

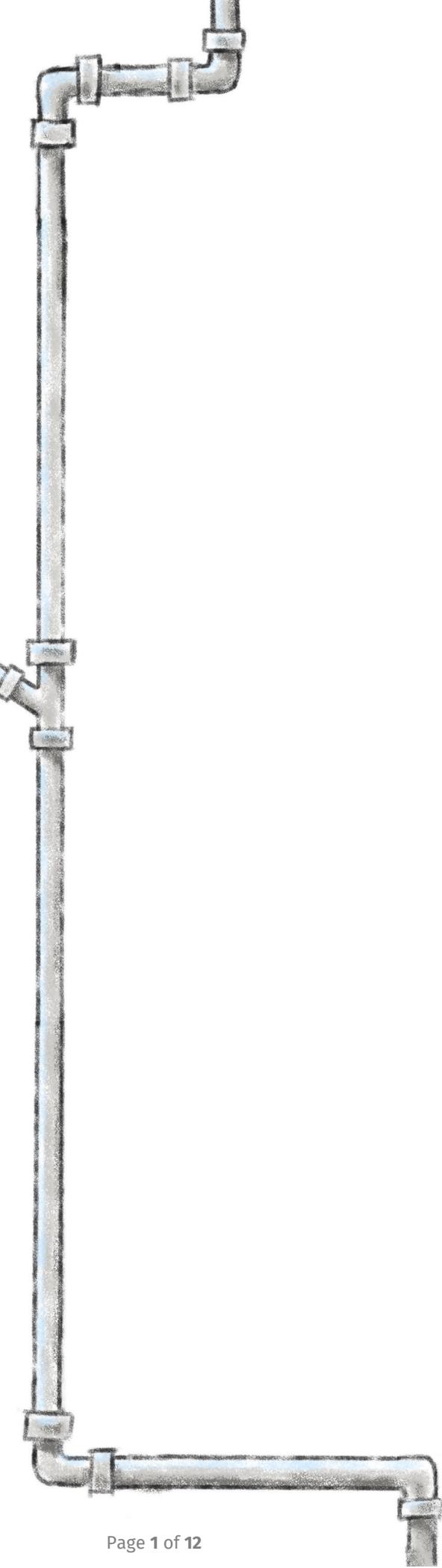
Continuous Integration

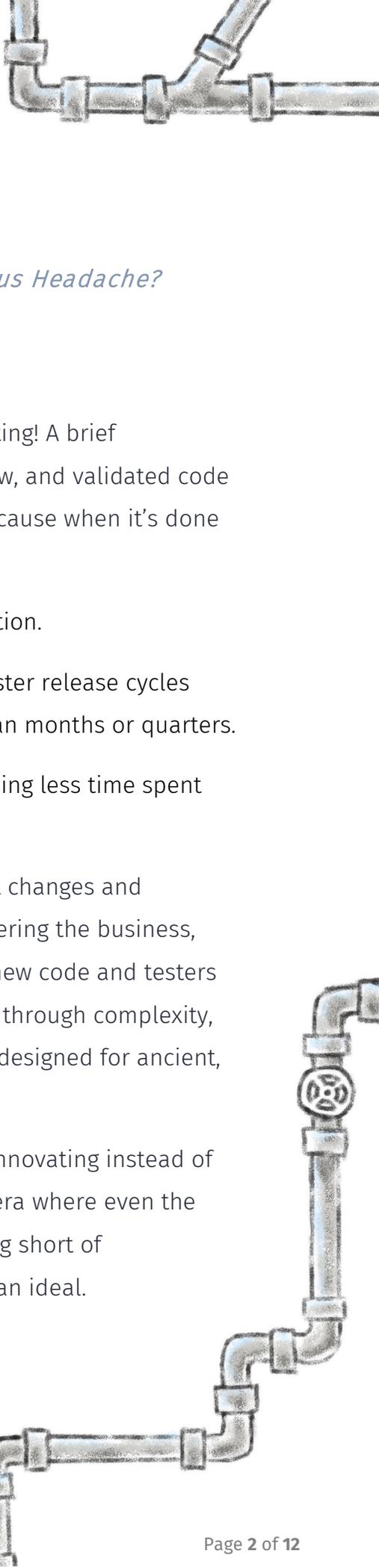
Continuous Delivery

Continuous Headache?

