Continuous Delivery & DevOps

…or the **agile organisation**

Leif Sørensen
les@praqma.com

Praqma

Continuous Delivery & DevOps experts and evangelists

Tools & Automation experts. We help customers with practical implementation of their development process.
We don't chop wood - we sharpen axes!

7 years, 25 employees, offices in Copenhagen, Aarhus, Oslo & Stockholm

Events: Jenkins CI User Events, Continuous Delivery & DevOps Conferences, DayOfContainers, Automation Nights, Code Academy (code-conf.com)

Service partner to:

Agile Manifesto

Four doctrines

Twelve principles

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Working software is the primary measure of progress.

Processes

Waterfall

Agile

Watergile? Agilefall?

Is this a problem?

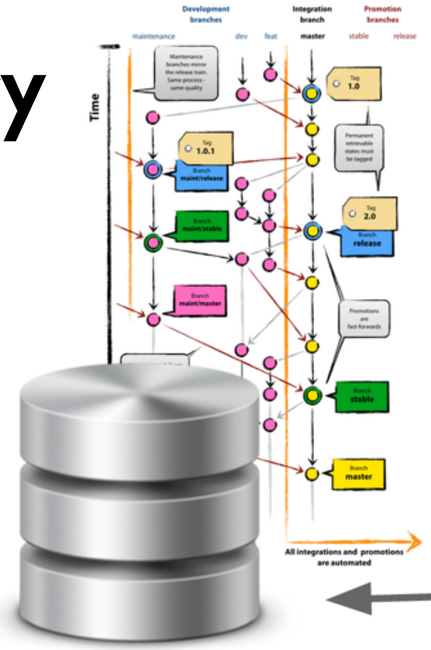
- No early and continuous deliveries to the customers
- Unpredictable deliveries
    - If the software doesn't have to work, you can always meet any other requirement (Gerald Weinberg)
- Building up technical debt
- Quality is put on as an afterthought
- The most expensive software - is the one developed but not (yet) used

Why?

The frequent delivery of working software to the customer (without compromising quality) requires:

- Agile architecture
- Agile Test / Quality Assurance
- Agile Infrastructure / Deployment / Operations

