Rolling Stones – vom Release überrollt

CONSULTANT

ELMAR DOTT

(M. Schulz) studierte an der HS Merseburg Diplominformatik und twittert regelmäßig über alle möglichen technischen Themen. Seine Schwerpunkte sind hauptsächlich Build und Konfiguration Management, Software Architekturen und Release Management.

Seit über fünfzehn Jahren realisiert er in internationalen Projekten für namhafte Unternehmen umfangreiche Webapplikationen. Er ist freier Consultant / Trainer. Sein Wissen teilt er mit anderen Technikbegeisterten auf Konferenzen, wenn er nicht gerade wieder einmal an einem neuen Fachbeitrag schreibt. https://elmar-dott.com

+ Consultant + Writer + Speaker + Trainer +

AGENDA

- Begriffe & Bedeutungen
- DevOps und Release Management
- Semantic Versioning
- Sprints & Agilität
- DevOps Release & Deploy Strategien

BEGRIFFLICHKEITEN

Release: (dt. : loslassen) bezeichnet also einen definierten Zustand eines Artefaktes, dessen Entwicklungszyklus abgeschlossen wurde, so dass es zur Verwendung freigegeben ist.

Deliver: (dt. : Auslieferung) bezeichnet die Bereitstellung der Artefakte zur weiteren Verwendung, z. B. durch die Installation auf einem Repository Server.

Deploy: (dt. : Einsatz) meint die Installation einer Software in die Zielumgebung, vornehmlich nach PRODUKTION.

