DevOps

Tearing Down the Walls

DHBW

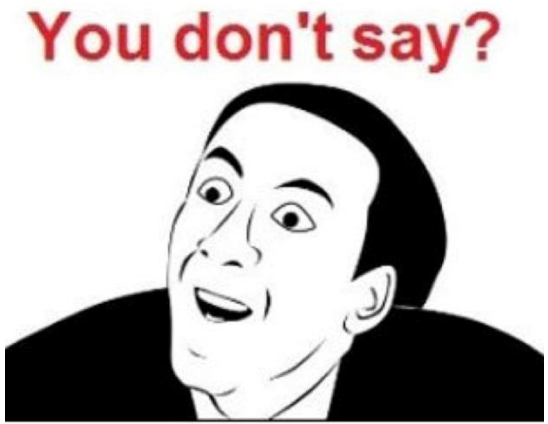
Aufbau

1. Was ist DevOps?
2. Die 5 DevOps-Prinzipien
3. Kritik
4. Fazit
5. Ausblick

DHBW

Was ist DevOps?

Deveploment + **Op**erations = **DevOps**



DHBW

Was ist DevOps?

DevOps ist eine **Bewegung**, die das Ziel hat, die **Time-To-Market** einer **Änderungseinheit** zu **reduzieren** - bei gleichzeitiger Gewährleistung **hoher Qualität**.

Dies soll durch die konsequente Anwendung von **Lean-Prinzipien** auf den **gesamten Software-Wertstrom** erreicht werden.

Vgl. Len Bass, Ingo Weber, and Liming Zhu. 2015. *Devops: A Software Architect's Perspective* (1st ed.). Addison-Wesley Professional., S. 13

DHBW

Was ist DevOps?

To make error is human. To propagate error to all server in automatic way is #devops.

DHBW

Relevanz

“Gartner Says By 2016, DevOps Will Evolve From a Niche to a Mainstream Strategy Employed by 25 Percent of Global 2000 Organizations”

Press Release, 05.03.2015, http://www.gartner.com/newsroom/id/2999017

“The number of DevopsDays conferences have increased from 1 in 2009 to 18 all over the world in 2014.“

Effective DevOps, O'Reilly Media, Early Release May 2015

DHBW

Relevanz

Google Trends, abgerufen am 05.02.2016
https://www.google.de/trends/explore#q=DevOps&cmpt=q&tz=Etc%2FGMT-1

DHBW

Begriff „DevOps“

- Wurde geprägt von Patrick Debois durch Ausrichtung der „DevOpsDays“ 2009 in Belgien
- Hat seine Wurzeln in „Agile Infrastructure“ (Deboir and Shafer) und „Lean Startup“

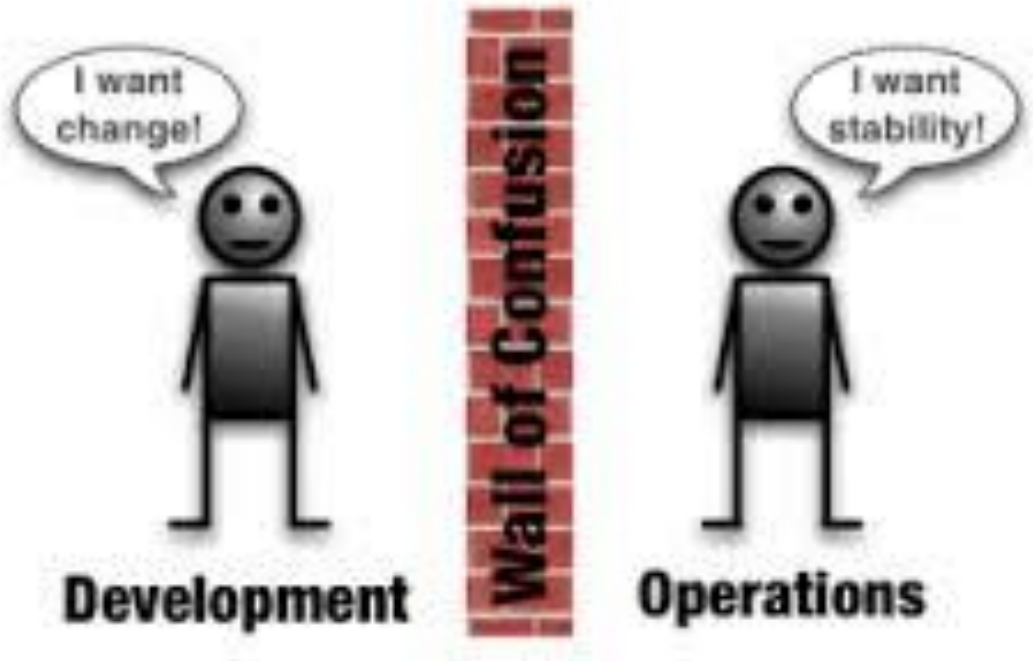
DHBW

Das Grundproblem: Warum DevOps

- Entwickler (Dev) sollen **schnell Veränderungen** umsetzen
- Administratoren (Ops) sollen **Sicherheit und Stabilität** der Systeme gewährleisten
- Die unterschiedlichen Ziele führen zu „**Silo-Denken**“ und bilden eine imaginäre „**Mauer** “ zwischen Dev und Ops
- Agile Methoden erhöhen Dev-Geschwindigkeit und damit Druck auf Ops

DHBW

Der Zielkonflikt zwischen Dev und Ops



Conway‘s Law

“organizations which design systems … are constrained to produce designs which are copies of the communication structures of these organizations”

Teams, die nicht gut zusammenarbeiten produzieren Lösungen, die nicht gut zusammenarbeiten.

DHBW

Silo-Denken

Stereotypen

Dev aus Sicht von Ops:

- Ungeduldig
- Unbeherrscht
- Leichtsinnig
- „Prima Donna“-Künstler
- Keine Ahnung von stabilem Service

Ops aus Sicht von Dev:

- Langsam
- Unbeherrscht
- Zu restriktiv
- Feinde des Fortschritts
- Keine Ahnung von zeitgemäßen Anwendungen

DHBW

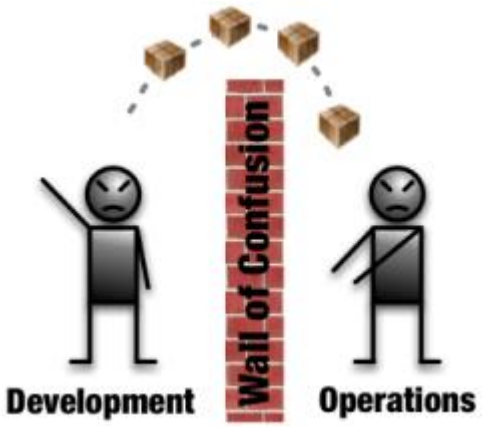
Silo-Denken


It's not my problem, the hole is in their side of the boat!
SPLOOSH!
DOOMED COMPANY
@rahuldighe

Silo-Denken

Silo-Denken: Konsequenzen

- mehr und länger andauernde System-Ausfälle
- Mehr Fehler
- Höheres Risiko (=weniger Gewissheit)
- Angst vor Veränderung -> Lähmung
- Längere Time-to-Market, schlechtere Qualität

