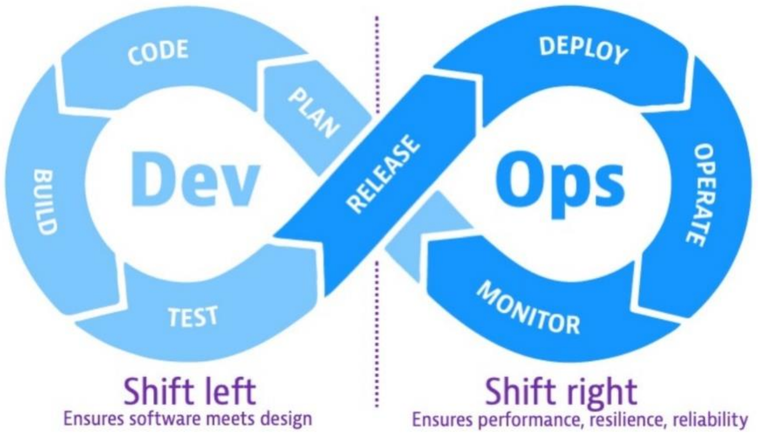

CODE
PLAN
BUILD
Dev
TEST
RELEASE
DEPLOY
Ops
OPERATE
MONITOR
Shift left
Ensures software meets design
Shift right
Ensures performance, resilience, reliability

Diagram courtesy [Dynatrace website](#) (article by Saif Gunja)

This is a classic representation that's a model of Development and Operations in terms of moving testing closer to Development with programmatic approaches

This seems like Waterfall Development with a twist – literally. This diagram keeps coming back to the MIDDLE – both shifts might be good, but their captions might benefit from some context and perspective.

"Shift Left" might really mean: "We do our best to get it right before it goes out"

"Shift Right" might really mean: "We do our best to learn from what happens in the field so that we can do better in the next iteration."

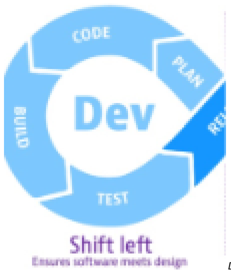
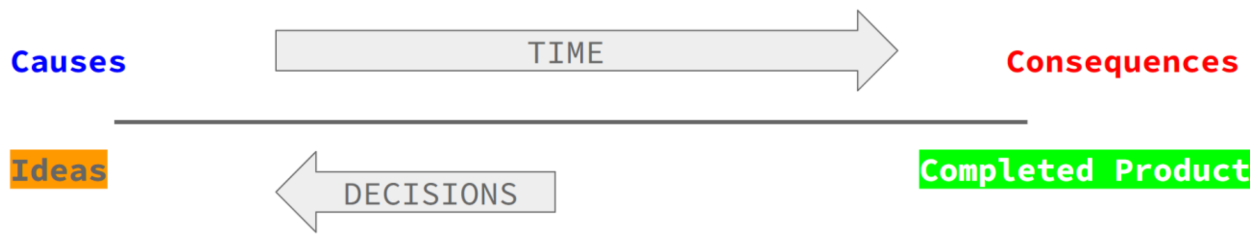


Diagram courtesy [Dynatrace website](#) (article by Saif Gunja)

LEFT is the DevOps principle of moving testing closer to Development with programmatic approaches like Build/Code/Plan/Test pipelines.

It's meant to be linear, with time going from left to right, from CAUSES to CONSEQUENCES, but there's also a spectrum between IDEATION and a COMPETED PRODUCT, from which things like decisions are meant to be moved closer to IDEATION to prevent the risk of costly rearchitecture and redesign.

The ultimate LEFT is looking at every line of code as it's built and seeing quick ways to prevent problems before they happen. This could mean unit tests and code reviews, advanced IDE syntax or pattern-checking features, continuous integration on every pull request, checkin, and merge, or recent claims that AI can find, assess, suggest, and create fixes to problems that haven't happened yet.



Now, let's take its complement – shifting RIGHT – the DevOps principle of moving testing closer to Production with programmatic approaches like Release/Deploy/Operate/Monitor.

