






NERZ e.V.
Nutzer der einheitlichen Rechnerzentralensoftware

NERZ Kolloquium 2018 - Mainz

Das Konfigurationsmanagement des NERZ e.V.

FTB NERZ
Dipl.-Ing. H. C. Kniß

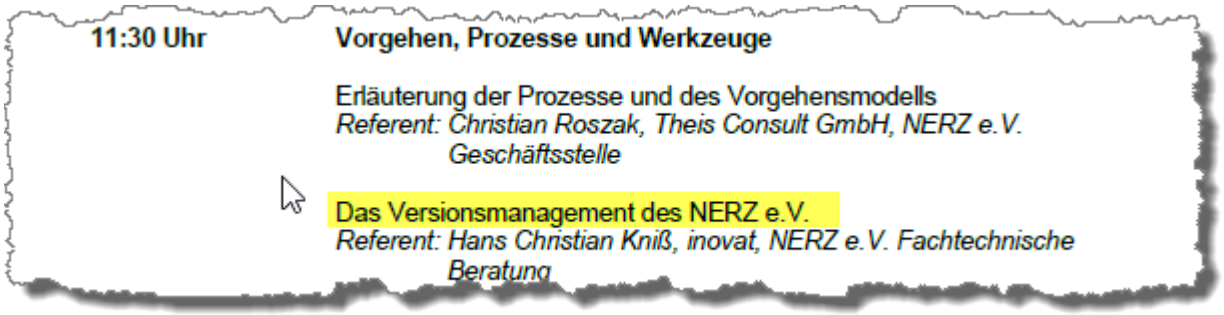
Version 1.2

Das Konfigurationsmanagement des NERZ e.V.

Überblick

- Versionsmanagement vs. Konfigurationsmanagement
- Zielsetzung
- bisherige Vorgehensweise ohne KM
- aktuelle Vorgehensweise mit KM
- aktueller Stand
- Beispiel Aufbau gradle.build und Erstellung eines Distributionspaketes
- Erfahrungen mit dem KM
- Zusammenfassung

Versionsmanagement (Definition)***



Versionsverwaltung

- Eine **Versionsverwaltung** ist ein System, das zur Erfassung von Änderungen an Dokumenten oder Dateien verwendet wird. Alle Versionen werden in einem Archiv mit Zeitstempel und Benutzerkennung gesichert und können später wiederhergestellt werden. Versionsverwaltungssysteme werden typischerweise in der Softwareentwicklung eingesetzt, um Quelltexte zu verwalten.

*** aus Seite „Versionsverwaltung“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 28. Mai 2018, 19:29 UTC. URL: https://de.wikipedia.org/w/index.php?title=Versionsverwaltung&oldid=177832195

Konfigurationsmanagement (Definition)***

Konfigurationsmanagement (**KM**)

- ist eine Managementdisziplin, die organisatorische und verhaltensmäßige Regeln auf den Lebenslauf eines Produkts und seiner *Konfigurationseinheiten* von Entwicklung über Herstellung und Betreuung bis hin zur Entsorgung anwendet.

Konfigurationseinheit

- bedeutet in diesem Zusammenhang eine „beliebige Kombination aus Hardware, Software oder Dienstleistung“.
- Konfigurationsmanagement geschieht innerhalb eines sogenannten *Konfigurationsmanagement-Prozesses* (KMP), dieser bedarf der *Organisation und Planung* (KMO).

*** aus Seite „Konfigurationsmanagement“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 21. Juli 2018, 15:37 UTC.
URL: https://de.wikipedia.org/w/index.php?title=Konfigurationsmanagement&oldid=179334061

Zielsetzung

Zielsetzung

Ziele

- **einfache Bereitstellungsmöglichkeit** des aktuellen Quellcodes der Softwaremodule, Konfigurationen, aktueller Distributionspakete und der Dokumentation für **Entwickler** und Anwender