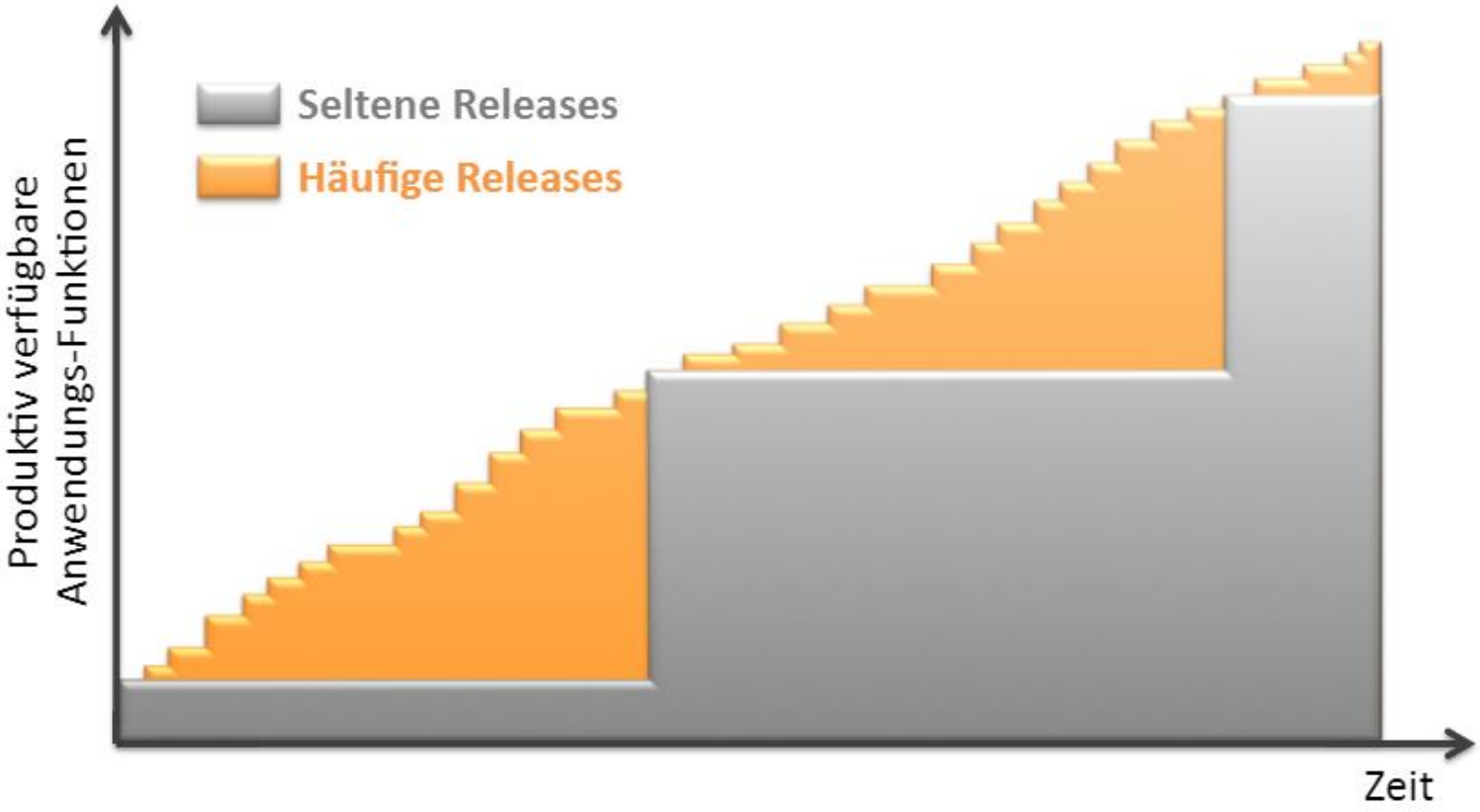


DHBW

Konsequenz: Wertstrom stoppt an „Wall of Confusion“

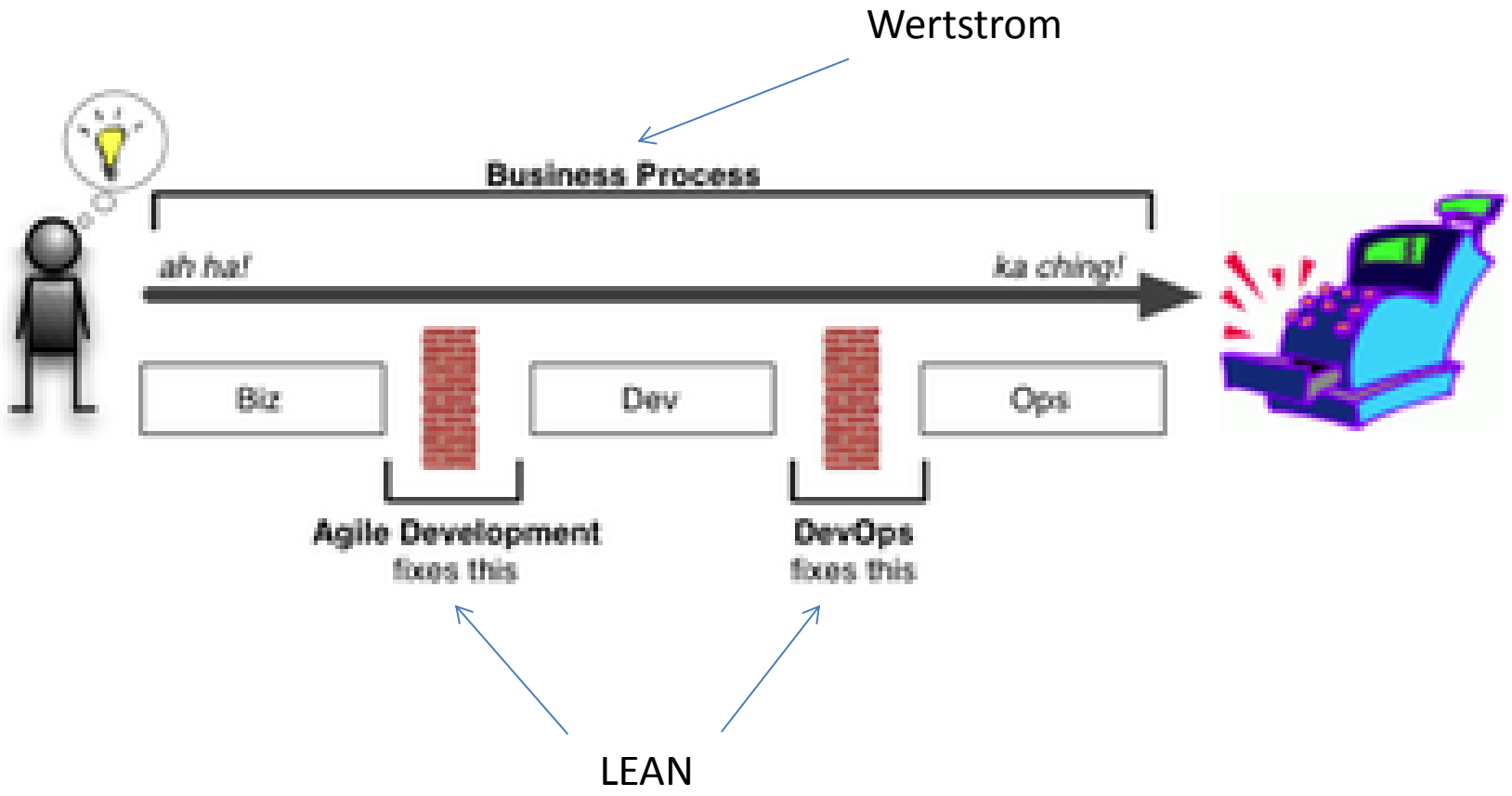
- Der Wertstrom „stoppt“ an der Mauer zu Operations
- Viel Zeit geht verloren (Verschwendung) - Features sind fertig, aber nicht produktiv verfügbar

Silo-Denken: Konsequenzen



DHBW

DevOps to the Rescue!



DHBW

CALMS – Die 5 DevOps-Prinzipien

- Culture
- Automation
- Lean
- Metrics
- Share

DHBW

Unsere Dramaturgie

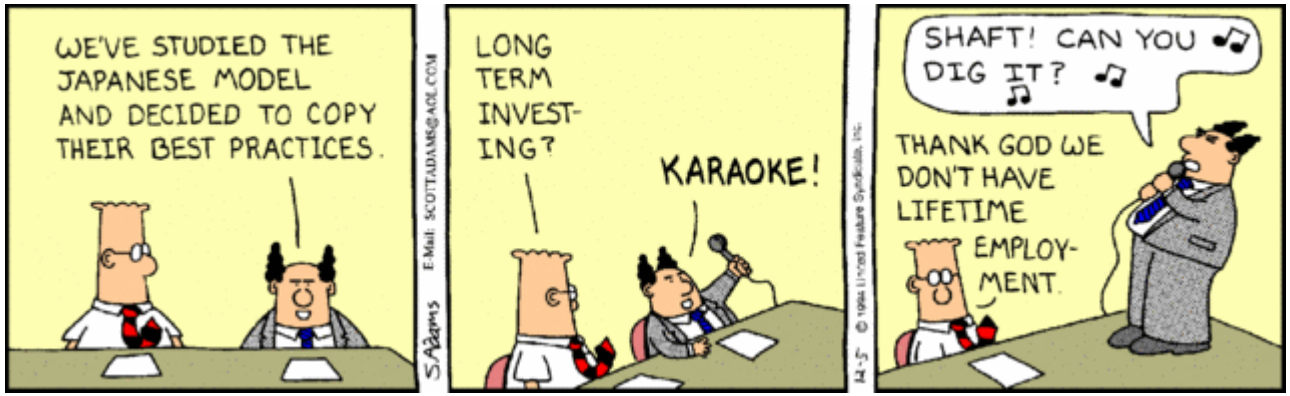
- LEAN
- AUTOMATION
- METRICS
- CULTURE
- SHARE

DHBW

Lean


LEAN
PRODUCTION
MANUFACTURING
WASTE
WORK
IMPROVEMENT
IMPLEMENTATION
MANAGEMENT
PRINCIPLES
SYSTEM
TOOLS
PROCESS
QUALITY
APPROACH
TIME
VALUE
CUSTOMER
IMPROVE
PRODUCT
PERFORMANCE
CHALLENGE
INVENTORY
PROBLEMS
FLOW
REDUCTION
TPS
OFTEN
CONTINUOUS
MANY
ONE
MAKE
PEOPLE
CONCEPT
ACTIVITY
EXAMPLE
CHANGE
BASED
THINKING
SERVICES
COST
DESIGN
TEAM
SEEN
EXAMPLES
DEVELOPMENT
GOALS
CONCEPTS
EFFORT
SEE
FOCUS
BECOME
COSTS
ALWAYS
LESS
INCREASE
RATHER
MUST
MAY
GOAL
WAY
EFFICIENCY
PROCESSES
PULL
THEREFORE
STARTED
ACHIEVE
BEST
TECHNIQUES
MOVEMENT
REQUIREMENTS
JUST
COMPANY
MUDA
MURI
THUS
MANY
WASTES
ELIMINATION
NEW
MAJOR
KEY
MURA
NOW
MASS
JIT

Was ist „Lean“?


WE'VE STUDIED THE JAPANESE MODEL AND DECIDED TO COPY THEIR BEST PRACTICES.
LONG TERM INVEST-ING?
KARAOKE!
SHAFT! CAN YOU DIG IT?
THANK GOD WE DON'T HAVE LIFETIME EMPLOY-MENT.

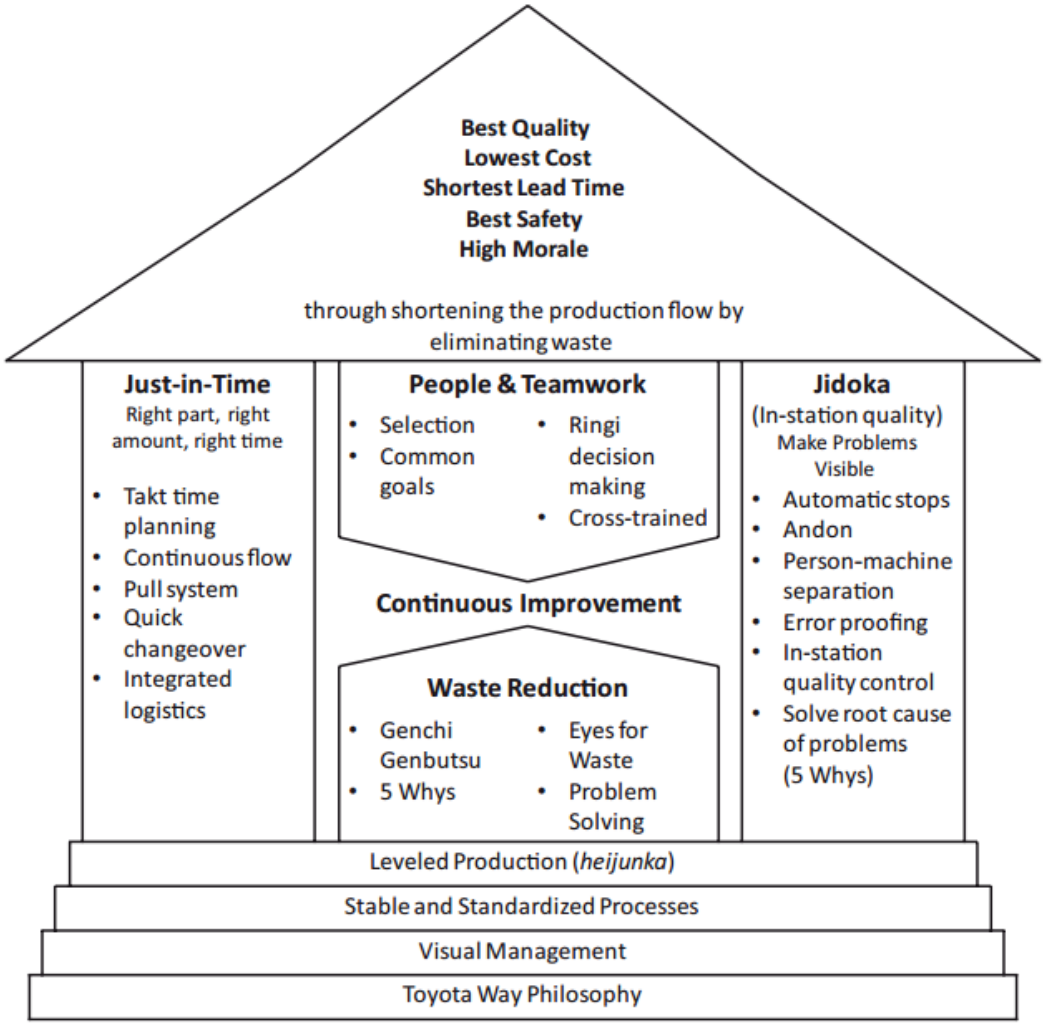
Was ist „Lean“?