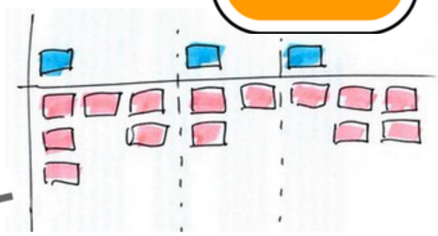
Continuous Delivery Story-line



VCS

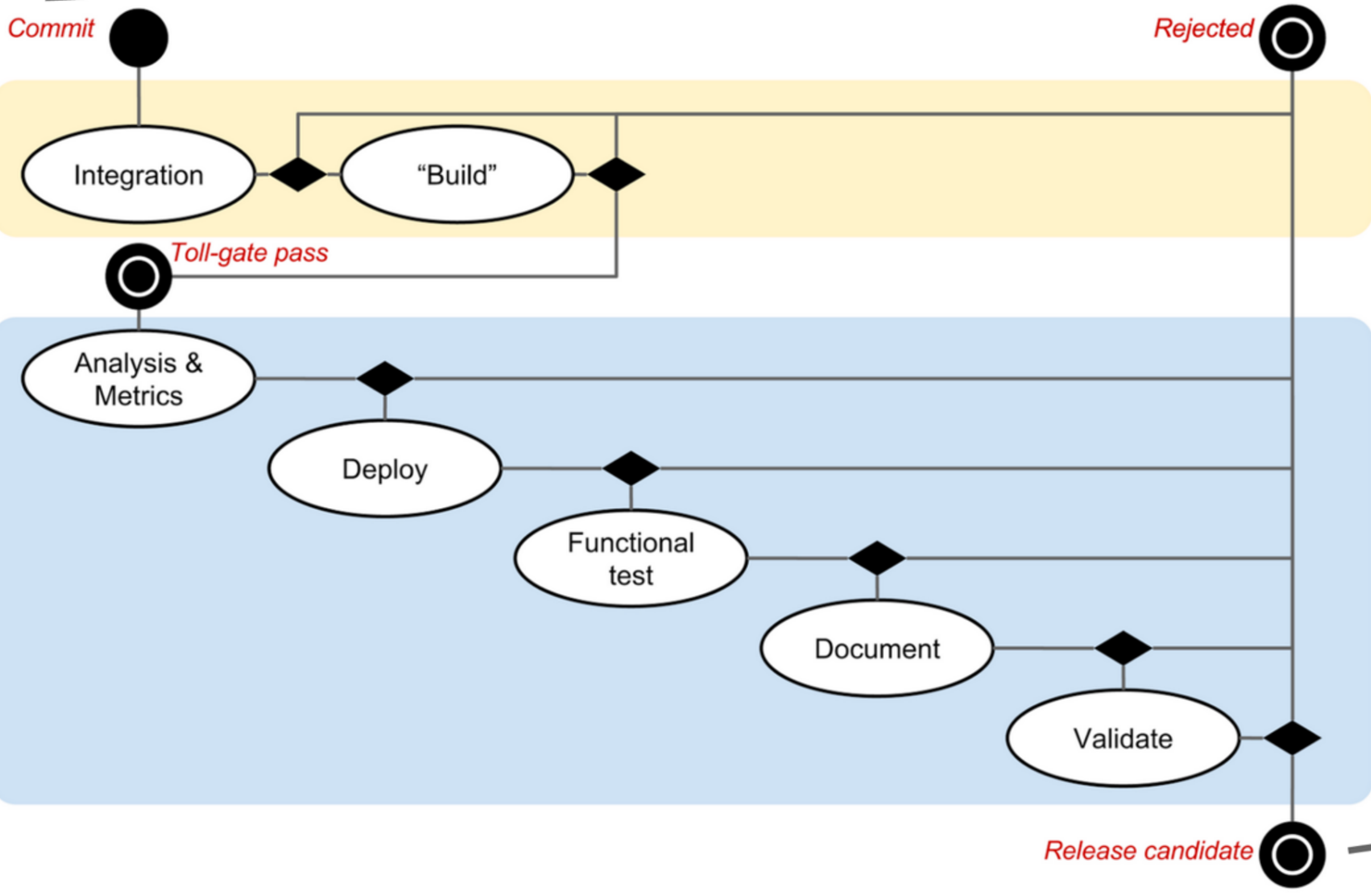
Automation Platform

Developer



Work item

...nope!



Done done

Continuous Delivery

Why CoDe?

- Quality
- Broken process / workflow
- Developer productivity - cost of development
- Time to market - time of development
- Predictability / visibility
- Receiver organisation (Customer) requirement - quality gateway
- Continuous Delivery - release to production
- Facilitate customer / user feedback
- Because it is a requirement for agile development

- Continuous Improvement - natural next step

Additional benefits with CoDe?

- Errors / problems found earlier
- ‘Automatic’ traceability
- Documentation & visibility into the process
- ‘Automatic’ historic development
- ‘Forced’ focus on things you should have done anyway: automation, testability, DevOps,...

CoDe Maturity

	Novice	Beginner	Intermediate	Advanced	Expert
Build	Automated builds	Artifacts are managed	Automated release notes	Full trace	Delivery pipeline
Test	Unit testing, mock-ups and proxies	Automated functional tests	Maintain test data	Adaptive test suites	Test in production
Version Control	Commits are tied to tasks	Release train branching strategy	Version numbers matter	Individual history rewrites in DVCS	Pristine integration branch
DevOps	One Team	Automated deployment	Access to production-like environments	Infrastructure as code	Live monitoring and feedback
Architecture & Design	Code metrics	Testable code	Dependencies are managed	Individually releasable components	Full audit trail in production
Organization & Culture	Agile process	Buy-in from management	Tasks are groomed	Designated roles	Explicit knowledge transfer

Technical debt

Henrik Kniberg: The Solution to Technical Debt http://blog.crisp.se/2013/07/12/henrikkniberg/the-solution-to-technical-debt