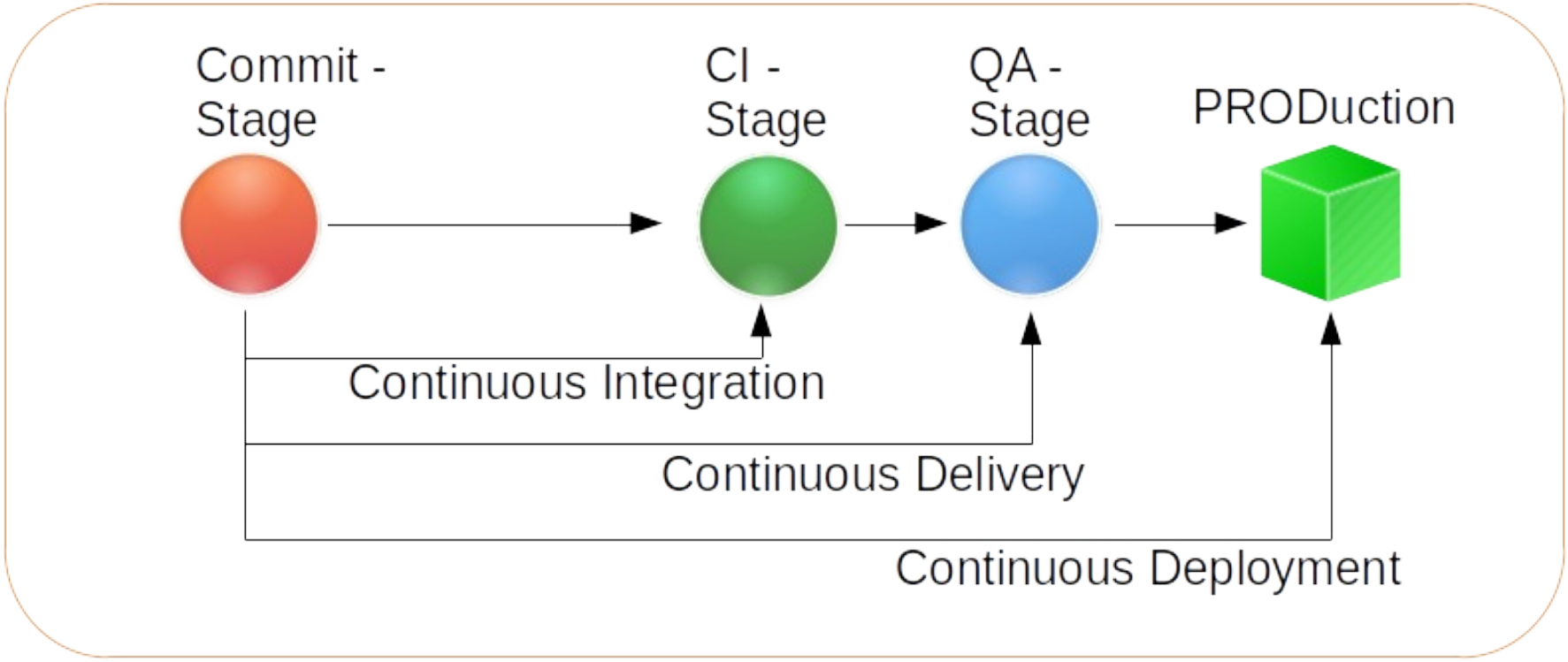
PROZESSKETTE

Release : package

Deliver : storage

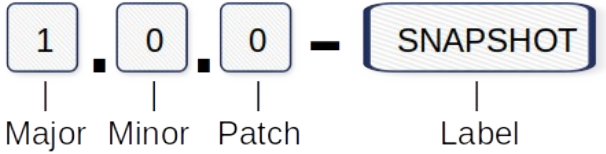
Deploy : install

AUTOMATISIERUNGSSTUFEN



SEMANTIC VERSIONING

Erhöhen der Versionsnummer [MAJOR . MINOR . PATCH]:



1. **MAJOR** – inkompatible API Änderungen
2. **MINOR** – Hinzufügen neuer Funktionalität, die rückwärts kompatible ist.
3. **PATCH** – Fehlerkorrekturen, die rückwärts kompatibel sind

Optional existieren Auszeichner (Labels), um Pre-Releases zu markieren.

Mythen & Missverständnisse

- Versionierung ganzer Anwendungen kann individuell sein (vergl. Firefox)
- SemVer zwingend für Module, Bibliotheken & Microservices (Integration)
- Versionsnummern gehören immer unveränderlich in das Artefakt!
- API Änderungen zusammen fassen und langfristig planen
- Build Nummern haben in Versionsnummern nix verlohren

src/main/resources/org/europa/together/configuration/**version.properties**

```
01: version=${project.version}
```

src/main/java/org/europa/together/utils/**Constraints.java**

```
029: public static final String MODULE_VERSION = getVersion();

099: private static String getVersion() {
100:     String filePath
101:         = "org/europa/together/configuration/version.properties";
102:     PropertyReader propertyReader = new PropertyFileReader();
103:     propertyReader.appendPropertiesFromClasspath(filePath);
104:     return propertyReader.getPropertyAsString("version");
105: }
```

Source: https://github.com/ElmarDott/TP-CORE

Reproduzierbarkeit & Nachvollziehbarkeit

Reproduzieren bedeutet ein bereits erzieltes Ergebnis unter gleichen Bedingungen belibig oft erneut und ohne Abweichungen zu erhalten.

→ **Wiederholung**

Nachvollziehen heist den Weg wie ein Ergebnis erziehlt wurde vollständig zu erfassen, um diesen bei Bedarf anpassen zu können.