Learn about practical CI/CD implementation strategies with effective real-world benefits.







CI/CD/CH?

Continuous Integration / Continuous Delivery / Continuous Headache?

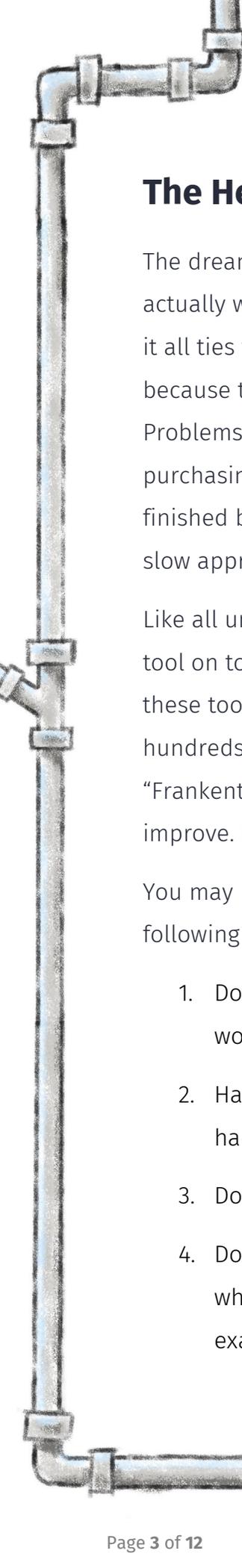
The Promise

CI/CD was a dream come true. Automated builds! Automated testing! A brief verification and then—*click!*—instant deployment of beautiful, new, and validated code sailing straight to production. It can and should work like this because when it's done correctly, it delivers:

- **Better** quality software with fewer defects through production.
- **Faster** delivery of business ideas to market by enabling faster release cycles and allowing software changes in days or weeks rather than months or quarters.
- **Cheaper** implementation across the entire lifecycle, including less time spent coding, deploying, and testing software changes.

The true value of CI/CD is that organizations can adapt to market changes and innovate faster than their competitors. Change the software powering the business, and then release those applications as fast as developers write new code and testers verify it. Prior to CI/CD, unpredictability stalled software changes through complexity, bugs or defects during testing, poor code quality, and processes designed for ancient, mainframe systems.

Ideally, CI/CD allows business stakeholders to spend their time innovating instead of dealing with problems. Compared to the pre-historic, pre-CI/CD era where even the smallest of software changes led to big, risky releases, it's nothing short of revolutionary. Unfortunately, the real-world experience is less than ideal.



The Headache

The dream of CI/CD has not materialized for most organizations. In fact, things are actually worse than before. Despite plenty of automation, no one seems to know how it all ties together. Critical releases move slower and have more defects than before because too many tools overlap duties, and each tool has its own set of idiosyncrasies. Problems lead to finger pointing. Senior decision makers wonder why they approved purchasing these tools. Managers worry about their team morale and getting releases finished bug-free. Engineers can't access the detailed execution logs they need and a slow approval process stymies up-and-coming developers.

Like all unholy messes, a CI/CD headache isn't built overnight. It emerges after bolting tool on top of tool in various parts of the software delivery process over time. Each of these tools may make sense, but when you bring everything together—including the hundreds of interdependent applications relying on these tools—you end up with a “Frankentool” no single person can fully understand, let alone properly use or improve. It becomes a monster.

You may have already created a “Frankentool” and not even know it yet. Answer the following questions, and then keep reading.

1. Do only a handful of engineers understand how your software delivery process works?
2. Have you experienced an information bottleneck because only 1 or 2 people had the information integral to a build or release?
3. Do less than 5 people interface with the CI/CD process?
4. Does it take hours to track down basic information—like which changes are in which build artifact, which environment those builds were deployed to, or what exactly was tested?

5. Have business stakeholders built their own “metasystems” using a series of issue tracking tools, spreadsheets, meetings, sticky notes, and emails just to figure out which application is in which state?
6. Can your organization deliver software only as fast as the handful of people who understand the complex systems can deliver information and decisions?