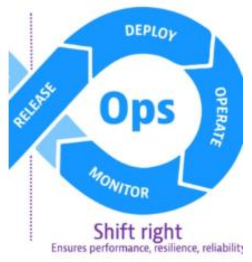
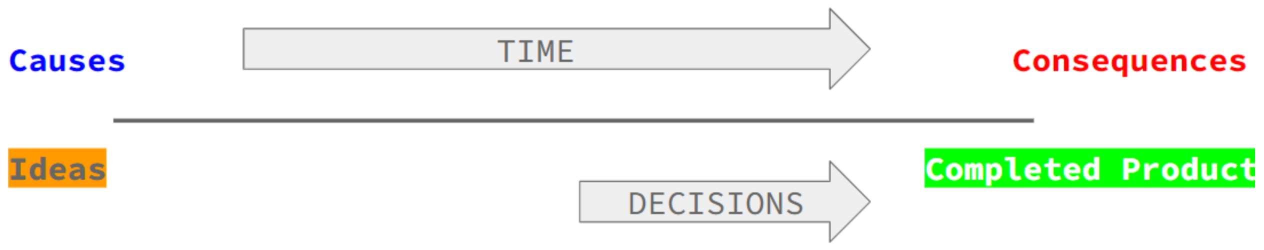


Diagram courtesy Dynatrace website (article by Saif Gunja)

It's also meant to be linear, with time still going from left to right, from CAUSES to CONSEQUENCES, also against the backdrop of IDEATION and a COMPETED PRODUCT, but where decisions are meant to be moved AWAY from Ideation because there's not enough data or context to make a call.

The ultimate shift RIGHT is Procrastination. Procrastination isn't bad. Delaying a decision or an action gives people a chance to learn, acquire better equipment and facilities, better tech, and less *guessing*. Working on a Minimum Viable Product or a 1.0 where you expect early adopters from your prototyping and incremental development is really shift RIGHT. You see how people like it first. Sure, if you could remove the cause of a bad event, that would be great, but there's a paradox: we want to go as far back as we can to make good things happen, but we don't want to make any decisions until we know how it all turns out! If you don't have a time machine, you will have to ITERATE to adjust your INVESTMENT.



Alaska Airlines Flight 1282 (January 5, 2024)

Let’s anchor these concepts in a recent example – Alaska Airlines Flight 1282.

There’s an excellent report up on the National Transportation Safety Board website (https://www.ntsb.gov/investigations/Pages/DCA24MA063.aspx)

“On January 5, 2024, about 1714 Pacific standard time, Alaska Airlines flight 1282, a Boeing 737-9, N704AL, returned to Portland International Airport (PDX), Portland, Oregon, after the left mid exit door (MED) plug departed the airplane leading to a rapid decompression. The airplane landed on runway 28L at PDX without further incident, and all occupants (2 flight crewmembers, 4 cabin crewmembers, and 171 passengers) deplaned at the gate. Seven passengers and one flight attendant received minor injuries.”

The report has the following elements:

- Crew Experience and History of Flight
- Recorders: Cockpit Voice Recorder and Flight Data Recorder
- Operator and Airplane Information
- Mid Exit Door Plug Description
- Cabin Pressurization/Cabin Description
- Materials and Structure Examination
- Manufacturing Records/Human Performance
- Safety Actions

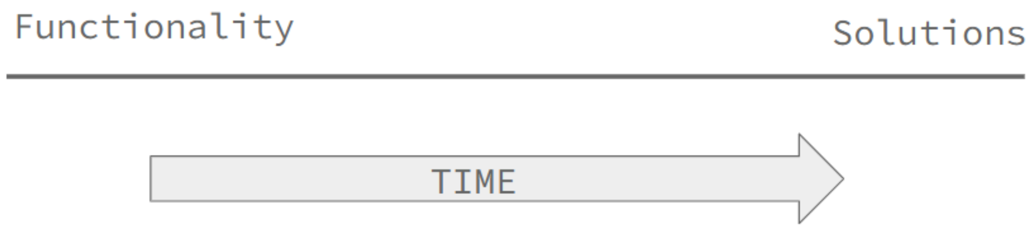
Working on the LEFT *(pushing decisions closer to ideas and causes)*

A lot of this is evidence of working toward the LEFT. Considerations like the relationship to the assembly line, time pressure, psychological safety, having information to do the right thing; knowing there's a problem and being able to solve it and WANTING to solve it.

Working on the RIGHT *(learning from the consequences of incidents in the completed product)*

In other words, are we as software professionals on the lookout for signs and symptoms of problems in the field? (For example, the NTSB looked at maintenance logs indicating a pressure controller light had illuminated on three previous flights.)

In software development, shifting right could mean going from creating FUNCTIONS to verifying SOLUTIONS. You can’t have functionality without thinking “Does anyone NEED this?”



IN vs OUT

Now let’s look at two other important dimensions that few people talk about as a representation of something other than LEFT or RIGHT.

In versus Out is akin to using a microscope or a telescope.

One context-centering question we could ask is:

- Who's looking? What problem are they trying to solve?