- Eine **Philosophie**, die das Ziel hat, einen **Prozess** durch die **Eliminierung von Verschwendung kontinuierlich** zu **verbessern** und dabei die **Bedürfnisse der Kunden** als **Ausgangspunkt allen Handelns** sieht

Vgl. http://www.dummies.com/how-to/content/lean-for-dummies-cheat-sheet.html, abgerufen am 02.02.2016

DHBW

Ursprung: Toyota Produktions-System



Toyota-Produktions-System, nach Liker (2004)

Die drei Arten der Verschwendung

Muda:

„… jede menschliche Aktivität, die Ressourcen verbraucht, aber keinen Wert erzeugt.“(Wallace J. Hopp)

Mura:

Unausgeglichenheit, Unausgewogenheit, Inkonsistenz

Muri:

Überlastung von Menschen (psychische/physisch) oder Maschinen

Ausprägungen von Muda

„TIMWOOD“

- Materialbewegungen (*TRANSPORTATION*)
- Bestände (*INVENTORY*)
- Bewegungen (*MOTION*)
- Wartezeiten (*WAITING*)
- Verarbeitung (*OVER-PROCESSING*)
- Überproduktion (*OVER-PRODUCTION*)
- Korrekturen und Fehler (*DEFECTS*)

Die 5 Lean-Prinzipien

1. Den (zu produzierenden) Wert aus Sicht des Kunden definieren
2. Den Wertstrom identifizieren
3. Das Fluss-Prinzip umsetzen
4. Das Pull-Prinzip einführen
5. Perfektion anstreben

Vgl. https://de.wikipedia.org/wiki/Lean_Management#Prinzipien, abgerufen am 02.02.2016

DHBW

1. Wert aus Sicht des Kunden definieren

Was ist „Wert“?

- Eine Leistung, die den Anforderungen des Kunden hinsichtlich
  - Qualität
  - Verfügbarkeit (im Sinne von „da, wenn der Kunde es will“) und
  - Preis

  entspricht

- **„Wert“ ist, was der Kunde kauft**

DHBW

2. Wertstrom identifizieren

- engl. *Value Stream*
- alle wertschöpfenden und nicht-wertschöpfenden Aktivitäten, die notwendig sind, um ein Produkt bzw. eine Dienstleistung herzustellen und anzubieten
- Ziel des Wertstroms: Kunden zufriedenstellen
- Jeder Wertstrom umfasst einen oder mehrere Geschäftsprozesse
- Unterscheidung dreier Arten von Tätigkeiten:
  - Wertschöpfende Tätigkeiten
  - Nicht-wertschöpfende Tätigkeiten, die unvermeidbar sind
  - Nicht-wertschöpfende Tätigkeiten, die vermeidbar sind (Verschwendung)

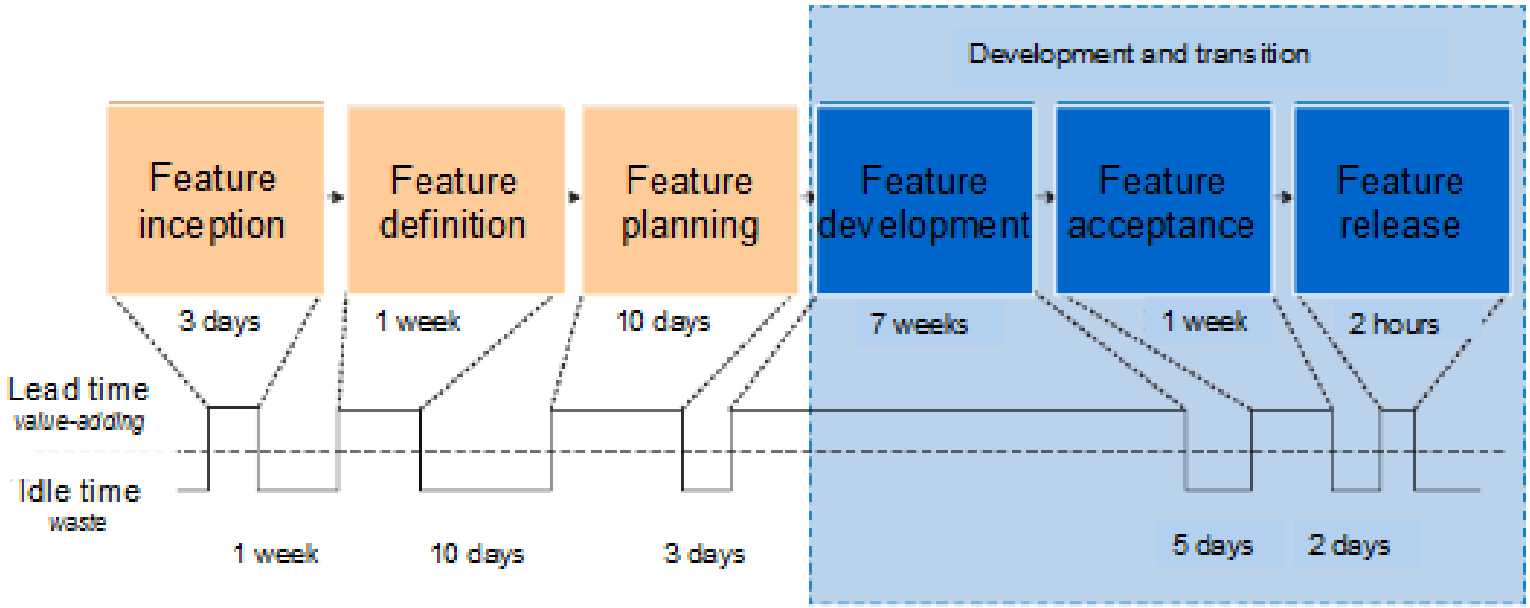
Vgl. https://de.wikipedia.org/wiki/Wertstrom (abgerufen 06.02.2015)

DHBW

2. Wertstrom identifizieren

- dient hauptsächlich der Aufdeckung von Verschwendung
- forciert ganzheitliche Betrachtung aller am Wertstrom beteiligten Prozesse
- trägt zu silo-übergreifenden Optimierung des Gesamtprozesses bei

Beispiel Wertstrom



Beispiel Wertstrom

- I asked him “When you did the Value Stream Mapping, did you do it across engineering and operations?” After about 15 seconds of dead silence he sheepishly answered, “Shit, we never thought about that.”

http://itrevolution.com/devops-culture-part-1/

DHBW

3. Fluss-Prinzip umsetzen

Das Produkt oder die Leistung soll:

- kontinuierlich
- mit möglichst wenigen Unterbrechungen, Wartezeiten und Umwegen
- über alle Stationen/Abteilungen hinweg
- in kleinen Änderungs-Einheiten (Features, Bugfixes, …)

zum Kunden **“fließen“.**

Dies wird durch die Beseitigung der identifizierten Verschwendung erreicht.

Kleine Änderungseinheiten

- Geringere TTM
- Geringere MTTR
- http://www.startuplessonslearned.com/2009/02/work-in-small-batches.html

DHBW

4. Pull-Prinzip einführen

- Es wird nur produziert:
  - **Was** der Kunde will
  - **Wenn** der Kunde etwas will

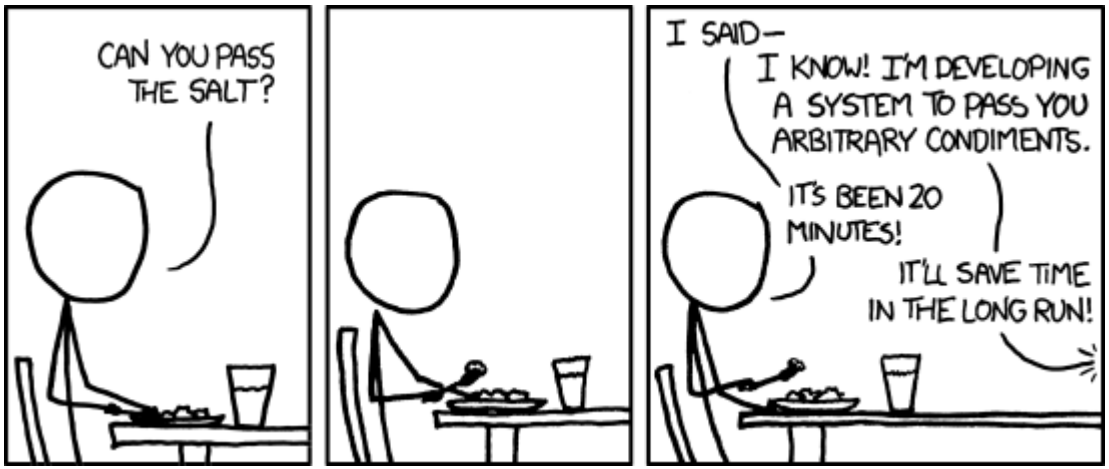
5. Perfektion anstreben

- Die vier vorgenannten Schritte werden kontinuierlich gemessen, geprüft und kritisch hinterfragt
- Dadurch werden immer weitere Verbesserungspotentiale aufgezeigt und umgesetzt
- Ziel: jede Tätigkeit und jedes Gut trägt zum Wert für den Kunden bei

Kaizen

PAUSE

Automation



DHBW

Automatisierung

- Das Ziel von DevOps ist **NICHT** die komplette Automatisierung des Wertstromes
- Automatisierung ist aber ein **wesentliches Hilfsmittel** zur **Optimierung** des Wertstromes durch die **Eliminierung von Verschwendung** und die Einführung eines **definierten Arbeitsflusses**
- Wichtig ist, zu wissen, wann es sich **lohnt**, etwas zu automatisieren – und wann **nicht**

DHBW

Automatisierungsstufen des Software-Entwicklung-Prozesses

(0) Automatisierte Entwicklungsumgebung

- Setup (IDE, Virtualisierung, …)
- build, test, check

1. Continuous Integration
2. Continuous Delivery
3. Continuous Deployment

DHBW

Continuous Integration

Continuous Integration:

- Automatisierter Bauvorgang aller Artefakte („deliverables“)
- Automatisierte Tests
- Automatisierte statische Code-Analyse

-> „Software kann immer gebaut werden und erfüllt bestimmte Qualitätskriterien“

DHBW

Continuous Delivery

Continuous Delivery: CI + Staging

- Automatisierte Installation und Konfiguration der verschiedenen Zielumgebungen
- Automatisierte Akzeptanztests
- Automatisierte Kapazitätstests
- Automatisiertes Deployment in verschiedene weitere Test-Umgebungen:
  - User Acceptance Testing (UAT)
  - Exploratory Testing
  - …
- **Aber: manuelles Auslösen des Deployments in Produktion!**
  (=Deployment erfolgt automatisiert, muss aber manuell gestartet werden)

->“Software **könnte** jederzeit in Produktion deployed werden“

DHBW

Continuous Deployment

Continuous Deployment:

- Konsequente Weiterentwicklung von CDelivery
- -> „Jede Änderung, die alle Stages erfolgreich durchlaufen hat, wird automatisch in Produktion deployed“

DHBW

DevOps und Automatisierung

- bereichsübergreifende Automatisierung zwischen Entwicklung und Produktivbetrieb erfordert Zusammenarbeit und gemeinsame Verantwortung für Installation und Konfiguration der verschiedenen Zielumgebungen (UAT, Staging, Produktion usw.)
- Lösung: **Infrastructure as Code**

DHBW

Infrastructure as Code

Grundidee:

- Infrastruktur wie Software behandeln („as Code“ eben)
- Anweisungen für das Aufsetzen und Konfigurieren der Umgebungen werden Teil des Quellcodes
- Änderung an Infrastruktur erfordert Änderung des Quellcodes

DHBW

Infrastructure as Code

„…Infrastructure as Code [liefert] keine Blaupause für jedes Unternehmen[…]. Vielmehr ist es das Ziel einer Reise, auf der Unternehmen ihre Entwicklungs- und Betriebsteams zusammenbringen, um verschiedene Bereiche in ihren Rechenzentren und Produkt-Pipelines anzupassen und zu automatisieren. So führt Infrastructure as Code schließlich zu einer gemeinsam designten Infrastruktur und gemeinsam definierten Abläufen, die stark an die Prozesse zur Entwicklung von Anwendungscode erinnern. Dadurch wird der Betrieb der Unternehmens-IT automatisier- und programmierbar.“