If you answered “yes” to any of the previous questions, you likely have a CI/CD headache on your hands. Don’t panic. There’s a way out of this mess. Inedo’s BuildMaster was specifically designed to help you navigate the complex and technical landscape to realize the dream of CI/CD without the headache.

A Little About Software Delivery with CI/CD Pipelines

Real world CI/CD relies on a balance between automated building, automated testing, and manual verification to quickly deliver ideas to production...and roll those ideas back quickly if they don’t work out.

Figure 1: The CI/CD Pipeline

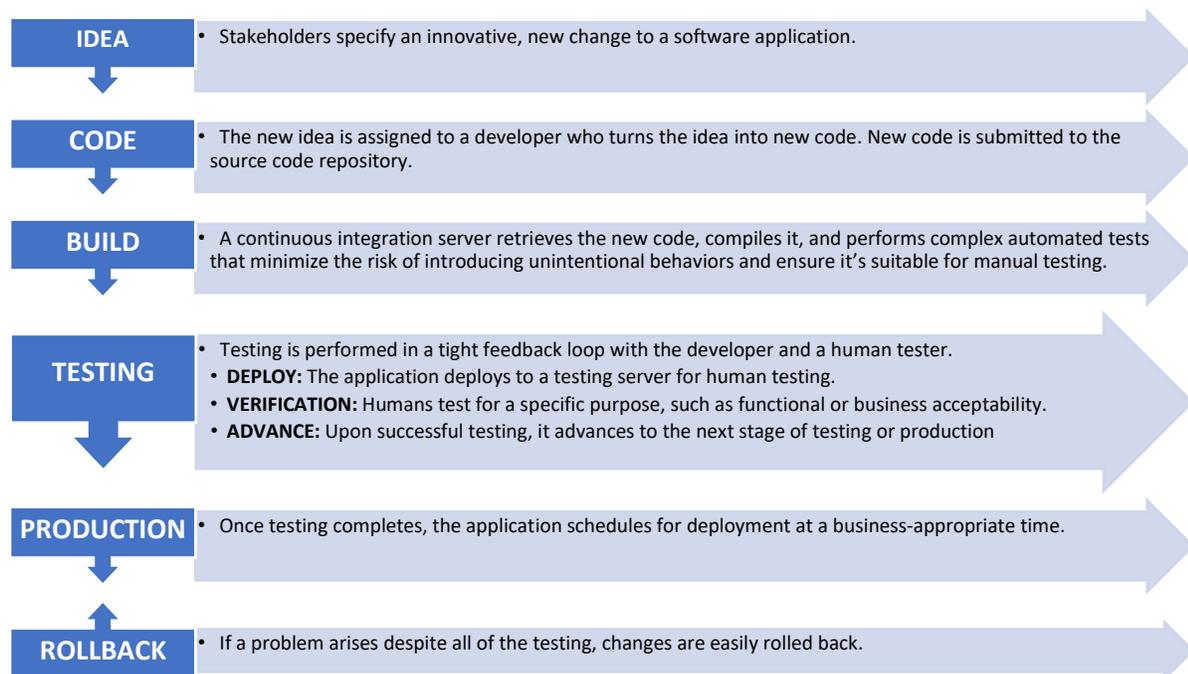
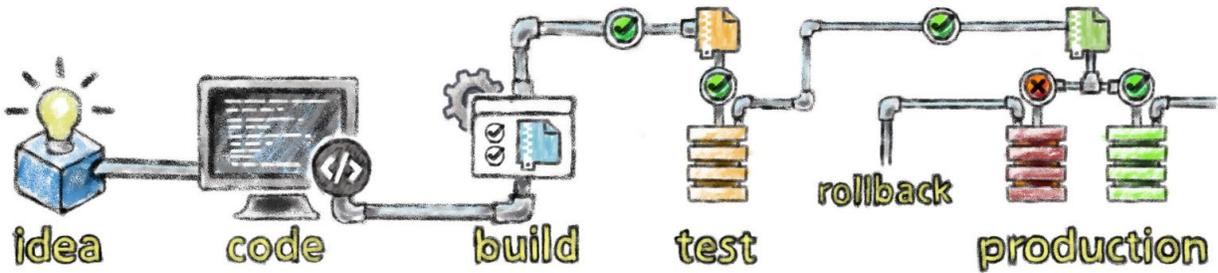