- **Versionierung**

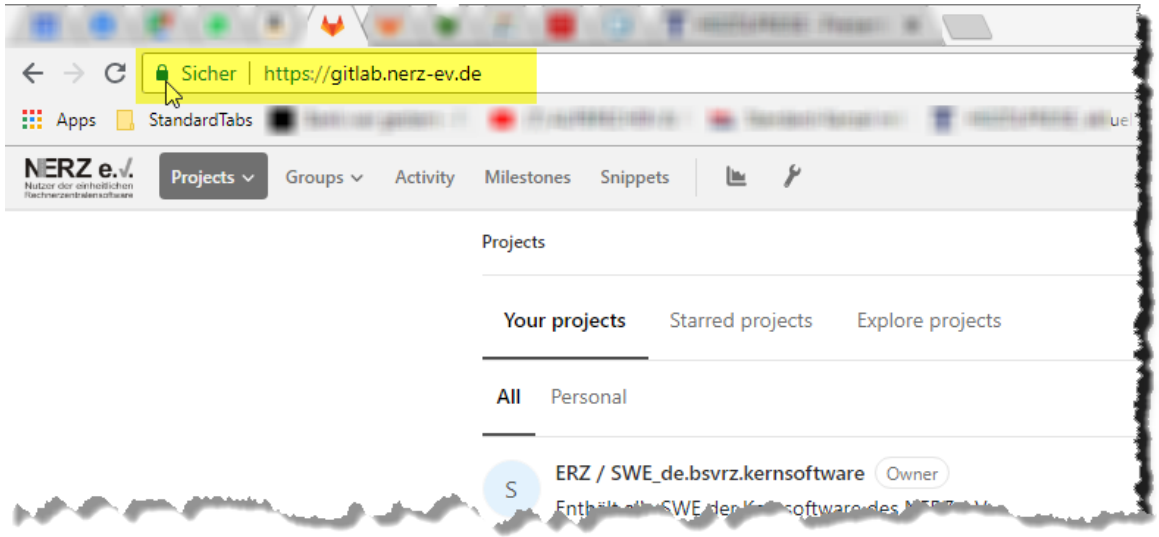
- **Erstellung der Distributionspakete etc.**

- **Verbesserte Weiterentwicklung, Behebung von Bugs, schnellere Verfügbarkeit**

Zielsetzung

Einfache Bereitstellungsmöglichkeit

- Alle
  - Zugriff auf das GitLab des NERZ e.V.
  - Zugriff auf alle Daten (lesend) ohne Anmeldung / ohne Benutzerkennung möglich (OpenSource)
- Entwickler
  - erhalten Benutzerkennung
    - Nachvollziehbarkeit von Änderungen, Informationsaustausch zwischen Entwicklern



Zielsetzung

Versionierung folgender Eingangsprodukte mit GitLab:

- Quellcode
- Dokumente
- Konfigurationen (KV, KB)
- Produktdefinitionen (PD). Die PD ist die formale Beschreibung der Abhängigkeiten zwischen (Teil-)Produkten zwecks eindeutiger Versionierung sowie zur Erzeugung von Ausgangsprodukten (Distributionspaketen, Datenkatalogen und Systemen, …)
  - Die PD ist letztlich das *Buildscript*, mit dem mittels eines geeigneten *Buildtools* aus den unterschiedlichen Quelldaten unter Berücksichtigung der Abhängigkeiten zu anderen Produkten die gewünschten Zielprodukte (z. B. Distributionspakete der SWE oder Updateseiten für die Plug-in) erstellt werden.
  - Beim Einsatz des Buildtools Maven entspricht das PD dem POM (pom.xml Datei), beim Einsatz von Gradle als Buildtool entspricht das PD der build.gradle Datei.

