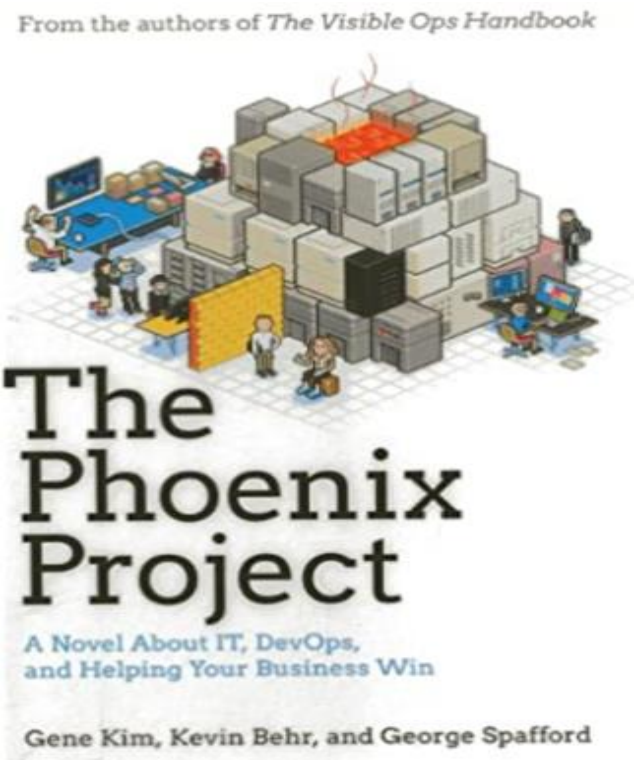

The Phoenix Project: Discover how DevOps can increase your company's value.

Parts Unlimited, an automotive manufacturing and retail company, has a new IT initiative. It's called "The Phoenix Project," an d it's supposed to seamlessly integrate the company's retail and e-commerce channels; this will give Parts a competitive edge and increase its customer base.

Phoenix is critical to the company's success. It's also *very* behind schedule – and over budget. So the CEO is looking to put someone in charge of fixing the mess. They'll have 90 days to turn things around, and if they fail, the entire IT department will be out of a job.

While the story that follows is fictional, the problems and solutions it illustrates are very real – as anyone in IT will recognize. These blinks outline how a core, ongoing conflict between Development and IT Operations spells failure for not only the IT department, but a company as a whole. And they reveal how a set of practices called DevOps can lead to incredible efficiency, cutting-edge performance, increased value – and a lot less headache.

In these blinks, you'll discover

- what IT has to do with a manufacturing plant;
- the four types of IT work; and
- why repeated failure is the recipe for success.

Chaos in the IT department spells failure for the entire company.

It's Tuesday morning, and Bill is running late for work. He's speeding down the highway when he gets the call – he's supposed to come see the CEO, Steve, as soon as he arrives.

Uh-oh, he thinks. *I'm going to get fired*. It wouldn't be a huge surprise. As head of a midrange technology group at Parts Unlimited, Bill has always taken pride in doing his job well. He's built a reputation for being reliable, efficient, and candid; he was a former Marine, after all. But Parts has been struggling, big-time.

Bill looks around the near-empty parking lot he's pulling into. It seems like its competitors are constantly innovating, and Parts is being left in a trail of dust. With all the layoffs in the past couple years, Bill's department has increasingly had to do more with fewer resources.

But Steve isn't firing him. He's giving Bill a "promotion." As the new VP of IT Operations, Bill will report directly to Steve and ensure that the impending Phoenix Project rollout is a success. The company's future depends on Phoenix, Steve says – Bill knows that, right? Customers need to be able to shop from Parts both online and in brick-and-mortar stores. Otherwise, soon there won't *be* any customers: the company will no longer exist.

The key message here is: Chaos in the IT department spells failure for the entire company.

Bill doesn't want the promotion – he'd effectively be agreeing to take on the role of janitor, in charge of cleaning up a colossal IT plumbing mess. So he's horrified to find himself shaking Steve's hand in agreement.