Figure 2: CI/CD Pipeline Idealized



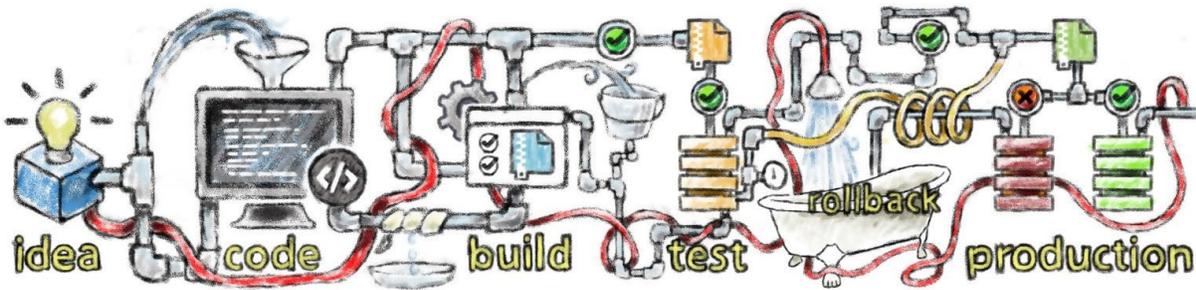
Nearly all organizations have a less-than-ideal software delivery process leading to less-than-ideal releases. A poor process can only result in delays, bugs, and a product falling short of the vision. In today's marketplace, the pressure to deliver releases faster and faster strains the software delivery process. Therefore, the process **must be seamless**. At Inedo, we have pinpointed multiple points of potential process breakdowns.

Breakdown 1: Brittle Toolchain Links

Software delivery processes and tools link together through a combination of people-driven conventions and software-driven integrations. For example, when a developer enters an Issue ID as a source control commit note, a script will post a comment to the issue tracking system with the Commit ID.

As the pace of software delivery increases in the real world, not only do these brittle links become more and more important, but so does establishing even more links between the disparate tools and systems. But, each new link remains just as brittle as existing links. They are often implemented with some hastily written script tied to an arcane feature of one of these tools.

Figure 3: CI/CD Pipeline Failure



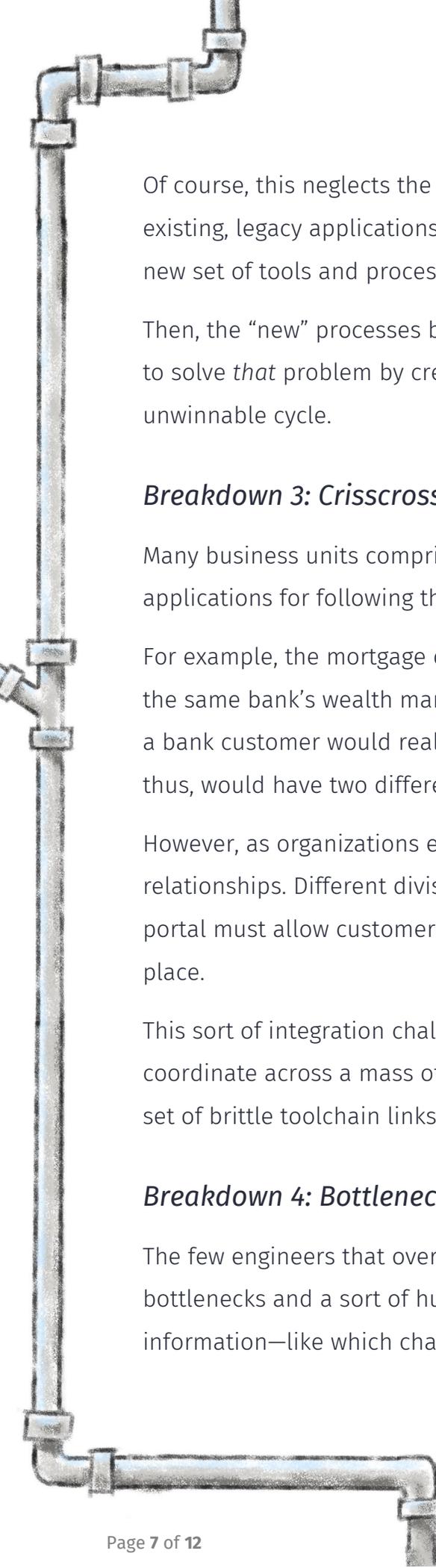
At the same time, the pressure to speed up software delivery increases and creates new opportunities for human error and script failure. These failures can be costly: if a developer neglected to properly link an Issue ID or the script processing it failed, imagine the challenge of trying to identify which one of dozens of commits were related to which one of dozens of issues. These brittle links compound and only get worse as tools and complexities grow.