For the NTSB investigator, it’s these:

- What kinds of things have to go into my report?
- What are the politics and legal implications?
- What are the forces that are guiding the report?
- What's relevant and what’s irrelevant?

Shifting IN (using a microscope) is akin to the following:

- Focusing on the fine details
- Making a “Situation Room”
- Writing the “play-by-play” on how it happened
- Seeing what the “bolts” (literally, in this case) look like

It’s also taking time to understand how the work happened:

- How do people read checklists?
- What are (or were) they told?
- Are they given too much work?
- What are they told vs. what do they actually do?
- Understanding the staff’s daily experience

Shifting OUT (using a telescope) is akin to the following:

- Addressing general policy changes that affect people's behaviors,

You can go out in different directions, for example -- toward Strategy or

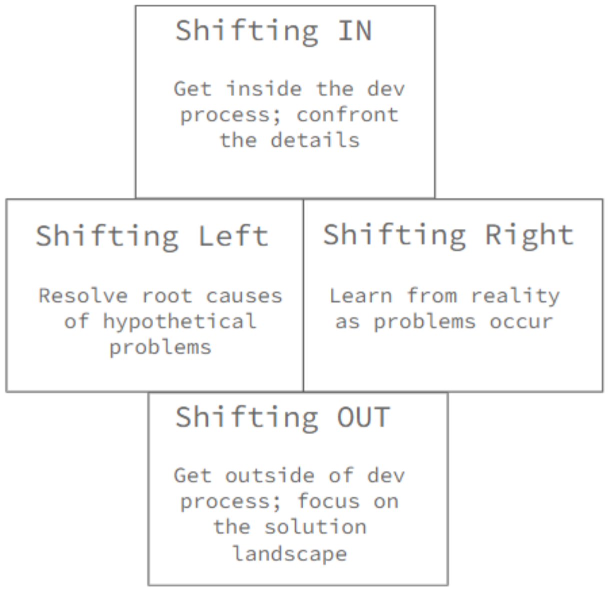
toward Big Picture, which includes timelines, system components, etc.

Shifting out could also be about delegation or outsourcing -- letting go of the problem so you specifically and purposefully become an outsider to get critical distance. You might deliberately find value in shifting your involvement.

Applications of these dimensions

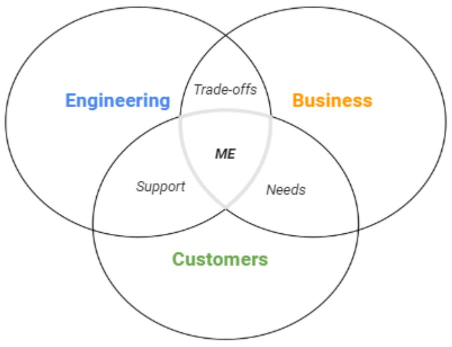
Shifting In or OUT is a complementary dimension to LEFT and RIGHT.

You can look at the dimensional focus like the movements of a joystick.

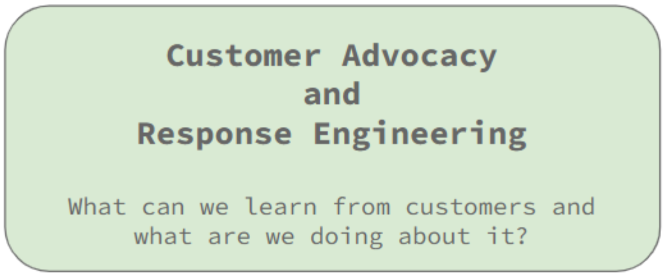

Shifting IN
Get inside the dev process; confront the details
Shifting Left
Resolve root causes of hypothetical problems
Shifting Right
Learn from reality as problems occur
Shifting OUT
Get outside of dev process; focus on the solution landscape

When I shifted careers from a QA Manager to a Program Manager after about 20 years, my perspective went from paying attention to one project to many. I went from finding BUGS in a PRODUCT to finding RISKS in multiple PROJECTS that were needed to comprise an enterprise PROGRAM. A program is a collection of deliverables from different teams who need a coordinator of their individual solutions to make one big solution -- much like individual features comprise a whole product.

Let's look at three major overlapping domains in Software Development:



When I was at eBay working on the Customer Survey platform, I needed to deal with many teams to either get the big picture of patterns of failure that were reported, or drill into specific bugs customers were reporting. I shifted Left, RIGHT, IN and OUT and it occurred to me I was in a new role that I'd never heard of in 30 years of software development, so I came up with a name for it:



Customer Advocacy
and
Response Engineering

What can we learn from customers and what are we doing about it?