Technical debt

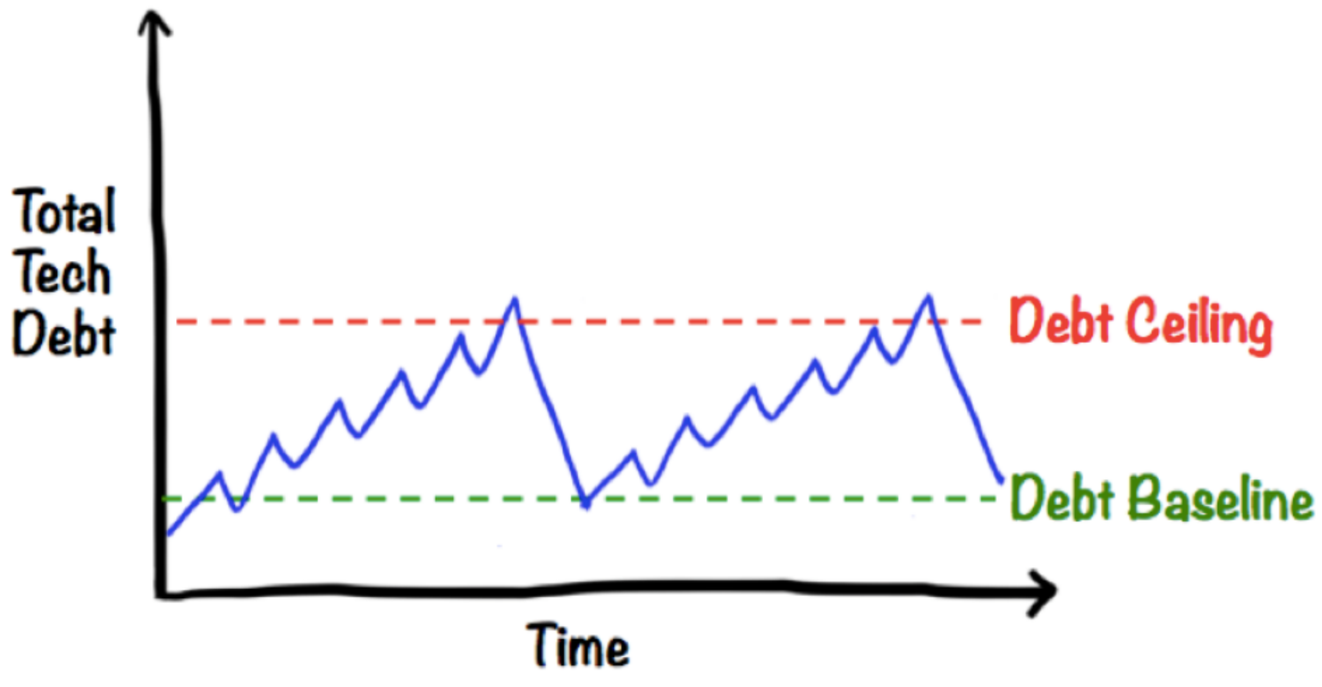
Anything about your code & development environment that slows you down. For example:

- Unclear, unreadable code
- Lack of test automation, build automation, deployment automation, and anything else that could be automated that you do manually today
- Duplicate code
- Tangled architecture & unnecessarily complex dependencies
- Slow, ineffective tools
- Uncommitted code & long-lived branches (hides problems that will slow you down later)
- Important technical documentation that is missing or out-of-date
- Unnecessary technical documentation that is being maintained and kept up-to-date
- Lack of test environments
- Long build-test cycle & lack of continuous integration