Breakdown 2: Browning Greenfields

A greenfield is a brand-new application built from scratch to solve an organization's problematic software delivery process. No patching. No maintaining old software. It's a fresh start! But, changing the whole system costs, and, inevitably, the new software becomes old software after two to three years.

A brownfield is an old greenfield. It's the legacy software systems and processes that are not ideal and more complicated to maintain.

Many organizations realize the challenge of overhauling their software delivery processes and invest in a "greenfield" process for new, "greenfield" applications. They hope, over time, the new applications will gradually replace the existing applications and underlying software delivery processes.



Of course, this neglects the actual, larger problem (i.e. effectively delivering the existing, legacy applications already creating business value), but it also establishes a new set of tools and processes unavoidably becoming brownfields.

Then, the “new” processes become too risky to overhaul, and someone else will decide to solve *that* problem by creating yet another “greenfield” to use. It’s a vicious, unwinnable cycle.

Breakdown 3: Crisscrossing Dependencies

Many business units comprise organizations. Each unit has their own sets of applications for following their own delivery processes.

For example, the mortgage division at a bank would use totally different systems than the same bank’s wealth management division. Historically, this hasn’t been a problem; a bank customer would realize these are, in fact, two different business divisions and thus, would have two different relationships with the bank.

However, as organizations evolve, consumers seek simpler and more integrated relationships. Different divisions must become aligned. For example, a unified banking portal must allow customers to see their mortgage and investment accounts in one place.

This sort of integration challenges developers in and of itself, but it also must coordinate across a mass of greenfield and brownfield processes, each with their own set of brittle toolchain links.