Vgl. http://www.heise.de/developer/artikel/Infrastructure-as-Code-Agile-Entwicklung-muss-ueber-den-Softwarecode-hinausgehen-3046561.html, abgerufen am 02.02.2016

Beispiel: Ansible

```
---
- hosts: webservers
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf

- hosts: databases
  remote_user: root

  tasks:
  - name: ensure postgresql is at the latest version
    yum: name=postgresql state=latest
  - name: ensure that postgresql is started
    service: name=postgresql state=running
```

Ansible-Playbook (yml)

DHBW

Beispiel: Puppet

```
node 'host2' {
  class { 'apache': }              # use apache module
  apache::vhost { 'example.com':  # define vhost resource
    port    => '80',
    docroot => '/var/www/html'
  }
}
```

Puppet-Manifest

DHBW

Infrastructure as Code: Tools

DHBW

Vorteile von Infrastructure as Code

- Keine Shadow-IT
- Erhöhung der Effizienz
- Infrastruktur wird mit Code kontinuierlich getestet und verwaltet
- Änderungen an Infrastruktur werden in Repository versioniert
- Zusammenarbeit von Dev und Ops wird forciert

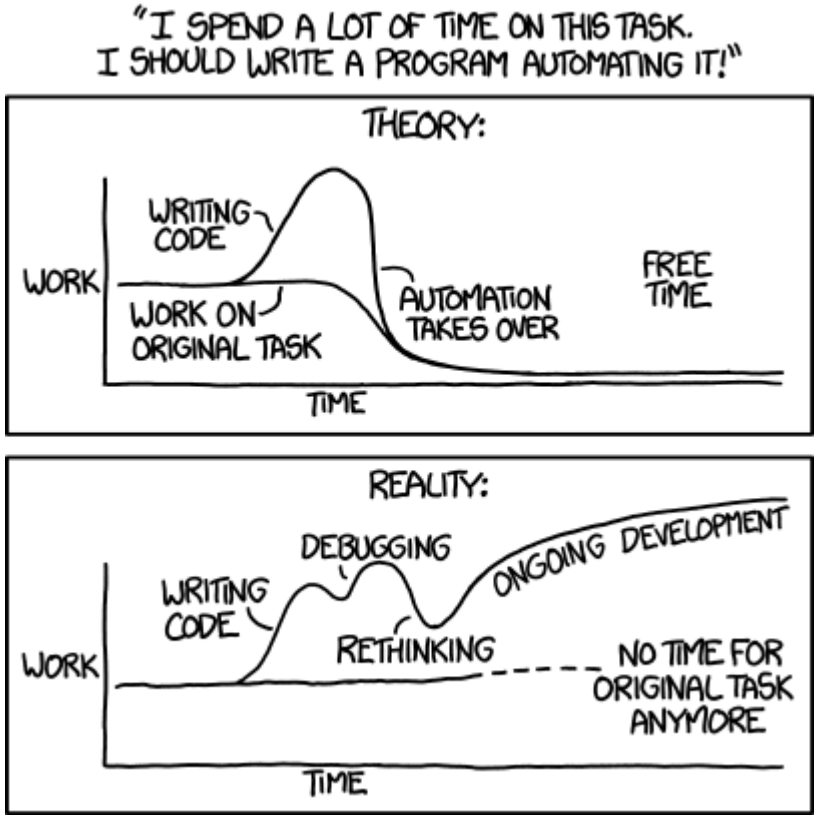
DHBW

Grad der Automatisierung

…wirklich?

DHBW

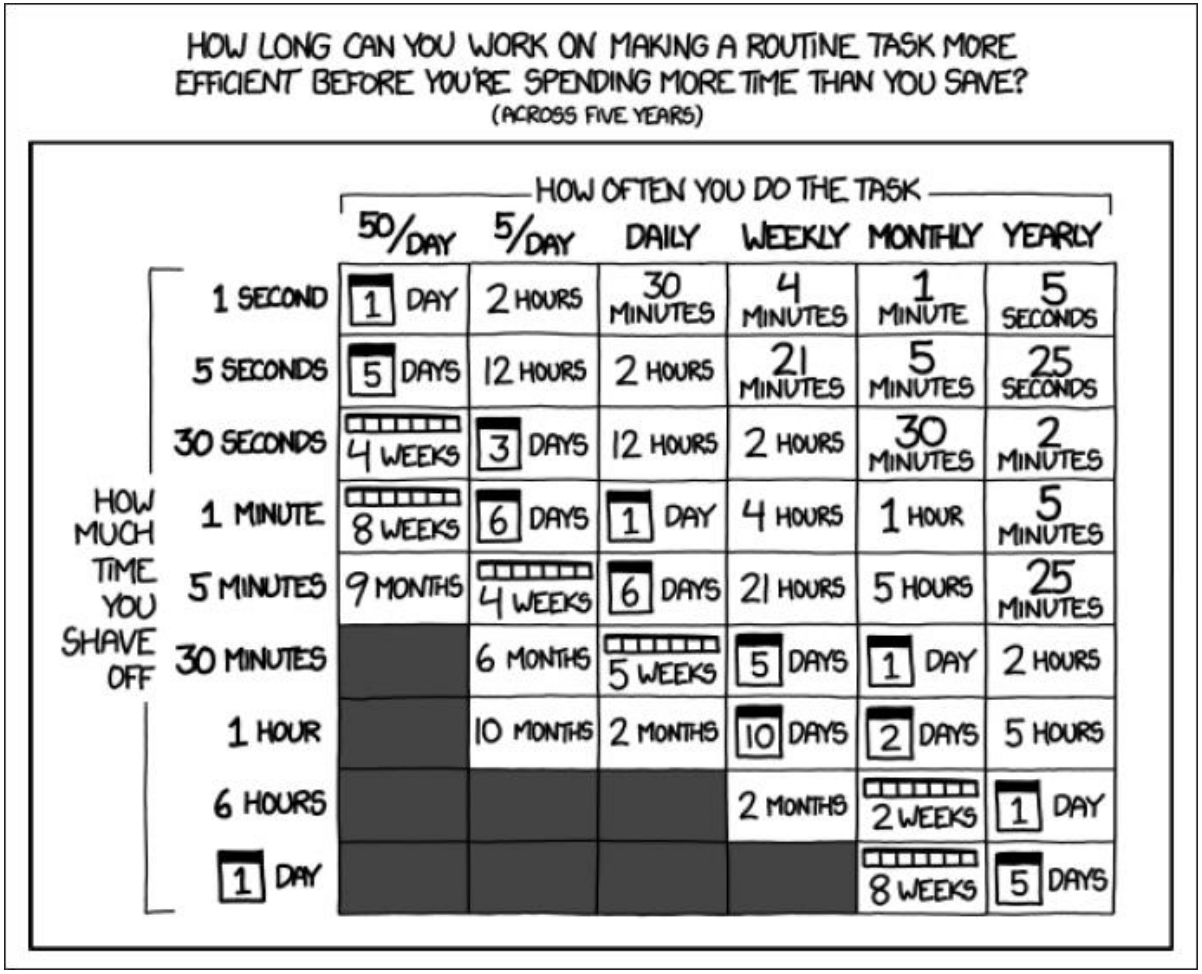
Wann lohnt sich Automatisierung?



Quelle: https://www.xkcd.com/1319/

DHBW

Wann lohnt sich Automatisierung?



Quelle: https://www.xkcd.com/1205/

Wann lohnt sich Automatisierung?

- wenn in relevantem Ausmaß Zeit gespart wird
  - Einbindung von Legacy-Systemen in automatisierte Prozesse kann beispielsweise unverhältnismäßig viel Zeit kosten
- Zeitinvestitionen sind nicht immer vergleichbar
  - Zuverlässiges Rollback-Script ist vermutlich „mehr wert“ als beispielsweise die automatische Generierung eines Change-Logs
- Zeitersparnis kann auch anderen zugutekommen und multipliziert sich dann

You are not Netflix

„ While DevOps tool providers may sell automation products that promise to make you the next Facebook or Netflix, the reality is that your organization isn't like Facebook or Netflix, and it never will be. If you try to replicate their automation solutions, you'll quickly find yourself in a world of hurt. The investment you have made in DevOps and automation will blow up in your face—not because you automated, but because you attempted to automate too much.”

Vgl. http://techbeacon.com/devops-automation-best-practices-how-much-too-much, abgerufen am 02.02.2016

DHBW

Automatisierungs-Paradoxon

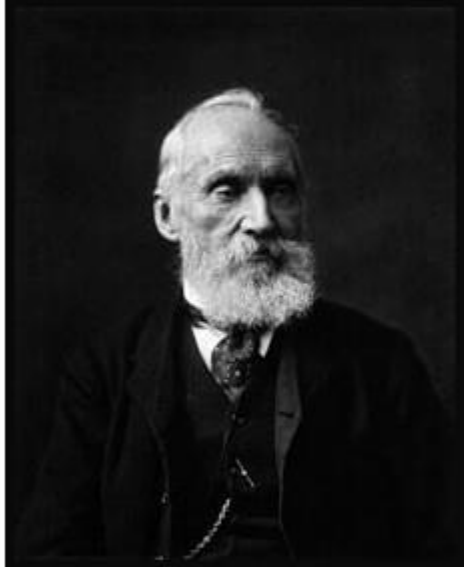
- Je höher der Grad der Automatisierung, desto kompetenter muss der Mensch sein, der den automatischen Prozess verwaltet
- “Efficient Automation makes humans more important, not less”

Automatisierungs-Ironie

- Wenn zu viel automatisiert wird, wird der Operator zum „Beobachter“ des Systems
- Er verlernt die Grundlagen der Prozesse, die automatisiert wurden
- Er kann daher im Fehlerfall nicht mehr angemessen und schnell reagieren
- Je hochgezüchteter die Automatisierung, desto gravierender die Effekte der Automatisierungs-Ironie
- Es kann daher sinnvoll sein, bewusst manche Teilprozesse oder Teilschritte nicht zu automatisieren

DHBW

Metrics



I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science, whatever the matter may be.

(Lord Kelvin)

izquotes.com

DevOps messen

- damit der Erfolg von DevOps überprüft und kontinuierlich verbessert werden kann, muss er **messbar** gemacht werden.
- Neben **klassischen** Metriken für Software-Entwicklung und Betrieb wird bei DevOps großer Wert auf Metriken gelegt, welche die **Leistungsfähigkeit des gesamten Wertstromes** aufdecken

„According to the lean philosophy, it is essential to optimize globally, not locally.“

DHBW

„klassische“ Software-Metriken

- Betreffen nur das Silo „Entwicklung“
- Beispiele:
  - Lines of Code
  - zyklomatische Komplexität
  - Stil
  - Testabdeckung
  - Technische Schuld

DHBW

„klassische“ Operations-Metriken

- Betreffen nur das Silo „Operations“
- Beispiele:
  - Uptime
  - Antwortzeiten
  - Ressourcen-Auslastung

DHBW

DevOps-Metriken

- Betreffen den **gesamten Wertstrom**
- Die vier wichtigsten Metriken:

Durchsatz	**Stabilität**
Häufigkeit von Deployments in Produktion	mittlere Reparaturzeit
Änderungs-Durchlaufzeit	Fehlerrate in Produktion

Vgl. https://puppetlabs.com/2014-devops-report

DHBW

Häufigkeit von Deployments in Produktion

- „Deployment Frequency“
- Wie oft wird in das Produktivsystem deployed
- Relativ einfach messbar
- Ggf. Unterscheidung in # erfolgreicher/nicht erfolgreicher Deployments

Grafik

DHBW

Änderungs-Durchlaufzeit

- „Change Lead Time“
- Wie lange benötigt eine Änderungseinheit vom Beginn der Arbeit bis zum Deployment in Produktion
- Problem: wann „beginnt“ die Arbeit an einer Änderungseinheit?
  - Erfassen der User Story?
  - Schreiben der ersten Code-Zeile?