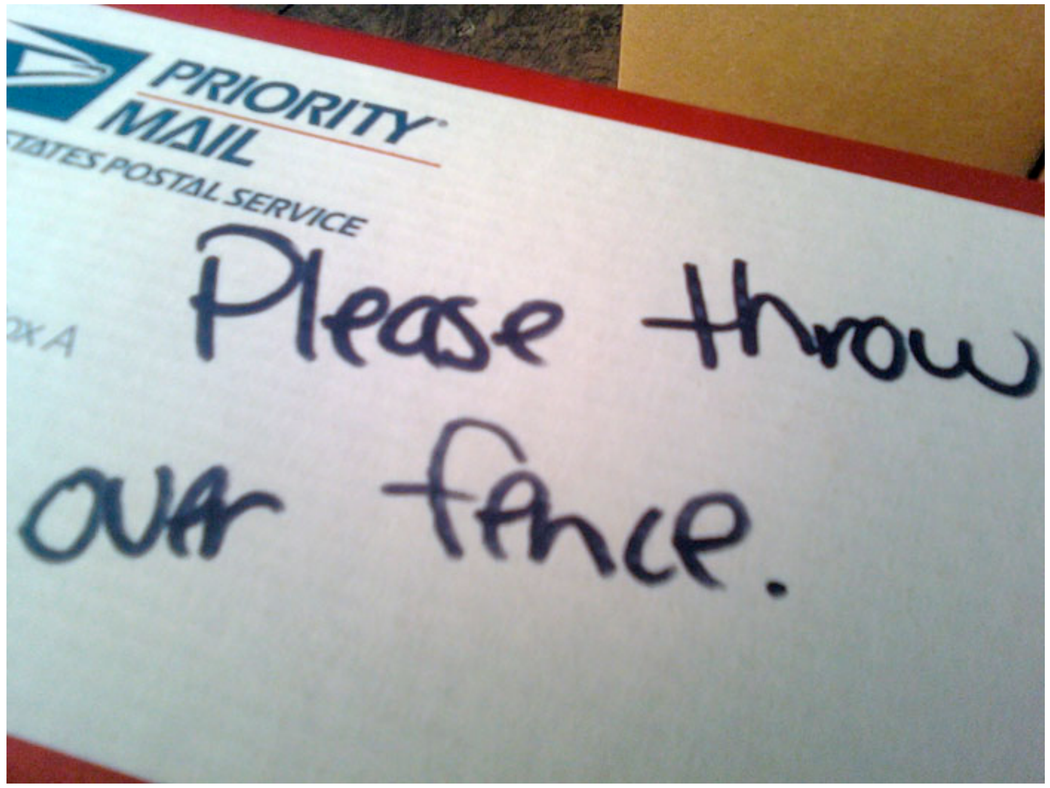
Technical debt

Things are the way they are because they got that way
(Gerald Weinberg)

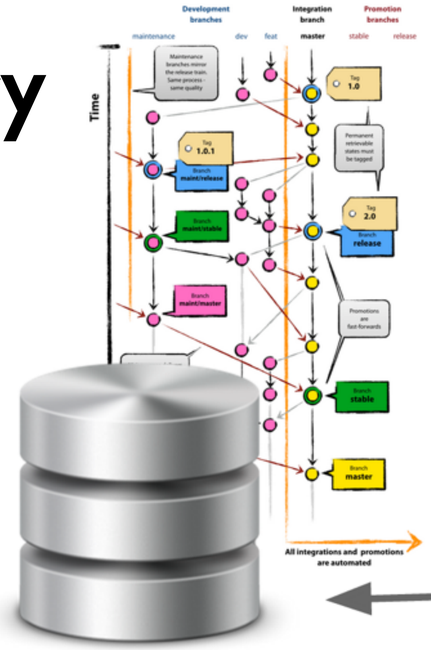
Solution: Debt management
(Henrik Kniberg)