Zielsetzung

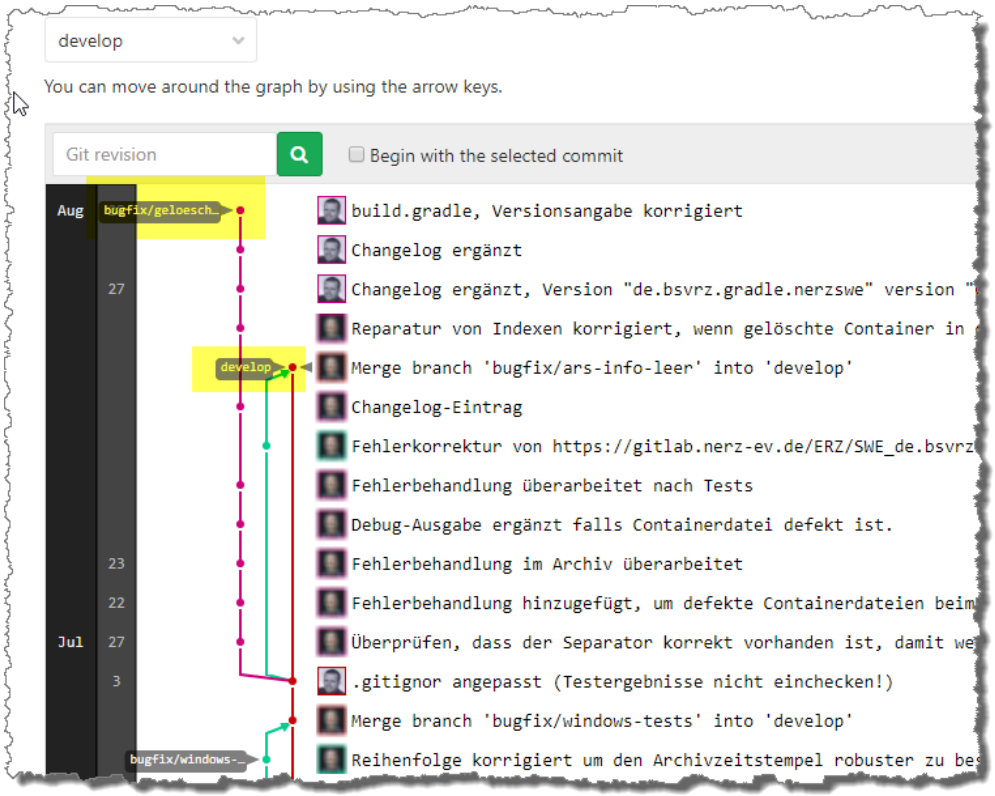
Erzeugung folgender Ausgangsprodukte durch den NERZ e.V.:

- Distributionspakete für SWE (Serverprozesse)
- Distributionspakete für Rahmenwerk (Basis-Bedienoberfläche, Client)
- Updateseiten für Plug-in (Erweiterungen der Bedienoberfläche)
- Dokumentationen
- aktueller Datenkatalog als HTML-Version auf Basis der aktuellen Konfigurationsverantwortlichen (KV) / Konfigurationsbereiche (KB).
- Konfigurationen (KV, KB)

Zielsetzung

Verbesserte Weiterentwicklung, Behebung von Bugs, schnellere Verfügbarkeit

- Release wird (wie bisher) veröffentlicht
- aktuelle Stände der Weiterentwicklung (develop) sowie Details weiterer Änderungen (z. B. Bugfixes) sind jederzeit verfügbar
  - durch einfachen Build-Prozess seitens eines Entwicklers

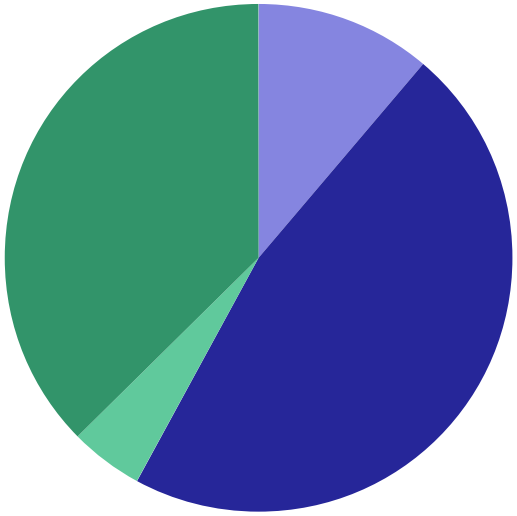


SWE, Plug-in und Dokumente / Pakete

Serversoftware	Anzahl
SWE	120
Veröffentlichte Dokumente / Einzelpakete	500

Bedienung und Visualisierung	Anzahl
Plug-in	50
Veröffentlichte Dokumente / Einzelpakete	400

Softwarepakete gesamt	**170**
Veröffentlichte Dokumente gesamt	**900**



Bisherige Vorgehensweise

Bisherige Vorgehensweise

- **Erstellung der Distributionspakete etc.**
  - durch den jeweiligen Hersteller!
  - Nur Bereitstellung durch FTB des NERZ

- **Verbesserte Weiterentwicklung, Behebung von Bugs, schnellere Verfügbarkeit**
  - nur in Ausnahmefällen bei direkter Bereitstellung durch bearbeitende Firma
  - Standard: nur Releases verfügbar, z. T. große Zeitabstände der Veröffentlichung