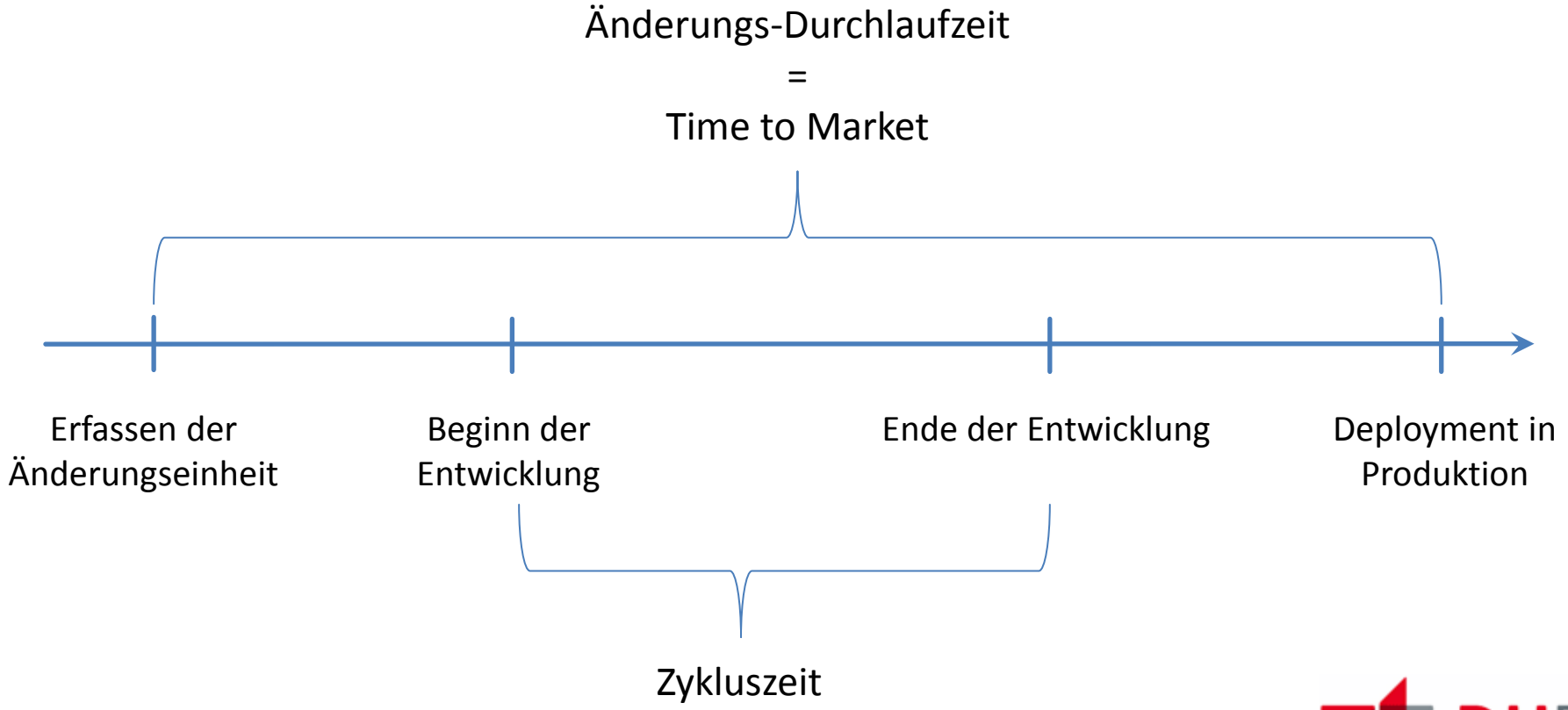
DHBW

Änderungs-Durchlaufzeit

Beispiel:

Änderungs-Durchlaufzeit

Mögliche Verfeinerung:



Änderungs-Durchlaufzeit

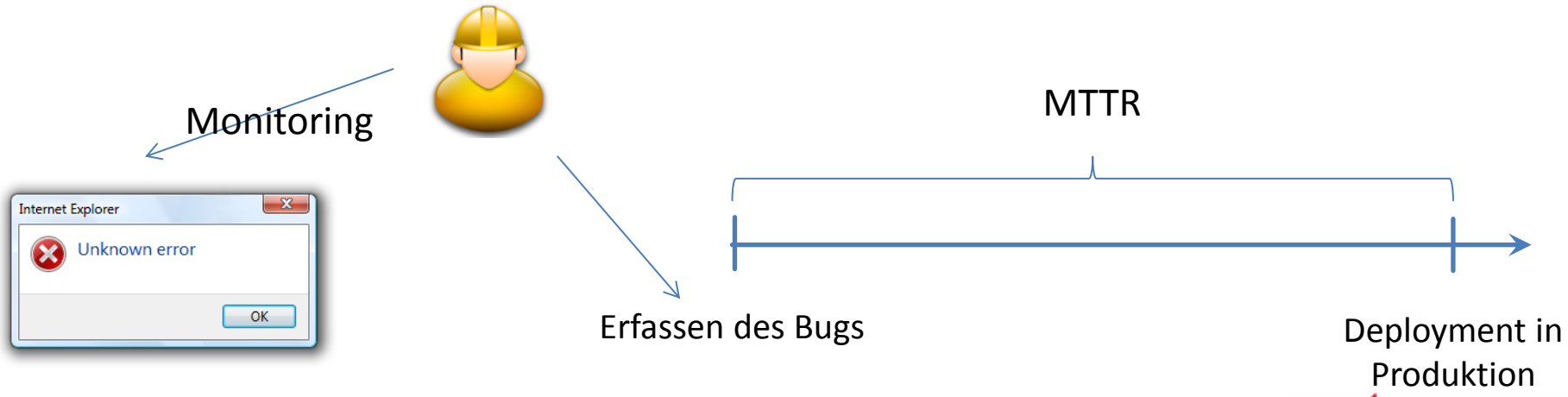
Einige Hinweise:

- Lead Time und Cycle Time werden teilweise synonym benutzt
- Begriffe kommen aber aus der Produktionsorganisation und sind unterschiedlich und klar definiert
- Starten Sie mit dem, was Sie am einfachsten messen können

DHBW

Mittlere Reparaturzeit

- „Mean Time to Recover/Repair“ (MTTR)
- Zeit zwischen Feststellung eines Fehlers in Produktion und dessen Beseitigung
- Beispielsweise messbar als Durchlaufzeit eines Issues vom Typ „Bug“



Fehlerrate in Produktion

- „Change Failure Rate“
- wie viele Bugs treten in Produktion auf
- Einfach messbare Metrik
- Ggf. Unterscheidung nach Schwere
- Evtl. Messung von Reaktionszeit

Zusammenhang zwischen Durchsatz/Stabilität

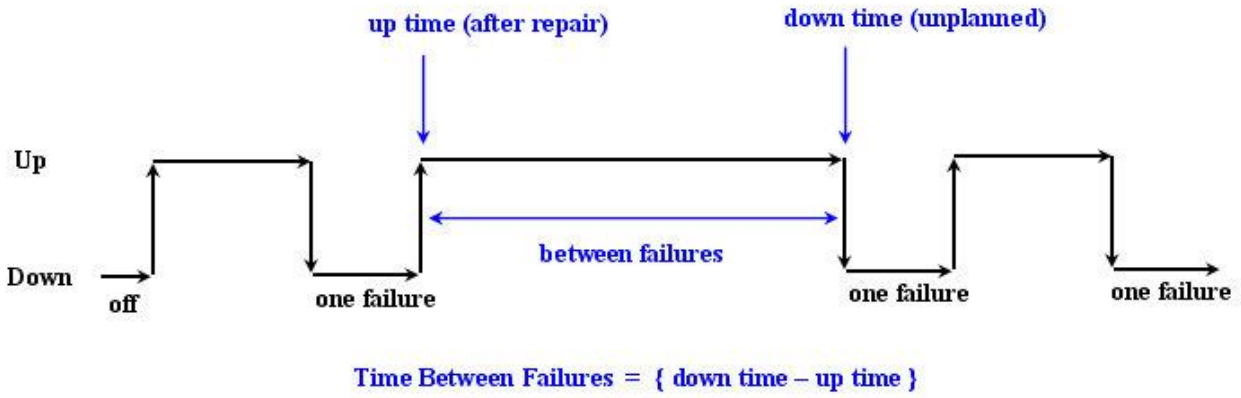
Häufige Deployments und kurze Durchlaufzeiten **können** Stabilität, Fehlerrate und Reaktionszeiten extrem verbessern.

Diskussion

DHBW

Weitere mögliche DevOps-Metriken

- Work in Progress (WIP)
- Verfügbarkeit
- Performance
- Mean Time Between Failures (MBTF)
- Kulturelle Metriken
  - Akzeptanz DevOps, Zufriedenheit, …
- Business-Metriken
  - Conversion Rate, Umsatz, …



MBTF

Exkurs: statsd und Graphite

- https://github.com/etsy/statsd
- http://graphite.wikidot.com/
- http://www.aosabook.org/en/graphite.html
- http://de.slideshare.net/jeremyq/statsd-and-graphite

DHBW

statsd

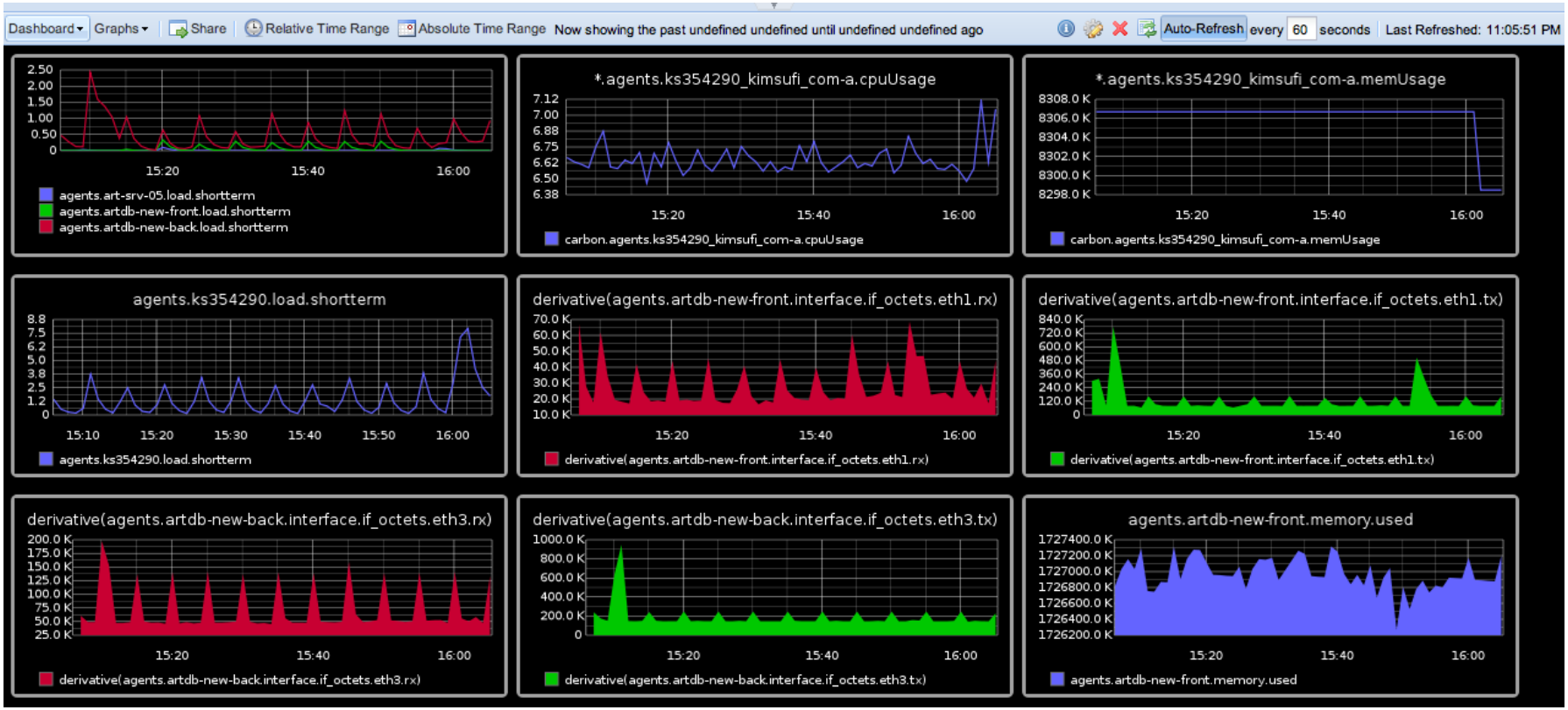
- Node.js-basierter Network-Daemon
- Lauscht per UDP oder TCP auf gesendete Messungen („stats“)
- Sammelt Messungen in „buckets“
- Aggregiert zeitgesteuert (10 sec/default) alle Messungen und sendet diese an irgendeinen Backend-Service, z.Bsp. Graphite

```
echo „myapp.auth.login:1|c" | nc -u -w0 127.0.0.1 8125
```