Operations handover



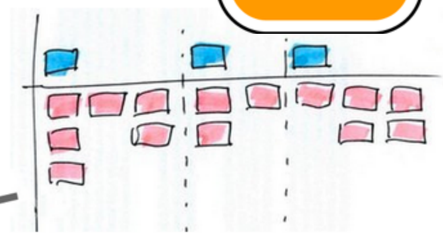
Continuous Delivery Story-line



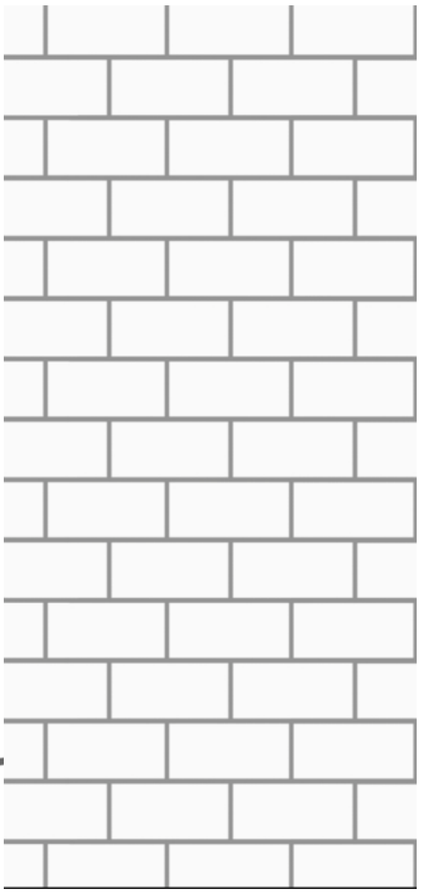
VCS

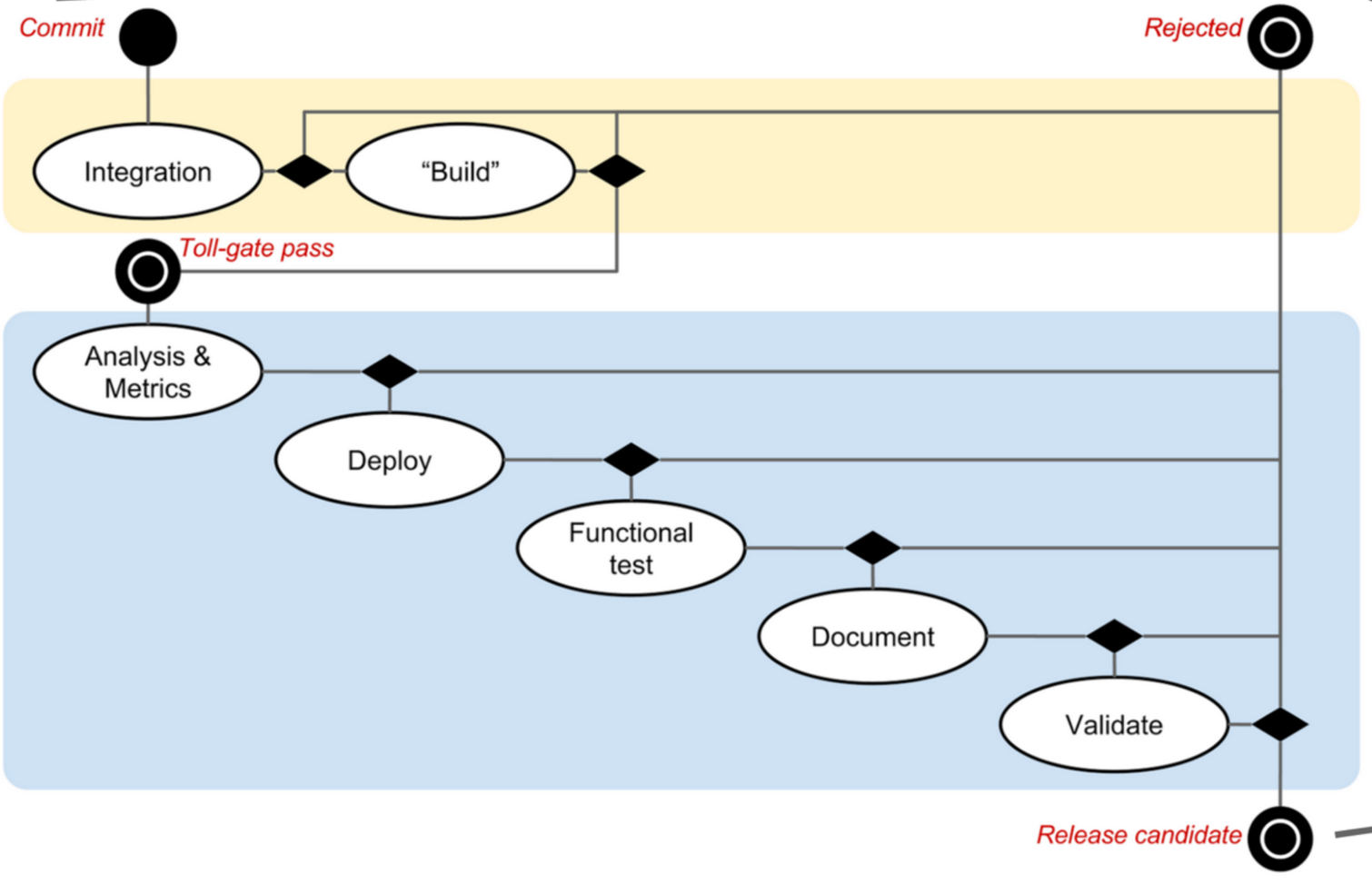
Automation Platform

Developer



Work item





Traditional Dev/Ops organisation

Development

- Focus on business functionality -
- Like Change
- Tools with focus on functionality
- Poor understanding of writing “operational” applications

Operations

- Focus on stability & availability
- Dislike changes
- Own working model
- Maybe outsourced

- No common tools
- Too little common understanding across the whole process
- Slow and expensive
- Support is difficult, and often across Dev & Ops
- A lot of ‘blame games’