Bisherige Vorgehensweise

- **einfache Bereitstellungsmöglichkeit** des aktuellen Quellcodes der Softwaremodule, Konfigurationen, aktueller Distributionspakete und der Dokumentation für **Entwickler** und Anwender
  - Bereitstellung auf NERZ-Seite
    - Distributionspaket mit gezipptem Source (aktuellste Version, tw. auch ältere)
    - Dokumente (aktuellste Version)

- **Versionierung**
  - nur indirekt durch Vergabe von Versionsnummern bei den Produkten, keine Versionierung im Sinne eines Versionsverwaltungsprogramms wie GIT.
  - keine Nachvollziehbarkeit von Änderungen (außer durch manuell gepflegte Änderungssicht)

- **Erstellung der Distributionspakete etc.**
  - durch den jeweiligen Hersteller!
  - Nur Bereitstellung durch FTB des NERZ

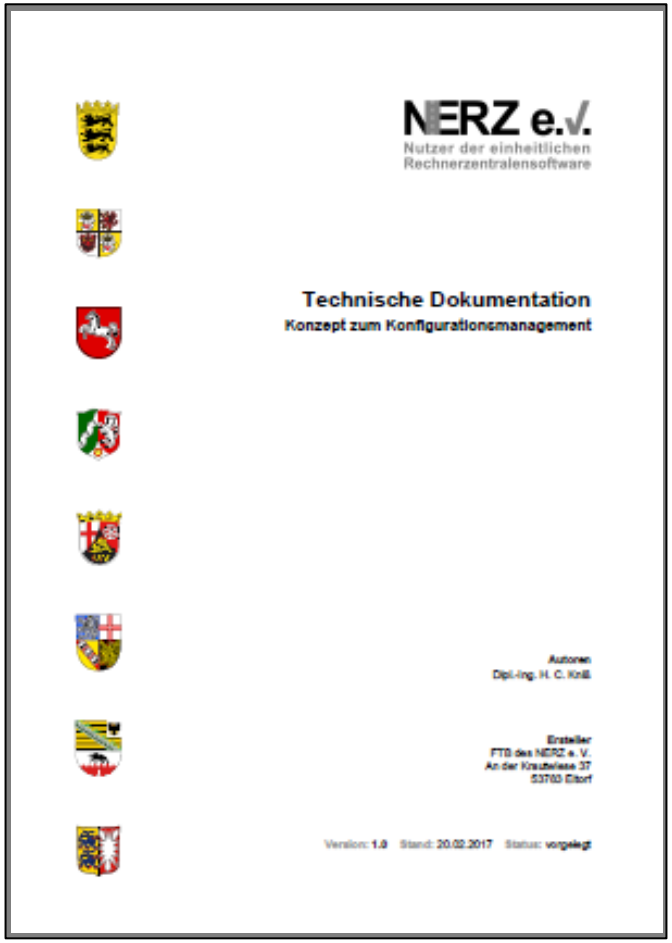
Aktuelle Vorgehensweise

Konfigurationsmanagement

- Umsetzung

  „Konzept zum Konfigurationsmanagement“

- getroffene Entscheidungen
  - Verwaltung über **GitLab**
  - Bereitstellung aller SWE, Plug-in, RW Dokumente, KV über **GitLab**
  - Builds für SWE auf Basis von **Gradle**
  - Builds für RW und Plug-in auf Basis von **Maven**