Breakdown 4: Bottlenecks

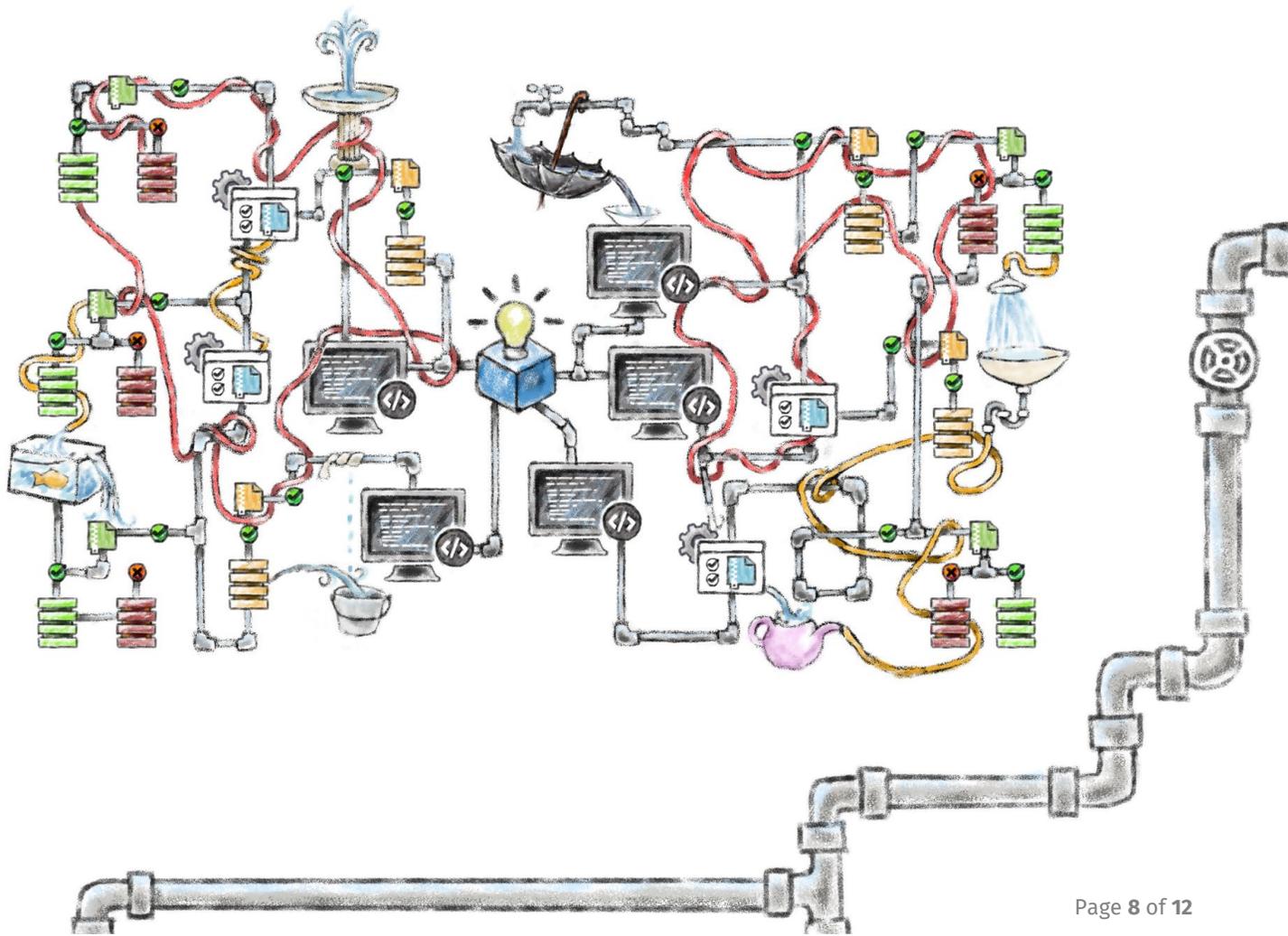
The few engineers that oversee these tools and processes quickly become information bottlenecks and a sort of human interface to the CI/CD process. Tracking down basic information—like which changes happened in which build artifact and in which

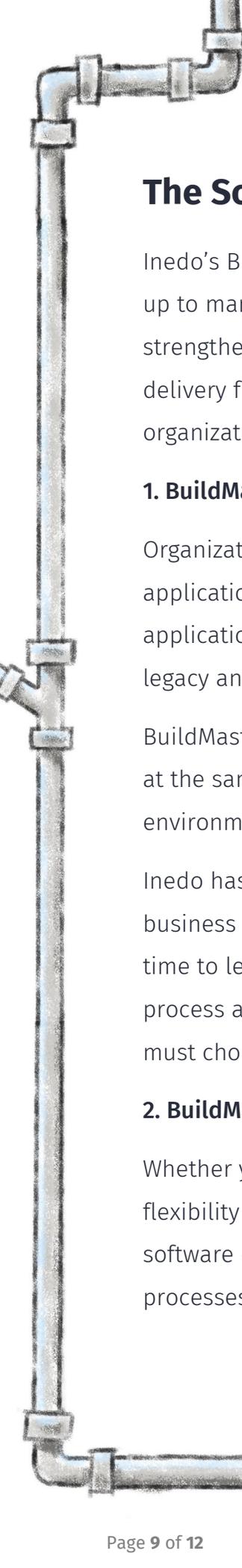
environment the build deployed to, and what exactly was tested—can take hours because only a few, very busy people can track down the answer. By the time they answer, it's already outdated.

As a result, business stakeholders build their own “metasystems” using a mess of issue tracking tools, spreadsheets, meetings, sticky notes, and emails just to figure out which application is in which state, and when they can release new changes. Then, the stakeholders become decision bottlenecks as well in emerging as another human interface to these incomprehensible metasystems.

Ultimately, such an organization can deliver software only as fast as the handful of people who understand the complex systems can deliver information and decisions. As more complexity and automation gets added, fewer and fewer people can manage them. The process only serves failure.

Figure 4: The Metasystem Mess





The Solution: BuildMaster

Inedo's BuildMaster delivers CI/CD for the real world. It was designed from the ground up to manage and automate the software delivery process. The application strengthens brittle links between the tools you already use and oversees software delivery for all of your applications—legacy or otherwise—in a way everyone in your organization can understand. Here's how we do it.