DHBW

Graphite

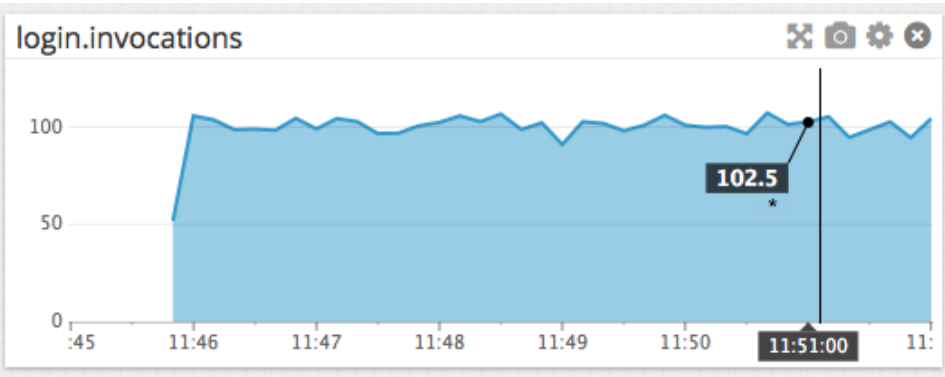
Monitoring-Tool

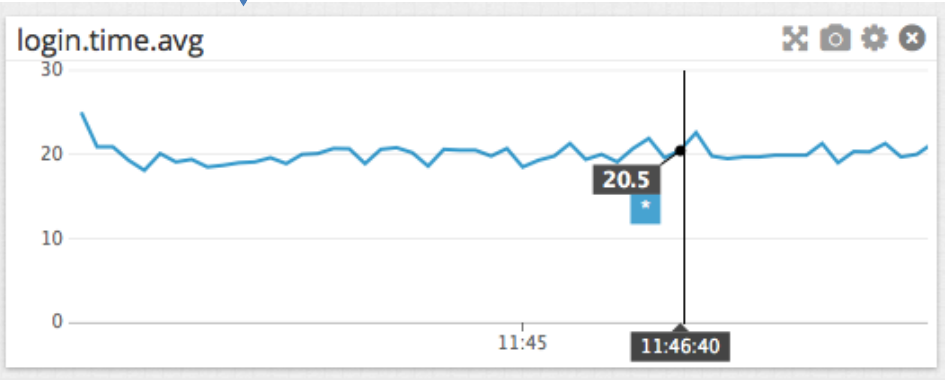


Exkurs: statsd und graphite

```
import statsd
statsd_client = statsd.StatsClient('localhost', 8125)

@statsd_client.timer('login.time')
def login(username, password):
    statsd_client.incr('login.invocations')
    if password_valid(username, password):
        render_welcome_page()
    else:
        render_error(403)
```



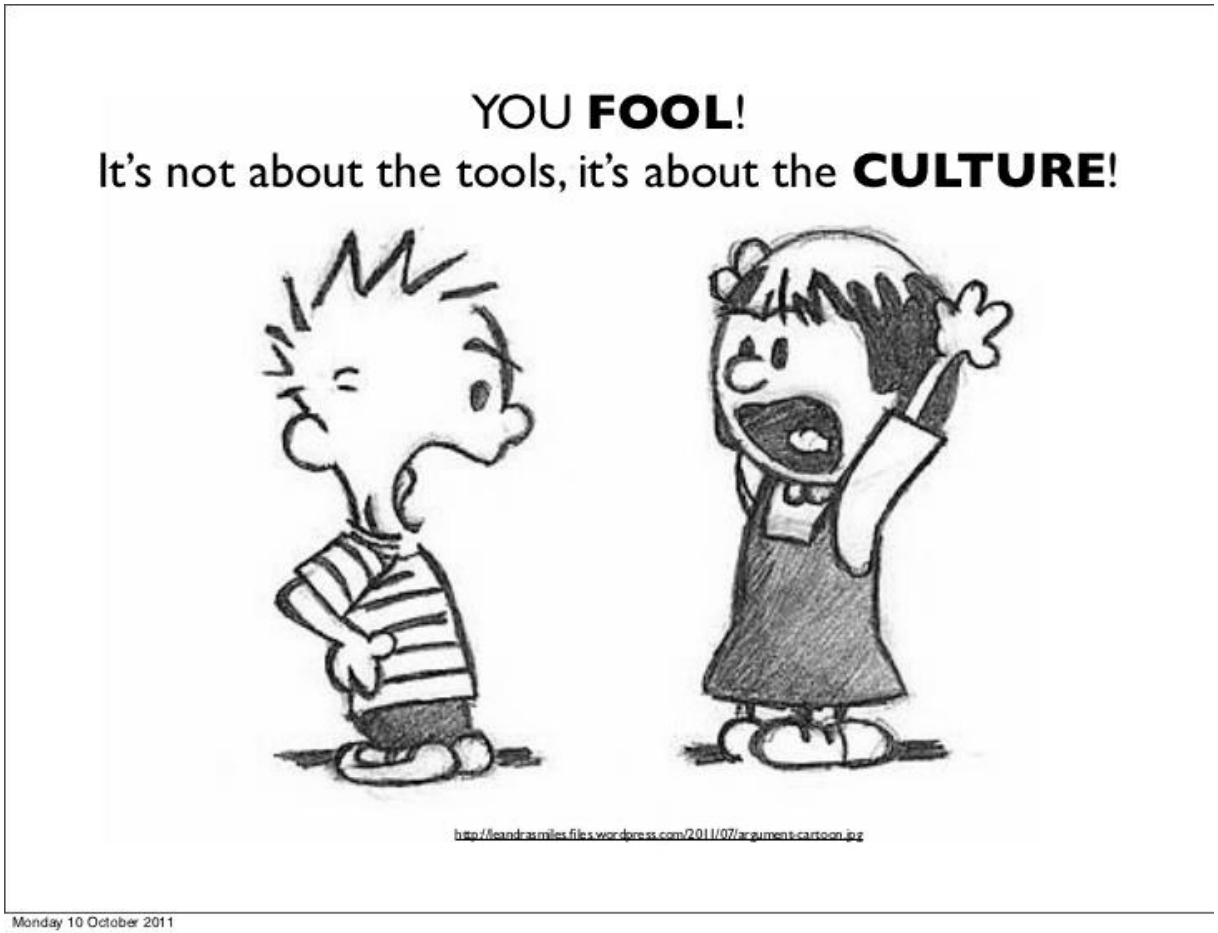


DHBW

Pause

DHBW

Culture


YOU **FOOL**!
It's not about the tools, it's about the **CULTURE**!
http://leandrasmiles.files.wordpress.com/2011/07/argument-cartoon.jpg
Monday 10 October 2011

DevOps-Kultur

Das **Hauptziel** von DevOps ist eine Veränderung der **Organisationskultur** von:

- **Silos** zu **Kollaboration** und
- **Schuldzuweisungen** zu **gemeinsamer Verantwortung** und **Vertrauen**

für die erstellten digitalen **Produkte**, **Leistungen** und der dazu notwendigen **Prozesse.**

DHBW

Was ist „Organisationskultur“?

Drei ausgewählte Definitionen:

„Konzept und Begriff der Kultur wurden aus der anthropologischen Forschung entlehnt. Es existiert kein Konsens über die Bedeutung des Begriffes, so dass auch in der Anwendung im Rahmen der Organisationstheorie teilweise beträchtliche Unterschiede bestehen. “

L. Smircich (1983), Concepts of Culture and Organizational Analysis; Administrative Science Quarterly 28, S. 339–358

„Organisationskultur ist die Sammlung von Traditionen, Werten, Regeln, Glaubenssätzen und Haltungen, die einen durchgehenden Kontext für alles bilden, was wir in dieser Organisation tun und denken.“

J. Marshall und A. McLean (1985): *Exploring Organisation Culture as a Route to Organisational Change*, in Hammond V. (ed), Current Research in Management, pp. 2-20, Francis Pinter, London.

„So machen wir das hier.“

D. Bright und B. Parkin: *Human Resource Management – Concepts and Practices*. Business Education Publishers Ltd., 1997, S. 13.

Vgl. https://de.wikipedia.org/wiki/Organisationskultur, abgerufen am 06.02.2016

DHBW

Was ist „Organisationskultur“?

- Kultur ist ein abstraktes und komplexes Konzept
- Definitionen meist „vage und allumfassend“
- Schwer messbar
- Aber:
  - **Es ist allgemein anerkannt, dass die Kultur einer Organisation Einfluss auf das Erreichen der Organisationsziele hat**

Vgl. https://de.wikipedia.org/wiki/Organisationskultur, abgerufen am 06.02.2016

DHBW

DevOps-Kulturmerkmale

- Gegenseitiger Respekt
- Vertrauen
- No-blame
- Growth mindset
- Gemeinsame Anreize (Incentives)
- Kollaboration

„Organizational culture is one of the strongest predictors of both IT performance and overall performance of the organization. “

State of DevOps Report, 2014, PuppetLabs

DHBW

Blame-Kultur

- „name, blame, shame“
- Bei Fehlern wird so lange gesucht, bis ein Sündenbock gefunden wurde
- Resultat:
  - Fehler werden verschleiert
  - Menschen haben Angst, Hemmungen und sichern sich ab
  - ->Verschwendung

No-Blame-Kultur

- Nicht Individuen führen zu Fehlern, sondern **Situationen**
- Vermeidung der Wiederholung ähnlicher Situationen steht im Vordergrund
- Keine Suche nach einem Sündenbock sondern **Suche nach Ursachen.**

https://www.youtube.com/watch?v=2PjVuTzA2lk

DHBW

5-Why-Methode

- Methode zur Bestimmung von Ursache und Wirkung
- So lange „Warum“ fragen, bis die Ursache klar ist (5 ist symbolisch)

Problemstellung: Das Fahrzeug startet nicht.