NERZ e.V.
Nutzer der einheitlichen Rechnerzentralensoftware

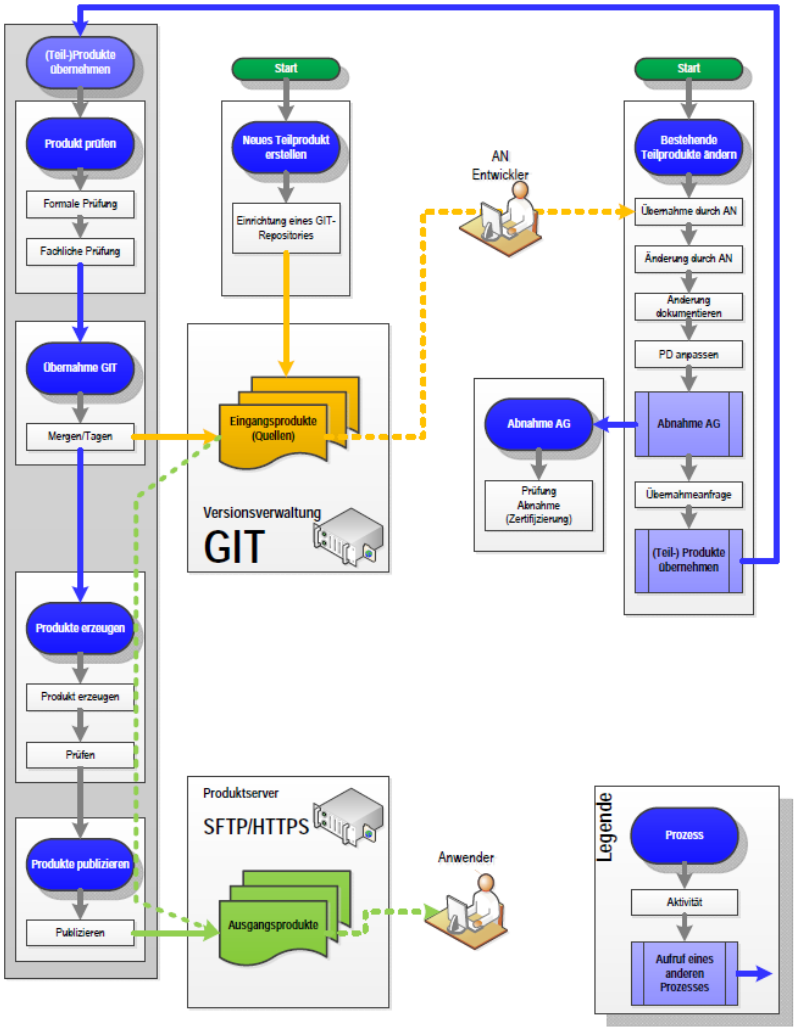
Technische Dokumentation
Konzept zum Konfigurationsmanagement

Autoren
Dipl.-Ing. H. C. Kniß

Ersteller
FTB des NERZ e. V.
An der Krautwiese 37
53783 Eitorf

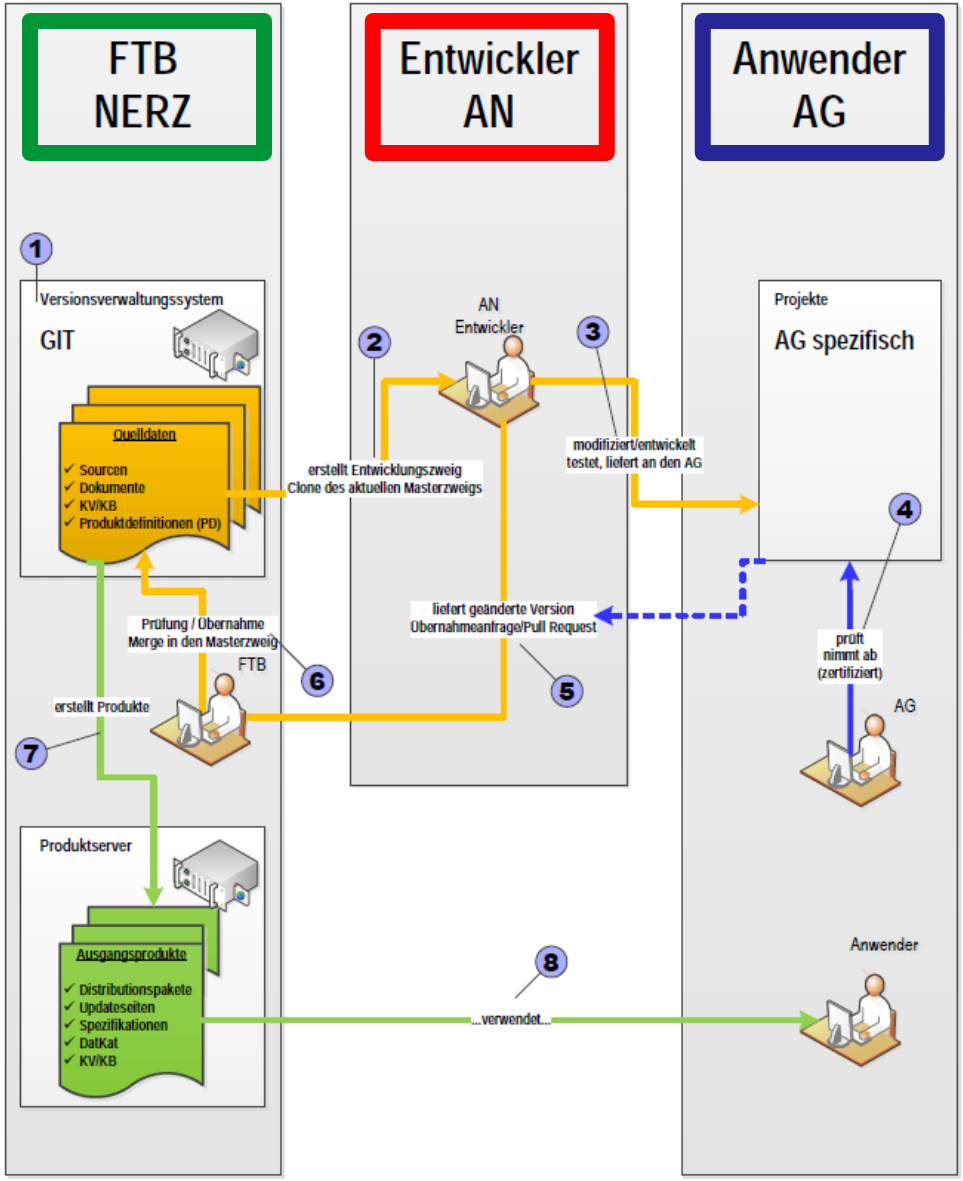
Version: 1.0 Stand: 20.02.2017 Status: vorgelegt