KPIs in IT Operations

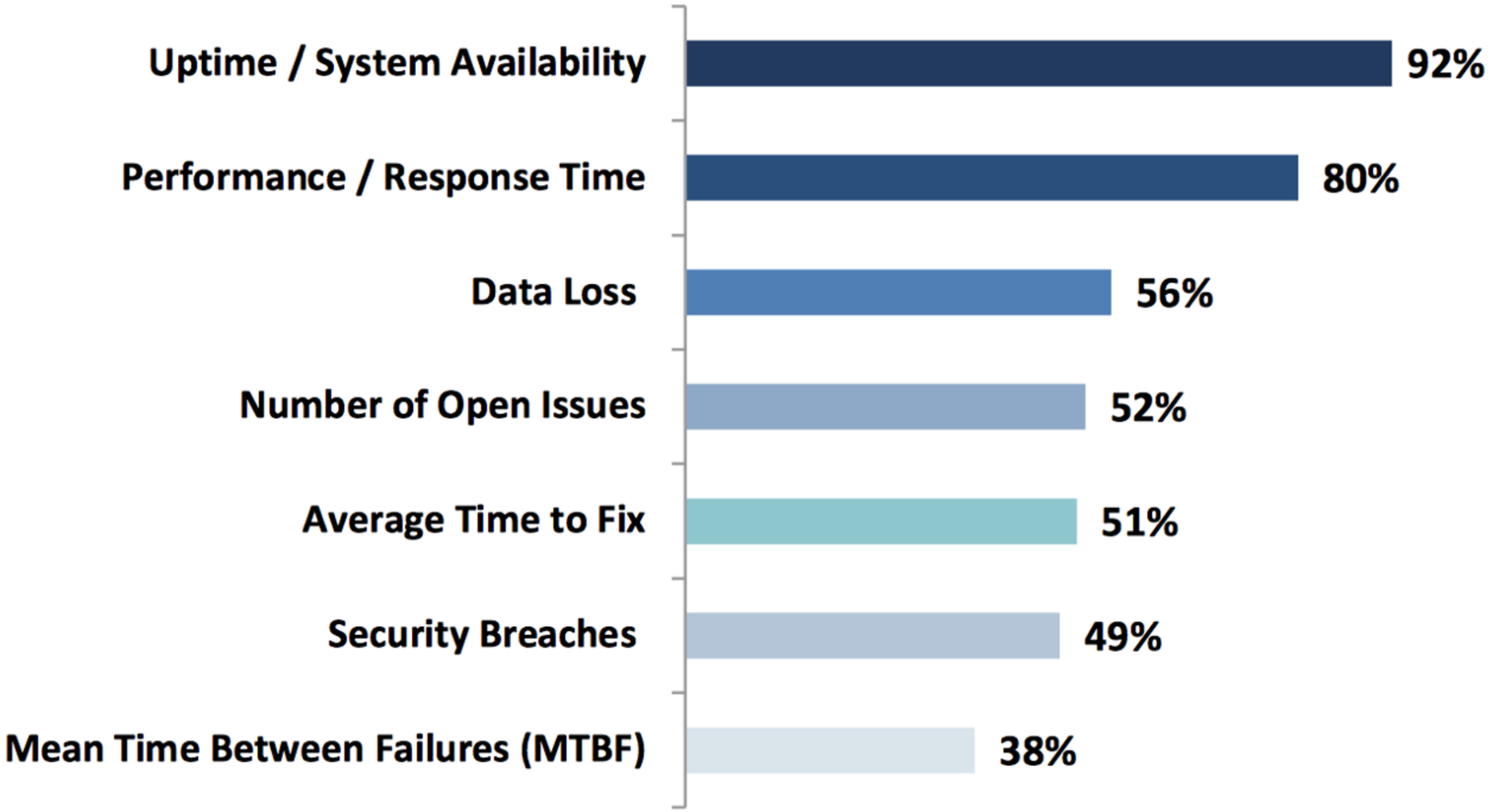


Figure 1: KPIs for IT operations excellence

DevOps Origin

August 2008: At the Agile Conference in Toronto, software developer Andrew Shafer posts notice of a "birds of a feather" session entitled "Agile Infrastructure". Exactly one person attends: Patrick Debois. Based on their talk, they form the Agile Systems Administration Group.

June 2009: At the O'Reilly Velocity 09 conference, John Allspaw and Paul Hammond give their now-famous talk entitled, "10+ Deploys a Day: Dev and Ops Cooperation at Flickr." Watching remotely, Debois laments on Twitter that he is unable to attend in person. Paul Nasrat tweets back, "Why not organize your own Velocity event in Belgium?"

October 2009: Debois decides to do exactly that - "I picked 'DevOpsDays' as Dev and Ops working together because 'Agile System Administration' was too long"

DevOps Definition

- Cooperation between Development & Operations
- Agile Infrastructure
- Agile System Administration

Or in other words:

Including Operations into the agile setup, so we can deliver the results of our agile development to the users

DevOps Definition

DevOps (a clipped compound of "development" and "operations") is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while automating the process of software delivery and infrastructure changes.[1][2] It aims at establishing a culture and environment where building, testing, and releasing software, can happen rapidly, frequently, and more reliably.[3][4][5]

Wikipedia

Why DevOps (now)?