Where's he supposed to start? The entire IT infrastructure is in complete and utter disarray.

John from the Information Security department is continually creating SEV1 outages – critical incidents that everyone has to scramble to fix – because he bypasses the proper clearance procedures to push through Development changes *he* deems important for auditors. Of course, Operations isn't ever able to test these changes first; there's no test environment because there's no budget.

Meanwhile, Patty, the director of IT Service Support, informs Bill that changes aren't ever documented; no one wants to spend time using the company's clunky change management tools. And no one attends the weekly Change Advisory Board, or CAB, meetings.

How does anyone keep track of anything that's going on? The answer, Bill realizes, is: *they don't*. No wonder things have gone to shit.

Bill looks at his desk. His old laptop was displaying the blue screen of death, so he's just received a replacement. It's at least ten years old – a bulky, heavy dinosaur of a machine. Half the lettering has worn off, and the battery is taped on.

He can't help but wonder if the universe is telling him something.

IT Operations, like Plant Operations, is governed by the theory of constraints.

Phoenix is supposed to deploy in ten days – but only three of the project's 12 final tasks are done, because the IT department has been handling emergency issues.

More accurately, Brent, its lead engineer, has been handling them. *Poor Brent*. Bill doesn't understand how one guy seems to be bolstering the entire organization. He's a wunderkind at solving any problem he's hit with. And so he's hit with them all.

The key message here is: IT Operations, like Plant Operations, is governed by the theory of constraints.

Just as Bill is diving into his new responsibilities, he's told he has to meet with Erik – a prospective board member who's apparently some technology bigwig. But when he gets to the conference room, the only person there is a donut delivery guy in wrinkled pants and a denim shirt. *Huh,* Bill thinks, as he starts stacking the donuts high on his plate. *Didn't know they delivered.*

Suddenly, the deliveryman turns around, extends his hand, and says, "I'm Erik." And then he immediately launches into telling Bill everything that's wrong with IT Operations at Parts. There's this myth that IT is pure knowledge work, Erik says. People think it's immune to standardization and documentation. But IT could learn a lot from Plant Operations. To prove his point, Erik tells Bill to grab his stuff – they're going for a ride.

Five minutes later, they're pulling up to one of Parts' manufacturing plants. Inside, as they look out over the plant floor, Erik introduces the *theory of constraints:* In most factories, there are just a few resources – materials, machines, or people – that dictate the whole system's output. These are the bottlenecks, or *constraints*.

To maximize efficiency, first *identify the constraint*. If you don't know where your constraint is, you don't know where to focus improvements. And any improvement not made at the constraint is useless. Picture it along a line: If an improvement is made after the constraint, it means more waiting. If it's before the bottleneck, that just results in inventory buildup.

Step two is to *exploit the constraint*. Your constraint shouldn't ever be idle or waiting on any other resource – and it should always be tackling the highest priority commitment.

Finally, *subordinate the constraint*. The slowest Boy Scout dictates the entire troop's marching pace – so move that Scout to the front of the line. In other words, release all work based on how quickly the constraint can manage it.

Brent, Bill realizes – the engineer who ends up solving *everyone's* problems. Of course! He's their constraint. That's why simple 30-minute changes have been taking Brent weeks to finish. He's always utilized at or above 100 percent, so any work given to him just sits in the queue unless there's an emergency.

It all seems so logical. Bill needs to figure out how to match the tempo of work to Brent. Who'd have thought that IT work would be like running a factory?

Understanding the four types of IT work will help you meet your commitments.

It's a few days before Phoenix's deployment, and the project management meeting isn't going well. Despite Bill's request for more time, Sarah from Retail Operations is pushing for the launch to go ahead as planned. When she accuses the IT Operations team of dragging its feet, Bill takes a deep breath.

He's seen this movie before. A project's shipment can't be delayed because of promises made to customers or Wall Street. The developers usurp the schedule, leaving no time for proper operations testing. As a result, absurd shortcuts are made – and the final software product is often unstable or even unusable. Who has to pull all-nighters, continuously rebooting servers and adhering metaphorical Band-Aids to offset shoddy code? IT Operations.