Konfigurationsmanagement – Prozesse und Aktivitäten



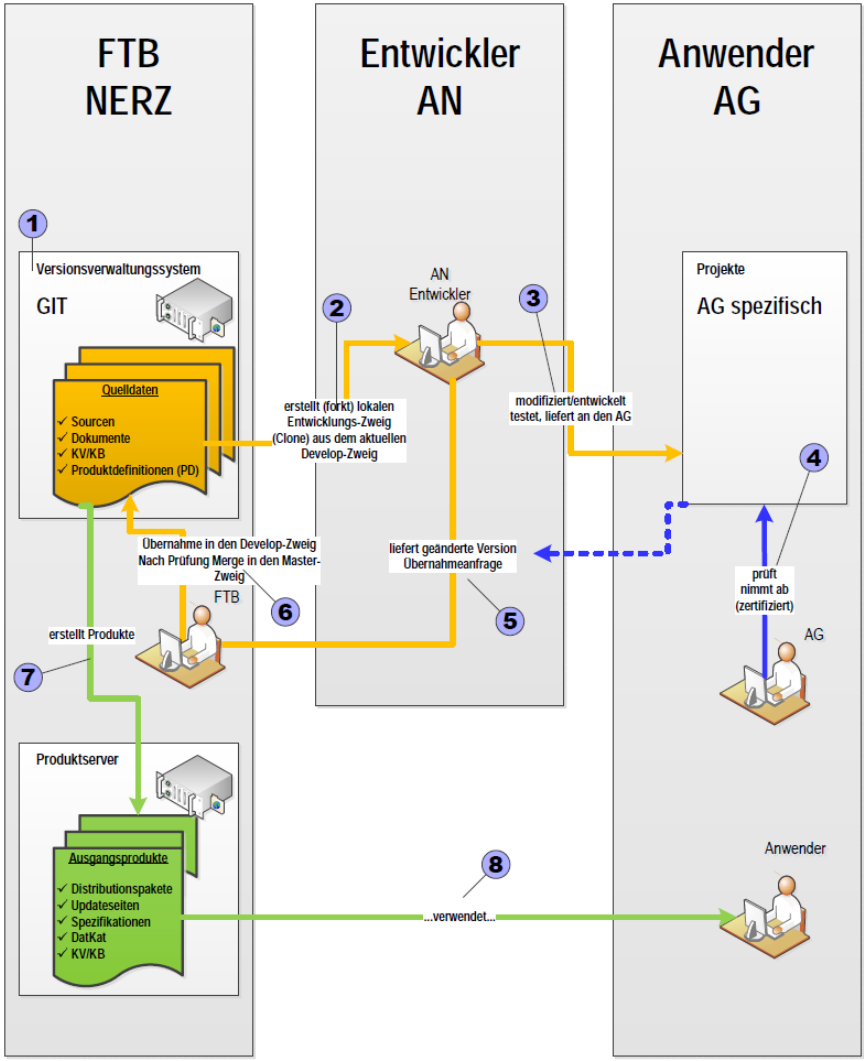
Abläufe

1. GIT / GitLab Repositories
2. Entwicklungszweig erstellen
3. Modifikation / Fehlerbehebung
4. Freigabe / Prüfung
5. Übernahmeanfrage an FTB
6. Übernahme durch FTB
7. Produkt / Release erstellen
8. Produkt nutzen