1. BuildMaster Embraces Complexity

Organizations fighting the “Browning Greenfield” cycle can relax with BuildMaster. The application does not make you choose between new and legacy systems. The application does not force a single way. In the real world, there will always be a mix of legacy and new systems.

BuildMaster allows you to use the old systems, develop new systems, and use the two at the same time in a seamless process. The application lets you adapt to a changing environment and embrace reality.

Inedo has designed BuildMaster for constant reuse and adaptation to fit your changing business needs. We address the complex nature of IT and make it work *for* you. It's time to let go of the vicious flip-flopping between an old, working software delivery process and shiny, new, ideal process that is expensive and unproven. You no longer must choose.

2. BuildMaster Works at Your Speed

Whether your organization aims for 5 deployments a day or 1 per year, BuildMaster's flexibility works at any speed. We designed BuildMaster, first and foremost, to manage software delivery processes and allow organizations to define and change those processes at their required speed.



Because software powers so many processes, the software must be able to adapt at an ever-increasing rate. Currently, your organization may have a mismatch between the ability of separate teams to deliver and the release cycle. Though one application team can deliver releases on a reliable two-week cycle, the organization can only change as fast as its slowest application. With BuildMaster, each team integrates and works at the optimal speed.

3. BuildMaster Decentralizes CI/CD

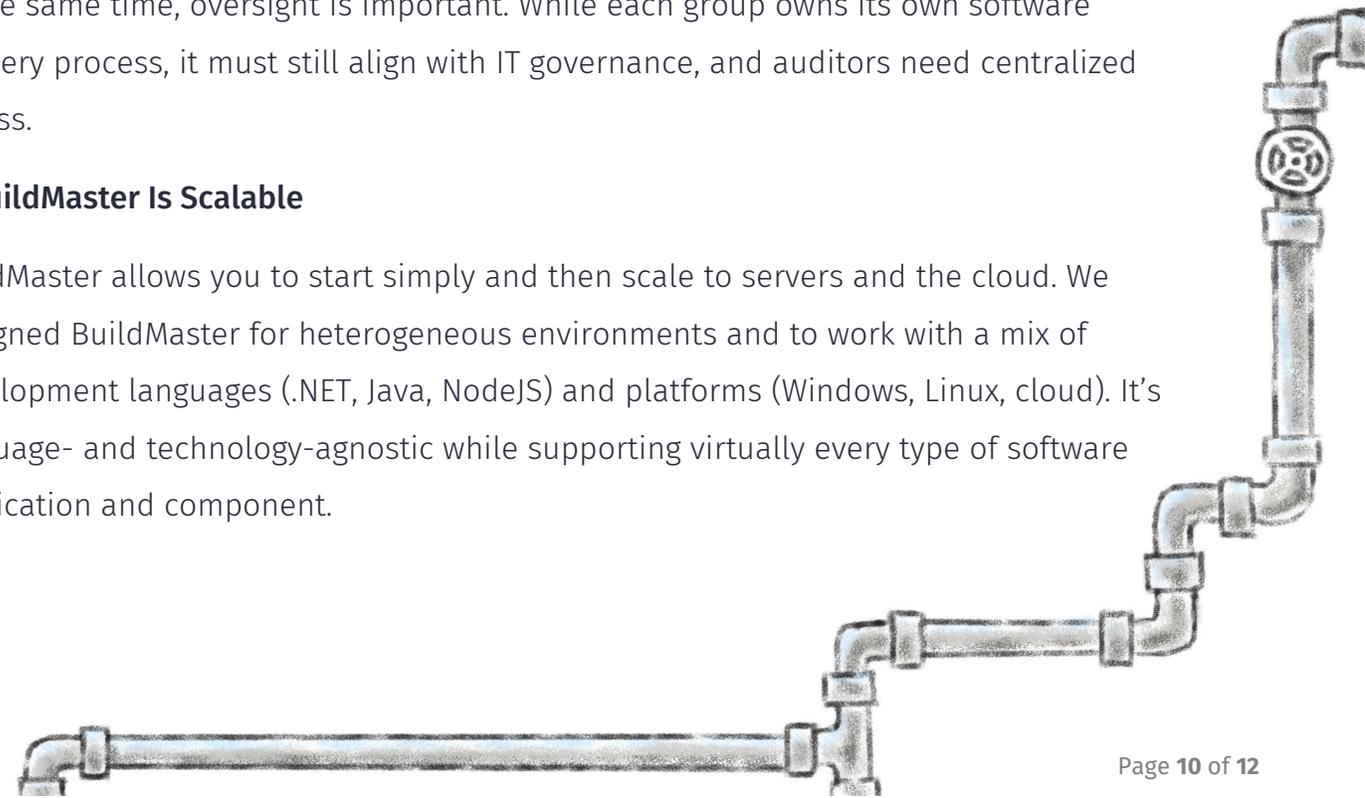
Forcing a uniform CI/CD process throughout the organization quickly results in noncompliance and, ultimately, an even worse mess of undocumented or even rough processes. Instead, we provide groups with tooling that lets them define and take responsibility for their own software delivery processes.