- Separation of Development & Operations was maybe not a good idea in the first place
- Agile & Continuous Delivery is hitting the Operations wall
- With or without Agile methods: a lot of companies have serious difficulties getting things into production
- Continuous Delivery needs agile infrastructure
- With Cloud and other tools, Operations becomes ‘easy & cheap’
- Unicorns show how it is done

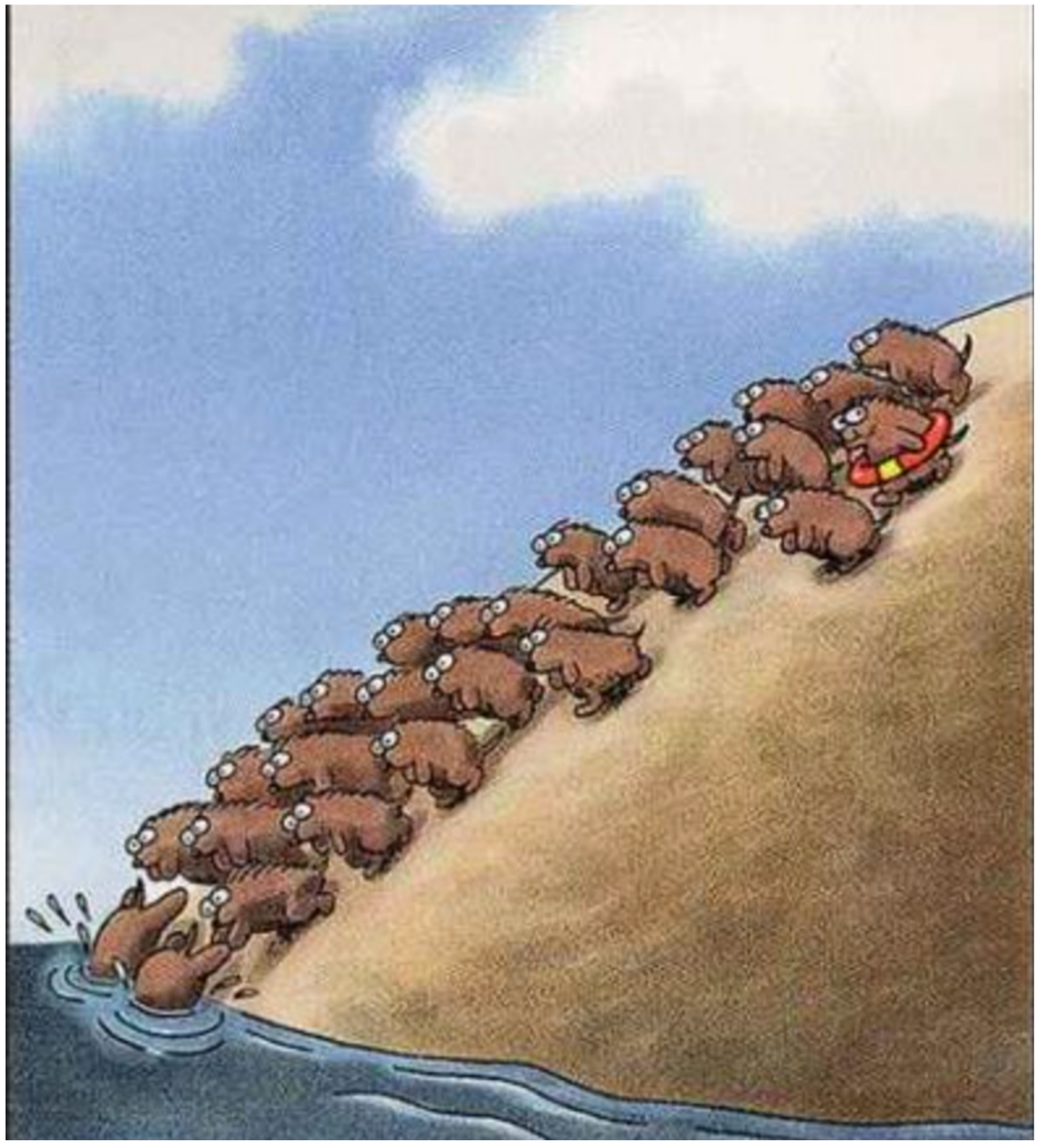
Business wants agility: early and continuous delivery of valuable software. Software developed, but not delivered, has no value

What is DevOps?