→ **Verständnis**

FEHLERKLASSEN

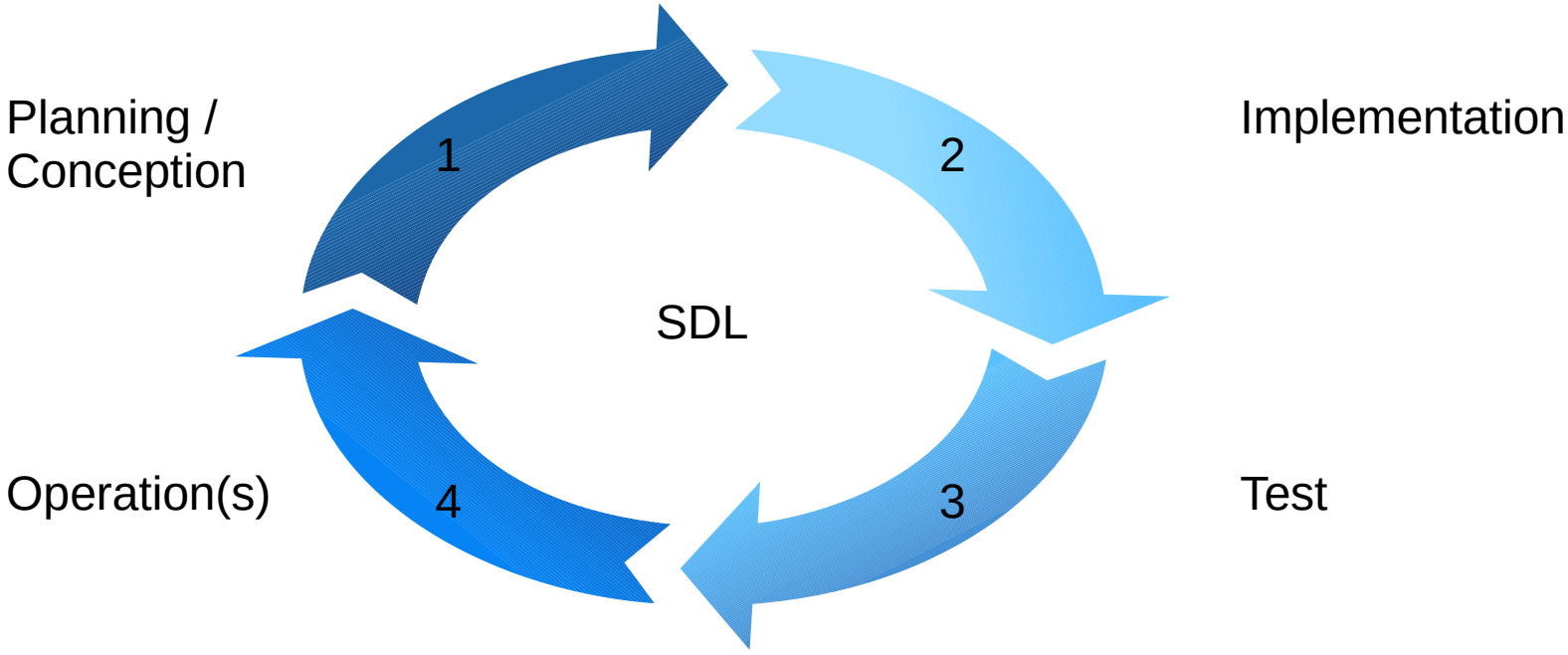
Fehlerklassen die eine Verwendung des Releases verhindern können:

- Einfache Fehler (GUI/ Layout/ Beschriftung)
- Mittlere Fehler (‚einfache‘ Logikfehler)
- Schwere Fehler (Abnahme verhindernd)