Bill sighs. He's been mulling over something Erik mentioned: *the four types of work.* Rigor and discipline will only get you so far, Erik said. Until Bill knew what IT Operations "work" really was, he wasn't going to be able to successfully tackle project deliverables, outages, and compliance.

The key message here is: Understanding the four types of IT work will help you meet your commitments.

Getting Phoenix deployed was work, Bill had told Erik. But Erik wasn't impressed. Official company initiatives, or *business projects,* were just *one* category of work, he'd said.

So Bill is now racking his brain. What could the other categories be? Suddenly, it comes to him.

The second type of work is *internal IT projects*. These are infrastructure or IT Operations projects that business projects create; they're also improvement initiatives like automating deployments or making new environments. Internal projects are often not centrally tracked, which makes their workflow tricky to control.

Changes, the third type of work, usually result from business and internal projects; they're any activity that could impact the services being delivered. Because changes in IT Operations and Development are tracked in separate systems, mistakes and wasted time are common – especially when it comes to handoffs.

Last, but definitely not least, there's *unplanned work.* Unplanned work is like firefighting. It's taking care of all the incidents or emergencies caused by the three other kinds of work. And it *always* gets in the way of planned work commitments – it's what is compromising the Phoenix launch.

Unplanned work blocks you from achieving your goals. In that sense, it's "anti-work" – so do whatever it takes to get rid of it. Life's not perfect. Things go wrong, and unplanned work will always pop up. The key is being able to handle it in an efficient manner; that starts with pinpointing where it's coming from.

Often, it results from *technical debt*, which you incur by taking convenient, but ultimately foolish, short-cuts. Like with financial debt, technical debt has compound interest that grows over time. If you don't pay down the debt, all your energy is spent paying interest – in the form of unplanned work.

Great teams highlight trust, communication, commitment, accountability, and a focus on joint success.

Bill tries to implement Erik's lessons. But Steve, the CEO, intervenes; he insists on doing things the old way, according to the impossible timeline. As expected, the Phoenix launch fails spectacularly. And Bill doesn't bat an eye.

Instead, he quits.

Without him, the project takes an even steeper nosedive. The different departments – IT Operations, Development, the business – all frantically try to create order out of chaos. But they're constantly at war with each other, tossing blame around like a hot potato.

They need Bill. So with a little cajoling, and an enticing bonus, he returns.

The key message here is: Great teams highlight trust, communication, commitment, accountability, and a focus on joint success.

When Bill walks into the conference room, Steve is talking about his childhood. His family was very poor, he says. He used to spend summers picking cotton with his brothers, and he was the first one to ever go to college.

Ugh, Bill thinks. *I hate this touchy-feely stuff*. But they need to break this dog-chases-tail cycle, or it'll be over for Parts Unlimited. So all the department heads are sitting together to discuss Steve's favorite book, *The Five Dysfunctions of A Team.*

Trust, Steve says, forms the foundation for teamwork – which is, of course, needed to solve any complex business problem. An *absence of trust*, then, is the first dysfunction of a team. Leaders can get mired in conflict and chronic underperformance simply because there's no trust. And there's a trickle down effect: when leaders don't trust each other, neither do teams.

So how do you cultivate trust? Well, it starts with vulnerability, which is why Steve is leading a *personal history exercise*. Share where you come from and what drives you in order to model vulnerability. And then ask each person in the group to do the same.

Fear of conflict is the second dysfunction. Choosing artificial harmony over constructive, heated debate is an open road to nowhere. Candid discussions about contentious issues can be hard. They require leaders and teams to overcome ingrained behaviors – but embracing conflict can drive progress.

The third dysfunction is a *lack of commitment*. Especially when it comes to group decisions, fake buy-in introduces ambiguity and apathy into the company environment.

Avoiding accountability is the fourth dysfunction. Not taking responsibility and, conversely, not calling peers out for counteractive behavior both set low standards.

Finally, there's *inattention to results*. When you put your own personal gains, status, or ego before the success of the team, it's never a win-win situation.