- Is it a culture?
- Is it a job title?
- Is it a team?
- Is it a way of organizing?
- Is it a tool stack?
- Is it a way of designing systems?
- Or just a way of thinking?

Or all of the above?

And do we all have to do it?



Outsourcing?

- Insource operations
- Insource development
- Or at least outsource to the same provider - and demand they do DevOps

- …but you can outsource your infrastructure - to the cloud

Organisation

- Delivering software from requirement to production is considered one process
  - Don’t change process & and tools in the middle
  - Don’t change ownership in the middle
  - Don’t change people in the middle
- Remove silos
- The teams responsible for developing the applications are also responsible for quality control, maintenance and operation

Culture

- Create common goals
- Culture of cooperation, respect & trust
- No blame games
- Dont shoot the messenger
- “Continuous Improvement” is part of the culture
- Quality is build into the whole process
- If something is difficult, do it more often

Team

- DevOps teams develop, automate and support the process
- Development teams have DevOps expertise included
- Full stack developers, both involved in development, support and operations

Platform

Pick a platform that supports development, implementation & operations

Architecture

- Architecture to focus on both development development and operationability (support for automation, surveillance,…)
- Avoid monolithic application architecture: Independent deployable components
- Automate everything
- Infrastructure as code
- Anything as code

Create an architecture that supports development, implementation & operations

Tools

- Integrated tool stack across the process
    - Version control
    - Case handling
    - Deployment
    - Artifact management
    - Automation and orchestration
    - Test management & automation
- Tools with good support for operationability

Pick tools that supports development, implementation & operations

Cloud and other tools

amazon web services

Microsoft Azure

vmware

ANSIBLE

puppet labs

CHEF

Jenkins

XL DEPLOY

urban{code} Deploy

docker

Cloud and other tools

Build Quality In - The Andon Cord

How to get started?

- Big change for a big organisation with a lot of legacy systems
- Can be done incremental
- Technology is ready
- Start by doing Continuous Delivery - use the results as leverage towards operations
- Assess yourself: Where are you? Where do you want to be?
- Introduce debt mangement: work actively with Maturity / Technical Debt

All Unicorns were horses

Amazon, up until 2001, ran on the OBIDOS content delivery system, which became so problematic and dangerous to maintain that CTO Werner Vogels transformed their entire organization and code to a service-oriented architecture

Twitter struggled to scale capacity on their front-end monolithic Ruby on Rails system in 2009, starting a multiyear project to progressively re-architect and replace it

LinkedIn, six months after their successful IPO in 2011, struggled with problematic deployments so painful that they launched Operation InVersion, a two-month feature freeze, allowing them to overhaul their compute environments, deployments, and architecture

All Unicorns were horses

Facebook, in 2009, was at the breaking point for infrastructure operations. Barely able to keep up with user growth, code deployments were becoming increasingly dangerous and staff were continually firefighting. Jay Parikh and Pedro Canahuati started their transformation to make code safe to deploy again

Gene Kim, https://opensource.com/business/15/4/6-common-devops-myths

Something to read

- CD Maturity Paper: http://www.praqma.com/papers/cdmaturity
- 10+ Deploys Per Day: Dev and Ops Cooperation at Flickr: http://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr/76
- What is DevOps, Damon Edwards: http://dev2ops.org/2010/02/what-is-devops/
- Top 11 Things You Need To Know About DevOps, Gene Kim: http://www2.netuitive.com/rs/netuitive/images/Top11ThingsToKnowAboutDevOps.pdf
- The Convergence of DevOps, John Willis: http://itrevolution.com/the-convergence-of-devops/)

Thank you!

PRAQMA

DOC-CPH

DAY OF CONTAINERS - COPENHAGEN

DATE
29 AUG 2016

TIME
09–18:00

WHERE
COPENHAGEN

CODE-CPH

CONTINUOUS DELIVERY AND DEVOPS CONFERENCE - COPENHAGEN

DATE
13 SEP 2016

TIME
09–17:00

WHERE
COPENHAGEN

CODE ACADEMY

LEARN THE ESSENTIALS OF CONTINUOUS DELIVERY

DATE
22 – 26 AUG 2016

WHERE
AARHUS

www.code-conf.com/