Groups are finally responsible for meeting their own goals, and they must become proficient at changing their own software-driven processes. Just like there's no one-size-fits-all software application, there's no one-size-fits-all software delivery process. BuildMaster, a self-service platform with fine-grained permissions, allows different groups to build and deploy their own applications while defining their delivery processes to any configured environment.

At the same time, oversight is important. While each group owns its own software delivery process, it must still align with IT governance, and auditors need centralized access.

4. BuildMaster Is Scalable

BuildMaster allows you to start simply and then scale to servers and the cloud. We designed BuildMaster for heterogeneous environments and to work with a mix of development languages (.NET, Java, NodeJS) and platforms (Windows, Linux, cloud). It's language- and technology-agnostic while supporting virtually every type of software application and component.





When engineers hear the word “scalable,” they tend to think of things like distributed systems and load-balancing. While these are important considerations for CI/CD tooling, it’s equally important that all of an organization’s different groups adopt.

BuildMaster is intuitive to learn and use. But, companies do not have to take our word for it. Inedo now offers multiple training options including hands-on, classroom-style and self-guided training. We understand different groups have different backgrounds and skill levels. Inedo commits to making sure training facilitates BuildMaster’s adoption.

5. BuildMaster Focuses on Business Stakeholders

BuildMaster’s design befriends every level of your organization. Executives, managers, engineers, and developers—everyone can use the program easily.

Managers have 24/7 access to the dashboard. It shows the information they need in real time and at a glance, making it easier and simpler to report. The dashboard tracks the status of every process in the production cycle and allows managers to approve and sign off on the dashboard to deploy the next environment. The days of delay when managers waded through a series of emails to check status before approving are over. It’s all on the dashboard. No email needed. In addition, managers can set up automated approvals triggered by the successful completion of milestones.

Inedo understands engineers live and die by the details. With BuildMaster, engineers have access to granular information including comprehensive execution logs showing exactly what happened on which server, when, and by whom. Developers can maximize their efficiency because they no longer must wait for approvals.

Next Steps

Step #1: The High-Level Interview

We understand the stress behind quickly delivering a bug-free, software product. We want to hear your vision and learn your goals. What are your current problems?



What keeps you up at night? Our executive team listens. It's free, easy, and happens at the management level.

Approximate time: 1 hour.

Step #2: Into the Weeds

Once you decide BuildMaster demands more investigation, we patch our technical and engineering team with your technical team. This conversation gets way down into the weeds. BuildMaster should fit every level of your organization and align with your organization's vision and technology goals. We ensure no one wastes time and resources.

Approximate time: 1 hour.

Step #3: Hands on Demonstration

Inedo's best engineers demonstrate exactly how BuildMaster automates, accelerates, and integrates software delivery processes. You learn the program and experience our tested and proven on-boarding strategies. Our experience and clients tell us BuildMaster is extremely intuitive, but, if you need them, our professional training services meet your needs.

Approximate time: 1 hour.

Step #4: Guided Proof of Concept

We can prove BuildMaster will work for you. Our team performs a real-life proof of concept. Our engineers collaborate with your team and solve a real problem you have. We know the value of testing a new program before implementing it, and we offer proof of concept in recognition of this.

Approximate time: Highly dependent.