The First Way involves the flow of work from Development to IT Operations to the customer.

The leadership meeting doesn't solve everything – but that's not the point. Bill sees that what the team has now is even better: a shared vision as they move toward a solution. It's time for everyone to work together. It's time, Erik says, for *DevOps*: a set of practices that integrates software development – "Dev" – and IT operations – "Ops."

Today, DevOps is doing for IT what the industrial revolution did for manufacturing. Instead of optimizing the process of turning raw materials into finished goods, DevOps shows how to upgrade the IT value stream – that is, speed up time to market, create more flexible and resilient systems, and integrate tests to quickly probe potential innovations.

The business principles behind DevOps work patterns mirror those in manufacturing. Erik calls them *The Three Ways*, and the first is all about workflow.

The key message here is: The First Way involves the flow of work from Development to IT Operations to the customer.

A critical part of the First Way is creating a fast flow of work through Development and IT Operations. Instead of wasting time trimming to-do lists and reprioritizing commitments at weekly executive reviews, set up a *kanban board*.

Essentially, a kanban board is a visualization of the ongoing IT projects. By putting all the projects in one place, everyone on a team can easily see exactly how much is going on and what needs to be done at any time; there's no need to sort through or keep track of fragmented emails, text messages, or phone calls.

To make a kanban board, create three columns on a wall: Ready, Doing, and Done. Label index cards with all the ongoing work activities, and place them in their respective columns. Now, here's the kicker: limit the amount of Work In Progress, or WIP, to just four or five cards at a time. A key element of the First Way is reducing the number of moving parts. This

allows the team to focus on and complete a small number of tasks quickly before moving on to the next. When a task is completed, move the card to its new placement. For example, when you start a task, move its card from Ready to Doing.

Erik's First Way focuses on the success of the entire system, as opposed to a specific department or silo of work. Beyond reducing WIP, kanban boards also help teams commit to the right work – and the right amount of it – by highlighting what matters most to the company's performance as a whole. The bottom line is that until code is in production, no value is actually being generated – so removing needless work from the system is often more important than adding more in. And being able to say no to commitments is crucial; remember, the less WIP, the more effective you'll be!

The Second Way entails fixing quality at the source to avoid rework.

A 2007 Suzuki Hayabusa dragster motorcycle is a thing of beauty, Erik says. It can go over 230 mph. It has a six-gear constant mesh box and a #532 chain drive. What it does *not* have is a reverse gear.

He's trying to make a point. Like the high-speed motorcycle, IT's workflow shouldn't have a reverse gear. Work that moves backward – a result of defects or ambiguity – is literally waste. Ideally, the flow of work should go in just one direction: forward.

To continue the automotive analogies: preventive oil changes and vehicle maintenance are what the Second Way is all about.

The key message here is: The Second Way entails fixing quality at the source to avoid rework.

The Phoenix Project release intervals are currently nine months long. But if Bill wants to design quality into the product at the earliest stages and ensure forward flow, that's way too long of a time frame. He needs much faster, amplified feedback loops from the customer or IT Operations back into Development.

Or, as Erik says, "Stop thinking about Civil War era cannons. Think antiaircraft guns."

The goal is *single-piece flow*, or one unit of product flowing between the different processes. In other words, work on one piece at a time. By cutting out wait time between steps, you maximize throughput, limit W IP, and minimize errors.

Take a manufacturing example: Toyota's infamous *single-minute exchange of die*. This term describes how, through a series of improvements, the company achieved single-piece flow and cut its hood-stamping changeover time from three days to under ten minutes.

In IT, single-piece flow can be achieved through the *continuous delivery* approach, which emphasizes three things. The first is small batch sizes. Working in smaller batches allows you to quickly apply feedback loops and course-correct if needed.

The second is stopping the production line when problems crop up. That means not taking on any new work when builds, tests, or deployments fail.

Third, create fast, automated test suites to ensure that code is always ready to be deployed. Remember, code isn't "done" when Development is done coding; it's when code has been tested and is operating as designed.

Erik suggests a little change to Phoenix's release intervals. Instead of having one deployment every nine months, Bill should aim for ten deploys – per day.