What I think I have here in this set of perspectives is a ***critical thinking tool on how to approach a project.***

Between **Engineering and Business**, I worked to consider and coordinate **Tradeoffs –** for example, Engineering can only do X, even though the business wants X, Y, and Z.

Between **Business and Customers**, I worked to collect and assess **Needs –** for example, Business needed more items on the site, and Customers wanted an easier time to list them.

Between **Customers and Engineering**, I worked to see what **Support** needed to look like **–** for example, Engineering needed more specificity about Production problems to get to the root cause more quickly, which worked well to support customers in rapid and clearly communicated bug fixes.

If we anchor these three sets of relationship back to something like an NTSB report, we see all 4 ways of coping with a problem:

1) IN: You get all the details of the problem and steep yourself in it. You become the expert, the ultimate insider, to get really close to it.

2) OUT: Think about its context and the dynamics that underlie it. Think about large flows of energy and flows through the system, as well as the players involved.

3) LEFT: Look at it in terms of its beginnings. Where did this problem come from? What was it like right when the prob started?

4) RIGHT: What did the problem eventually become? What did it manifest as?

At any time on any project we have to choose a frame of reference. We need to decide what's the object we're studying? For example, there's a perspective that a person in charge of QA at Boeing has AFTER the January event happened. They know what they know from this report and then they ask "Now what?" Is that person trying to go left / right / out / in, or is it the NTSB the people who are trying to write a report and it's the person in CHARGE of that team who needs to go left / right / out / in.

When I was a Director of Program Management at eBay on the customer survey platform team, my point-of-view was usually like the senior leaders at Boeing who read the NTSB report

But even then, there are opportunities to pivot – or, like a camera, to pan or zoom.

1) Pan LEFT -- how was this pattern of bugs created? Was there a recent release which had an infrastructure hiccup during the rollout? What is the relationship of a team rolling to Production and the Operations personnel? Was there time pressure? When someone did the wrong thing to cause a rollout problem, did they not know the process, or did they know but were overwhelmed.
2) Pan RIGHT -- if aspects of the release were screwed up, how could we know? Are we on the lookout (monitoring and alerting, for example) for signs and symptoms of stuff going wrong in the field? Do we hear about every problem, or do we fix things in silos one at a time and don't see patterns, perhaps even in other similar releases that day?

3) Zoom IN – it’s all about the details in exactly how things happened. It’s the play-by-play, the timelines of events. It’s about seeing the code or the ramp parameters, the server stats, understanding procedure and how people read and follow it. It’s about what they are told or the fact that they might have been given too much work; what they were told to do vs. what they actually did. It’s about understanding the daily experiences of the people involved.
4) Zoom OUT – this could be about corporate policy or changes that affect people's behaviors. You can go out in different directions -- toward strategy or the big picture; toward large-scale timelines, or the components of systems and how they interoperate. Shifting OUT is also about the meat-cognitive plan of what kinds of things have to go into an incident report. Maybe there are politics or legal implications. What are the forces that are guiding the report? What's irrelevant and relevant? What absolutely HAS to go not the report vs. what would I like to go in the report? Maybe it even involves coordinating or considering a whole panel of experts to qualify the report.

Summary

This year’s PNSQC theme (“The Future is Now”), could mean that this Customer Advocacy and Response Engineering role I served at eBay is the future of how Quality gets assessed – an appropriate combination and teamwork of Testing perspective and Program Management perspective, aided by not only appropriate shifts either to the LEFT or the RIGHT, but knowing also when to use a telescope or microscope to shift IN or OUT.

References

Tech Beacon. “Why software development needs shift not left or right” February 2019. Michael Giacometti. https://techbeacon.com/app-dev-testing/why-software-development-needs-shift-not-left-or-right

National Transportation Safety Board. “Alaska Airlines Flight 1282 Investigative Hearing” https://www.ntsb.gov/news/events/Pages/Alaska-Airlines-1282-Inv-Hearing.aspx

National Transportation Safety Board. “In-Flight Mid Exit Door Plug Separation”https://www.ntsb.gov/investigations/Pages/DCA24MA063.aspx

Dynatrace. “Shift left vs shift right: A DevOps mystery solved.” May 2024. Saif Gunja https://www.dynatrace.com/news/blog/what-is-shift-left-and-what-is-shift-right/

Dynatrace. “What is DevOps? Unpacking the purpose and importance of an IT cultural revolution” February 2023. Saif Gunja https://www.dynatrace.com/news/blog/what-is-devops/

Also credit to James Bach for deep conversations and for helping me develop out the parameters of the idea. (Satisfice, Inc.: April 2024)