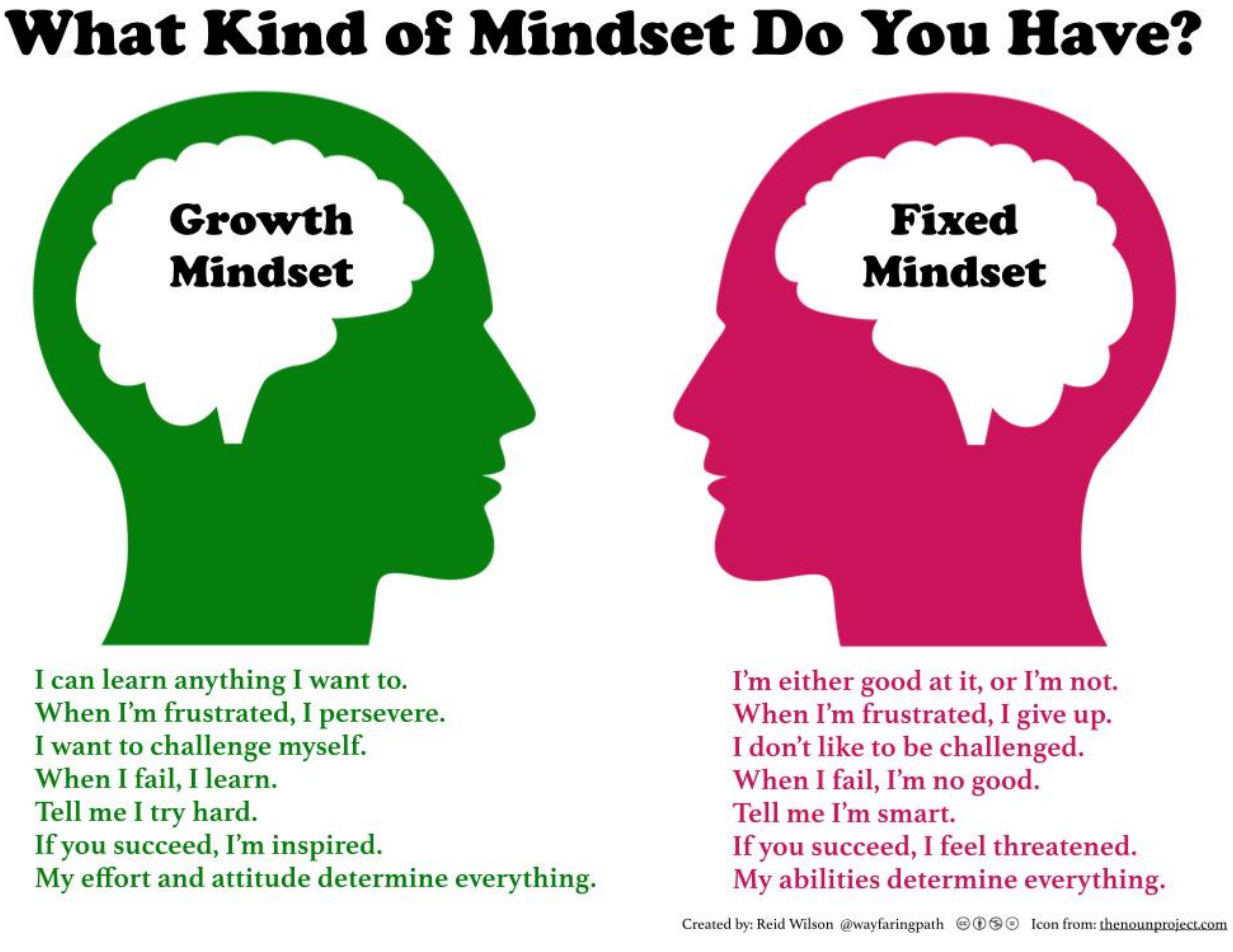
1. Warum startet das Fahrzeug nicht?
   Die Starterbatterie ist leer.
2. Warum ist die Starterbatterie leer?
   Die Lichtmaschine funktioniert nicht.
3. Warum funktioniert die Lichtmaschine nicht?
   Der Treibriemen ist gerissen.
4. Warum ist der Treibriemen gerissen?
   Der Treibriemen wurde nie ausgewechselt.
5. Warum wurde der Treibriemen nie ausgewechselt?
   Das Fahrzeug wurde bisher nie gewartet.

Vgl. https://de.wikipedia.org/wiki/5-Why-Methode, abgerufen am 06.02.2016

DHBW

Fixed Mindest vs. Growth Mindset

Mindest-Theorie nach Carol Dweck



DHBW

Fixed Mindest vs. Growth Mindset

Fixed Mindset:

- Intelligenz, Talente, Fähigkeiten sind gegeben und können nicht verbessert werden
- „Fehler sind Resultat ungenügender Fähigkeiten“
- Tendenz, sich nicht blamieren zu wollen

Fixed Mindest vs. Growth Mindset

Growth Mindset:

- Intelligenz, Talente, Fähigkeiten können verbessert werden
- „Fehler sind eine Gelegenheit, sich zu verbessern“
- Tendenz, auch nach Rückschlägen weiter hart zu arbeiten

Gemeinsame Anreize

- Entwickler werden nicht belohnt, wenn sie Tonnen von Code produzieren
- Operator werden nicht belohnt, wenn alle Server immer verfügbar sind
- Das **Team** wird belohnt, wenn Wert für den Kunden geliefert wird und alle gemeinsam daran arbeiten, den Fluss von Wert zum Kunden zu verbessern

DHBW

Incentives Gone Wrong

Kollaboration

- Art der möglichen Zusammenarbeit zwischen Dev und Ops hängt stark vom Unternehmen und dessen Leistungen ab
  - Bereichsübergreifende Teams
  - Temporärers(!) DevOps-Team
  - Austausch von Experten
  - „Infrastructure as a Team“

Vgl. http://blog.matthewskelton.net/2013/10/22/what-team-structure-is-right-for-devops-to-flourish/, 21.02.2016

Kollaboration

DHBW

Share

DHBW

Teilen

- Verantwortung
- Ziele
- Tools
- Information
- Wissen & Erfahrung
- Code
- Workflow
- Training

DHBW

Know-How

- "To be a master of anything, you must understand two layers above and two levels below the layer you're targeting.„
- „…ein guter Entwickler [sollte] verstehen, welchen Einfluss seine Applikation auf die Ebenen oberhalb *(und unterhalb – Anm. d. Doz.)* seines Tätigkeitsbereichs hat.„

  – http://www.heise.de/developer/artikel/Operations-heute-und-morgen-Teil-2-DevOps-im-Jahre-2015-2700954.html?artikelseite=3

DHBW

Kritik

- Hype(r) Hype(r)!
- „NoOps“->Abschaffung der Operator
- Sicherheit wird zugunsten von Geschwindigkeit vernachlässigt
- Manchmal ist Stabilität wichtiger als Geschwindigkeit
- Für manche Architekturstile nicht geeignet (bspw. komplexe Mainframe-Anwendungen)
- Für manche Anforderungen nicht geeignet (bspw. medizinische Anwendungen, Anlagensteuerung)
- Sicherheit wird zugunsten von Geschwindigkeit vernachlässigt
- Nicht jedes Unternehmen ist facebook, google oder netflix

Fazit

„Branchengrößen wie Facebook, Flickr, Twitter oder Amazon gelten als Wegweiser für DevOps. Doch deren Vorgehen dürfen andere Unternehmen nicht einfach nachahmen, denn sie besitzen eine andere Organisation, Firmenkultur oder Softwareausstattung. Stattdessen sollten sie bei der Planung von DevOps-Prozessen mit dem wichtigsten Ziel beginnen, der schnelleren Auslieferung von Software, ohne dabei die Qualität zu gefährden. Das müssen die Entwicklungs-, Test- und Betriebsteams gemeinsam verinnerlichen. Tools können sie dabei unterstützen, die Aufgaben entlang der Prozesskette effizienter zu automatisieren.“

– http://www.heise.de/developer/artikel/Mehr-Qualitaet-und-Geschwindigkeit-bei-DevOps-2859707.html, abgerufen am 15.01.2016

DHBW

Fazit

„Die größte Aufgabe für IT-Verantwortliche ist es herauszufinden, in welchen Teilen des Unternehmens ein DevOps-Einsatz am sinnvollsten wäre. Dies wird vermutlich dort sein, wo Geschwindigkeit eine wichtige Rolle spielt und das Unternehmen Möglichkeiten sieht, seinen Kunden ein besseres Erlebnis zu bieten als seine Wettbewerber (z.B. ein Einzelhändler, der den Bezahlvorgang auf seiner Website mit Hilfe von DevOps verbessert, oder eine Bank, die auf ihrer Website neue Möglichkeiten anbietet, um die Entwicklung von Fonds zu verfolgen).“

– http://www.cio.de/a/devops-in-einer-two-speed-architektur,3251208,3, abgerufen am 17.01.2016

DHBW

Fazit

„Der Wechsel zu einer DevOps-Umgebung kann sowohl die Produktivität als auch die Markteinführungszeit von Softwareprodukten deutlich verbessern. Mit der Einführung einer neuen IT-Methode ist es dabei allerdings nicht getan. Gefragt ist eine unternehmensweite Transformation, bei der Prozess- und Governance-Fragen genauso Rechnung getragen wird wie Technologiethemen.“

– http://www.cio.de/a/devops-in-einer-two-speed-architektur,3251208,4, abgerufen am 06.02.2016

DHBW

Fazit

- Ihr Fazit?

DHBW

Ausblick

- DevSec
- IT der zwei Geschwindigkeiten

DHBW

DevSec

- Agile Methoden führen zur Vernachlässigung der Sicherheit aufgrund schnellerer Release-Zyklen
- NSA-Skandal usw. hat Problembewusstsein geschärft
- **„bei DevSec [geht es] darum, das Sicherheitsdenken und entsprechende Praktiken fest in der Entwicklung und den zugehörigen Prozessen zu verankern.“**

Vgl. http://www.heise.de/developer/artikel/Auf-DevOps-folgt-der-Bedarf-an-DevSec-3016131.html, abgerufen am 04.02.2016

DHBW

IT der zwei Geschwindigkeiten

- „bimodale IT“, „Hybrid-IT“
- Die Interne IT gilt in Unternehmen oft als Innovationshemmer
- Unternehmen brauchen zwei verschiedene Arten von IT:
  - Traditionelle IT (Stabilität für Tagesgeschäft)
  - Agile IT (Innovation und Geschwindigkeit)