Impossible, Bill thinks.

But that's exactly what Flickr achieved back in 2009. At the time, most IT organizations were doing quarterly or annual deployments. With Dev and Ops working together with the business and implementing the Second Way's methods, Flickr was able to create a one-step environment create-and-deploy procedure – and do ten deployments a day! That's *one thousand* times faster than any of its contemporaries.

Today, that number's even higher. Amazon, Netflix, and Google – just a few of the organizations that have adopted the DevOps processes and cultural practices – are able to deploy thousands of production changes daily while still offering exceptional stability and security.

The Third Way is about creating a culture of continuous experimentation, failure, and improvement.

Bill is gazing at his shiny new laptop. His applications and data are all there, his email is functioning, every piece is intact – in short, it works perfectly. If Bill's old laptop was a metaphor for everything that was wrong with Parts Unlimited, his new laptop represents everything his team has accomplished together in the past few months.

SEV1 outages are down by two-thirds, and incident recovery time has been cut in half. The ticketing system has been purged of unnecessary work. There's a standardized environment creation process – or *build procedure* – to keep Development and Operations in sync and remove variance.

Developers are working with Security to create preventive projects instead of securing things after deployment. They've appeased the auditors – and reduced security-related work by 75 percent, allowing efforts to be focused elsewhere.

All this means that project flow has sped up. Phoenix is running smoothly. People are clear on what their tasks are; they feel fulfilled and happy because they can do their jobs. And, via the Third Way, they're innovating more than ever.

The key message here is: The Third Way is about creating a culture of continuous experimentation, failure, and improvement.

According to the Third Way, if you're not getting better, you're getting worse. There are three parts to this. The first is about *experimentation*. In order to beat competitors, you need to out-experiment them – so it's important to encourage innovation and risk-taking.

At Parts, there's a new project that focuses on innovation, called "Unicorn." Using data and environments copied from Phoenix, engineers created an identical but completely separate database where they can continuously develop and test things like customer-targeted promotions – without disrupting the live application.

The second part is about *failure*. To that end, Parts set up their "Simian Army Chaos Monkey" project. Its task? Create devastating bugs to kill processes or whole servers. At first, it was mayhem; the test infrastructure repeatedly crashed. But as Development and IT Operations collaborated to make the code and infrastructure more durable, the IT services became immune to failure.

The third aspect involves putting the *improvement* of daily work over daily work. At Parts, all managers must improve something – anything – every two weeks. These so-called *improvement kata* cycles keep the system under constant pressure and force it to advance.

In martial arts, kata is the act of practicing a pattern so it becomes second nature. Habits are what lead to mastery. Studies show that five minutes of daily practice are more effective than three hours once a week – so the more frequently a habit is practiced, the better.

In today's technology-driven world, IT isn't just a department. It's a skill – like being able to read or write. Business managers need to possess that skill in order to take calculated risks, which are in turn necessary to beat competitors. When business and IT realize they're fundamentally inseparable, and implement DevOps principles and practices, the entire organization wins.

Final summary

The key message in these blinks is that:

In today's technology-driven environment, the IT department often forms the backbone of a company. So when communication and workflow between the Development and IT Operations teams are misaligned, it has repercussions all the way to the top. But internal chaos doesn't have to be the norm. Enter DevOps: a unified IT strategy that centers on the "Three Ways." These business principles emphasize short feedback loops, streamlined communication, and a culture of experimentation, repetition, and practice. By implementing these processes, you can take your IT organization – and entire company – to unprecedented heights.

And here's some more actionable advice:

Create a personal kanban board.

In addition to preventing work jams and boosting productivity on the organizational scale, kanban boards can also increase efficiency on a more personal level, by helping you commit

to the right amount of work and visualize your progress. So swap out those detailed, contextual to-do lists for a blank wall and some Post-its. Categorize all the work you need to do into three columns: Ready, Doing, and Done. Limit your WIP to about four or five items at any given time – and get cracking! As you progress, those cards will move from left to right, and you'll likely achieve your goals faster than ever before.