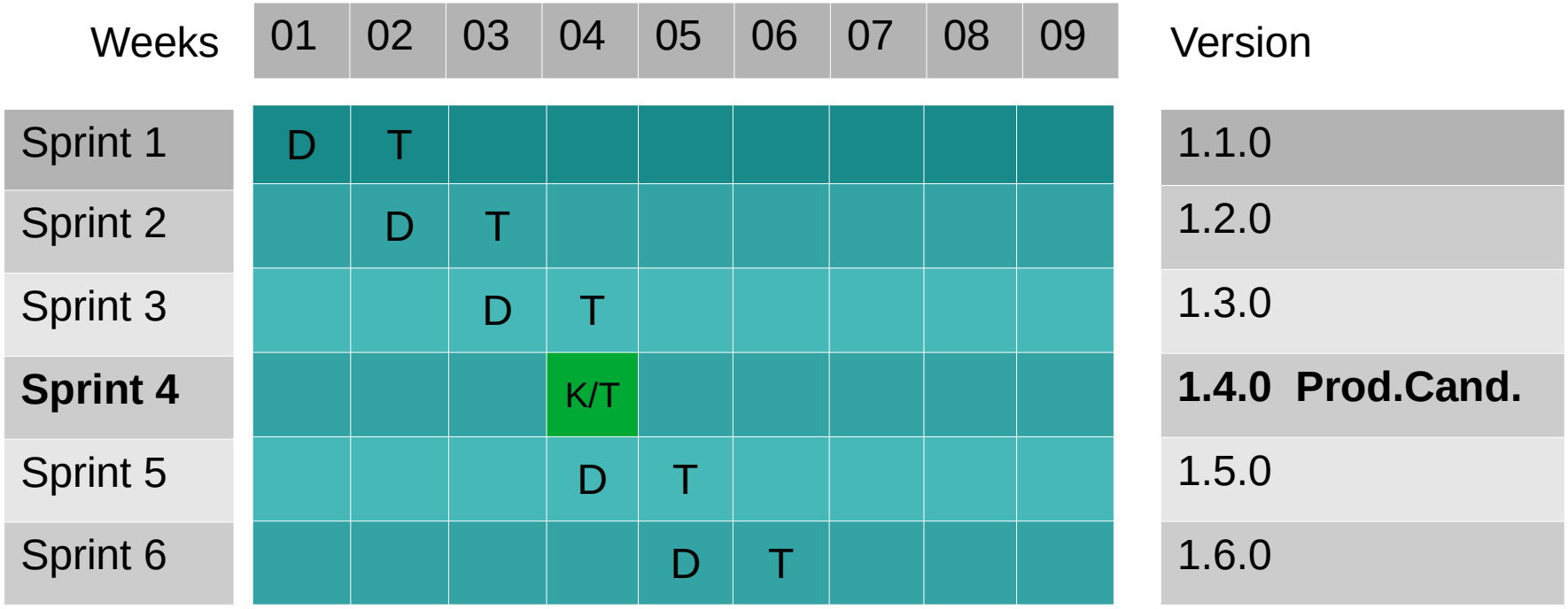
~~Release Candidate~~ → **Production Candidate**

Sprint vs. Release

1 Iteration = 1 Sprint = *n* Weeks = Release



PROJEKTPLAN



D = Develop / T= Test / K = Korrections

TESTSTAGES

DEVELOP

Line A: 1.1.0

Hourly Builds (Commit Stage)
Permanent compile and Unit Tests.

INTEGRATION

Line A: 1.1.0

Nightly Build (Integration & Deploy Tests)
The application is running. Preparing for Release

ACCEPTANCE

Line A: 1.1.0

External Quality Check (Retest)
Released – BugFixes

PRODUCTION

Line A: 1.1.0

Live System
Released – only critical BugFixes

RELEASE TYPES & DEPLOY STRATEGIEN

Blue/ Green Deployment: Redundantes Deployment mit Rollback Option.

Canary Releases: stufenweises Roll-out

A/B Tests: Beurteilung der Conversion (Nutzerakzeptanz) nach festgelegten Kriterien

Feature Toggels: Ein & Ausschalten von Funktionalität.

REFERENZEN

[1] Marco Schulz, 2021, Continuous Integration mit Jenkins, Rheinwerk, ISBN: ISBN 978-3-8362-7834-8 https://www.rheinwerk-verlag.de/continuous-integration-mit-jenkins/

[2] https://www.youtube.com/watch?v=RVCLIP7vOzo

[3] Sematic Versioning: https://semver.org

[3] https://dzone.com/articles/version-number-anti-pattern

[4] https://elmar-dott.com/articles/modern-times/

[5] https://elmar-dott.com/articles/automatisierungs moglichkeiten-im-software-konfigurations-management/

CREDENTIALS


```
-------------------------------------------------------------
   Homepage      : https://elmar-dott.com
   GitHub        : https://github.com/ElmarDott
   AnchorFM      : https://anchor.fm/elmar-dott
   Twitter       : https://twitter.com/ElmarDott
   Speaker Deck  : https://speakerdeck.com/elmardott

   Lbry          : https://lbry.tv/@elmar.dott:8
   BitChute      : https://www.bitchute.com/channel/3IyCzKdX8IpO/
-------------------------------------------------------------
```

Danke / tank you / Gracias

open source

ubuntu MATE

Libre Office
The Document Foundation

BITCHUTE

HECHO EN MÉXICO