

# Praktiken als Voraussetzung zur Verkürzung von Releasezyklen qualitativ hochwertiger Software

Claus Polanka Bernhard Fleck

Software Quality Management Wintersemester 2011/12

Institute of Software Technology and Interactive Systems
Quality Software Engineering
http://qse.ifs.tuwien.ac.at

# Übersicht



- Best Practices zur Verkürzung von Releasezyklen
  - Jahr auf Quartal
  - Quartal auf Monat
  - Monat auf Woche
  - Woche auf Tag
- Success Stories aus dem Bereich Continuous Deployment

  - Huitale
  - Digg
  - Flickr

# I TU

# Kategorisierung der Best Practices

- Organisatorische Aspekte
  - Wie müssen Arbeitsabläufe angepasst werden?
  - Gibt es organisatorichen Ballast?
- Werkzeug- und Methodenunterstützung
  - Was für Technologien können eingesetzt werden?
  - Welche Methoden realisieren diese Technologien?
- Geschäftliche Aspekte
  - Wie müssen Geschäftsmodelle angepasst werden?





noch keine

#### Werkzeug- und Methodenunterstützung

- Automatisierte Akzeptanztests
- Refactoring
- Continuous Integration

#### Geschäftliche Aspekte

Subscription Modell





- Programmierer schreiben Tests
- Status Meetings
- Task Board
- keine QA-Abteilung
- kein Change Management

#### Werkzeug- und Methodenunterstützung

- keine mehrfach releasten Versionen
- keine separaten Design Dokumente
- keine separaten Build- und Analyse-Teams

#### Geschäftliche Aspekte

Pay-per-Use Modell





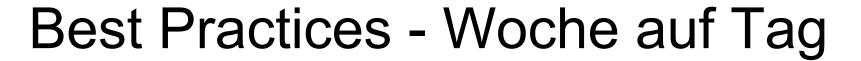
- Temporäre Branches
- Kanban
- kein separates Test Team
- keine Up-front Usability

#### Werkzeug- und Methodenunterstützung

- Keystoning
- One-Button-Deploy
- kein aktiver Release-Branch

#### Geschäftliche Aspekte

keine





- One-Piece-Flow
- kein Operations Department
- keine Status Meetings

#### Werkzeug- und Methodenunterstützung

- Immunization
- Data-Informed-Usability
- Feature-Flags
- kein Multi-Level-Staging

#### Geschäftliche Aspekte

keine















- Instant Messaging Virtual Universe
- Online Social Entertainment Website
- Lean Startup Unternehmen
- Hohe Testautomatisierung (~15.000 Tests)
- ~50 Deployments pro Tag

"This is a software release process implementation of the classic Fail Fast pattern. The closer a failure is to the point where it was introduced, the more data you have to correct for that failure." [1]





- Dienstleistungen im Haushaltsservicebereich
- Lean Startup Unternehmen
- setzen Kanban-Elemente erfolgreich ein
- keine Vollzeitentwickler





- Social News Online-Plattform
- Continuous Deployment ab Version 4
- setzen Gerrit (Code Review System für Git) für verpflichtende Code Reviews ein

"Does this mean we're never going to introduce bugs to our live site? Of course not - but we're going to keep the number of bugs to hit the live site to a minimum, and we've made it easy and fast to get bug fixes live as well." [2]





- Foto und Video Hosting Plattform
- Hohe Verantwortung der Entwickler
  - Entwickler sind Release Manager
  - jeder Entwickler kann über Tools in kurzer Zeit eine neue Version deployen
- Staging
- verwenden Feature Flags

"This style of development isn't all rainbows and sunshine. (...) But overall, we find it helps us develop new features faster and with fewer bugs." [3]

# Hauptvorteile



- Regression wird sehr rasch erkannt
- Fehler können rasch behoben werden da zwischen dem einspielen eines Fehlers und der Meldung über ein Problem nicht viel Zeit vergeht
- Release einer neuen Version erzeugt keinen zusätzlichen Overhead (Non-Events)
- Unmittelbares Feedback
- Feedback besteht aus sofort messbaren Kerndaten von echten Kunden



Vielen dank für eure Aufmerksamkeit!

# References



[1] Timothy Fitz. Continuous Deployment. 2009-02-08. http://timothyfitz.wordpress.com/2009/02/08/continuous-deployment/

[2] Andrew Bayer. Continuous Deployment, Code Review and Pre-Tested Commits on Digg4. 22. Juli 2010. http://about.digg.com/blog/continuous-deployment-code-review-and-pre-tested-commits-digg4

[3] Ross Harmes. Flipping Out. 2. Dez. 2009. http://code.flickr.com/blog/2009/12/02/flipping-out/