DHBW

IT der zwei Geschwindigkeiten

Bimodal IT = Marathon Runners + Sprinters. Deeply Different, Both Essential

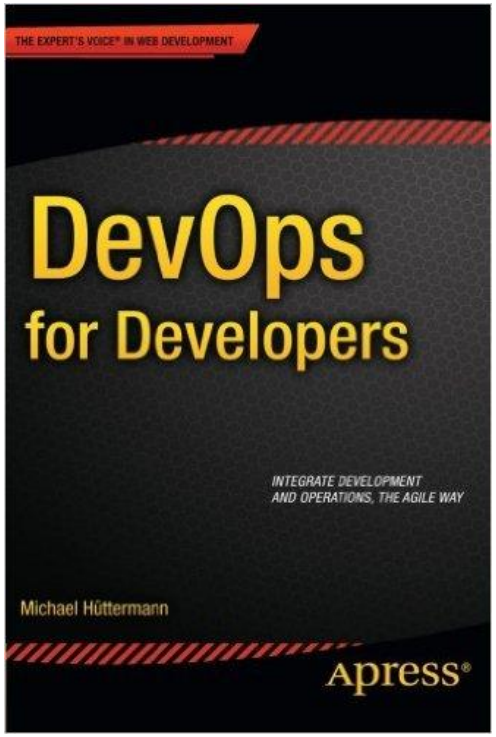
Think Marathon Runner

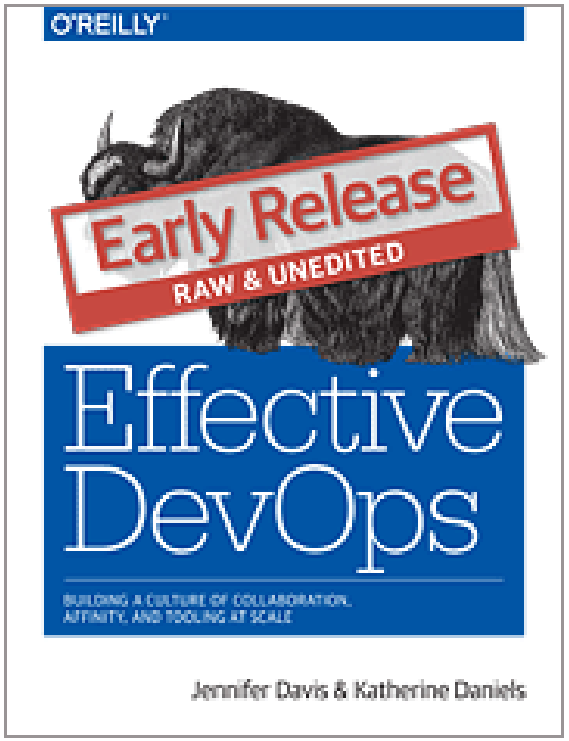
Mode 1		Mode 2
Reliability	**Goal**	Agility
Price for performance	**Value**	Revenue, brand, customer experience
Waterfall, V-Model, high-ceremony IID	**Approach**	Agile, kanban, low-ceremony IID
Plan-driven, approval-based	**Governance**	Empirical, continuous, process-based
Enterprise suppliers, long-term deals	**Sourcing**	Small, new vendors, short-term deals
Good at conventional process, projects	**Talent**	Good at new and uncertain projects
IT-centric, removed from customer	**Culture**	Business-centric, close to customer
Long (months)	**Cycle Times**	Short (days, weeks)

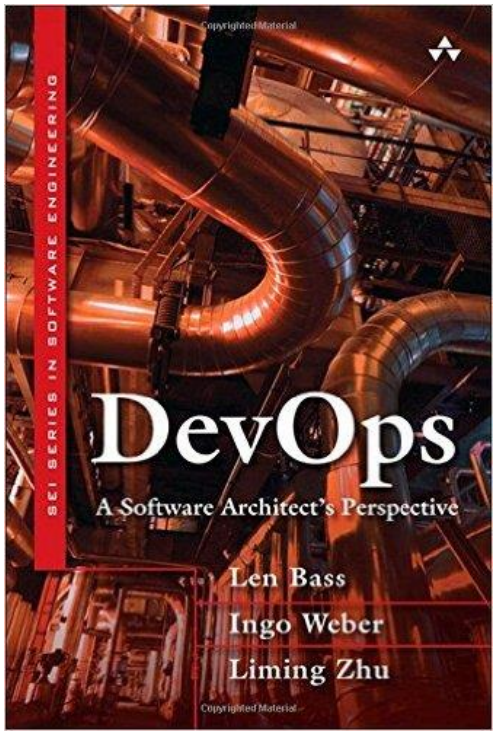
Think Sprinter

Gartner

Literatur

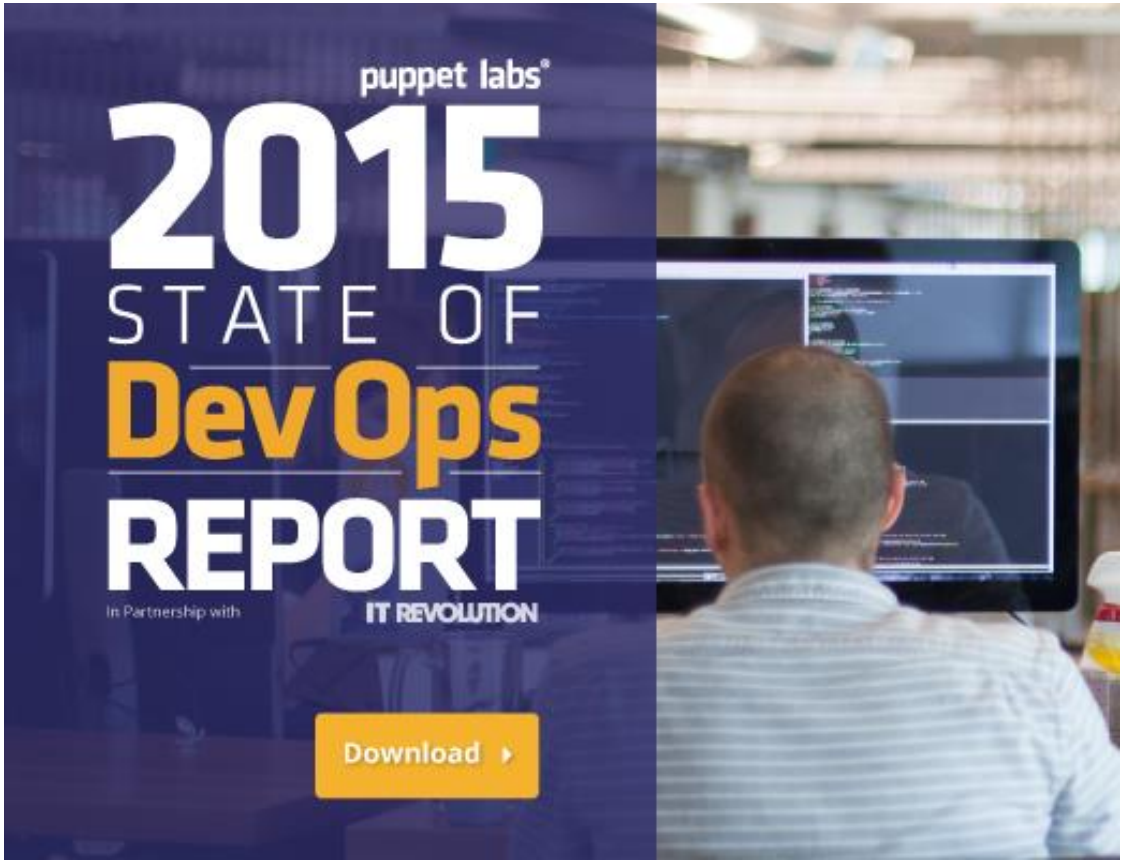




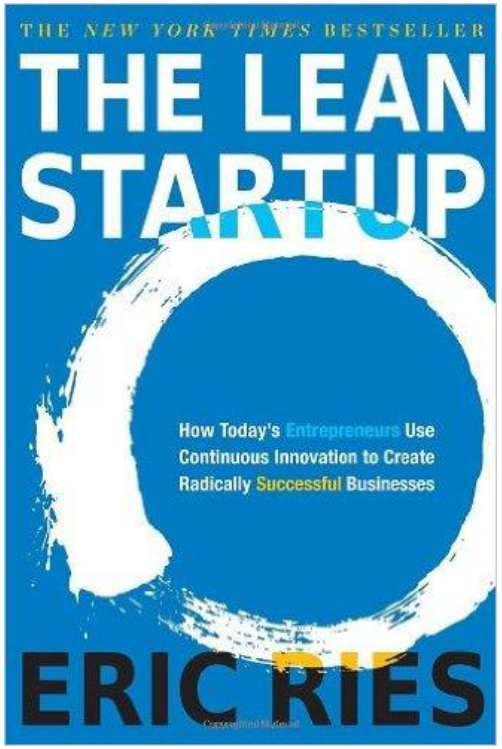


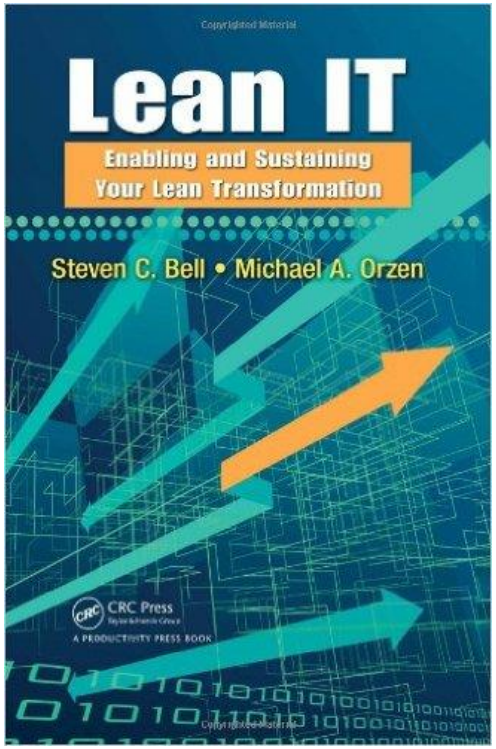
DHBW

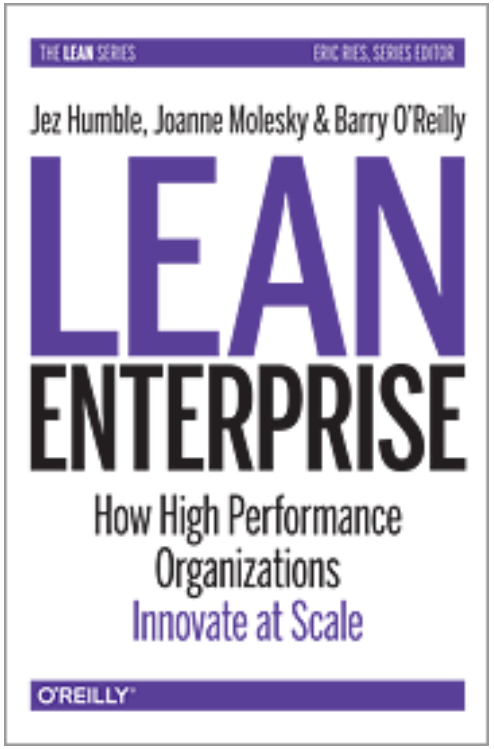
Literatur


puppet labs
2015
STATE OF
DevOps
REPORT
In Partnership with
IT REVOLUTION
Download

Literatur







DHBW

BestOf: DevOps Borat

- To make error is human. To propagate error to all server in automatic way is #devops.
- In Agile is all about fail quick. If you have success in company you do Agile wrong.
- If shit is never hit fan you are not get enough shit done.
- Pair programming is good concept but is involve too many programmer.
- In startup we are allow all developer use 20% of time for write code with not TDD so they can able of actual get shit done.
- In startup we are hire only rockstar, ninja or samurai. Vague familiar with computer is big plus.
- Is not in production until you are receive critical alert of it.
- For minimum viable product we are recommend of start with Hello world. #leanstartup
- In startup we are gamify site outage. Three outage and ops team is out.
- Best decision we ever make in startup was outsource all site outage to Cloud provider.
- Goal of sysadmin is replace itself with small shell script. Goal of devops is replace itself with small REST API.
- Hello World in cloud is involve 1 load balancer, 3 web server and 2 database server.
- In startup we are practice Outage Driven Infrastructure.

DHBW