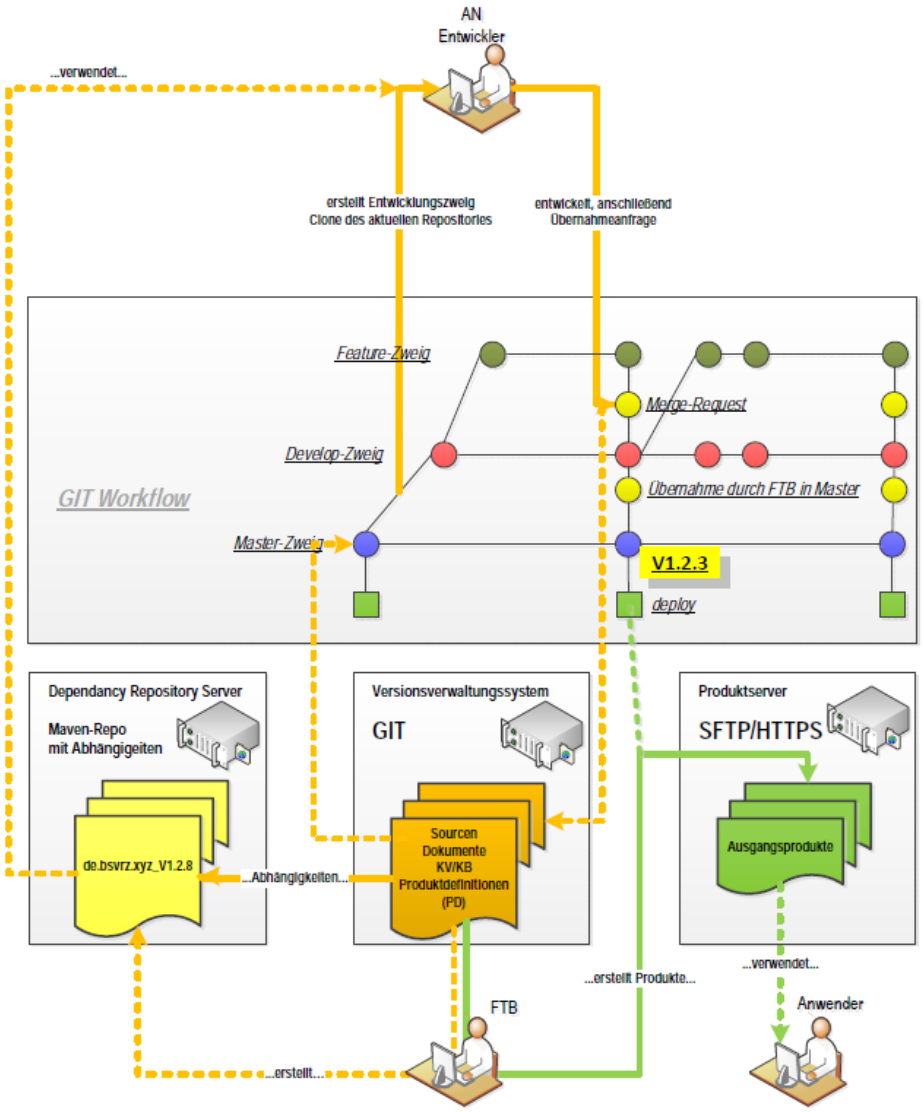
Abläufe - Überblick

1. Auf dem Versionsverwaltungsserver werden die Quelldaten in GIT-Repositories gespeichert.
2. Ein Entwickler (AN) kann die gewünschten Quelldaten (Source, Dokumente, KV, …) auschecken, indem er einen Clone (lokale Kopie) des seitens der FTB bereitgestellten Entwicklungszweigs (Develop-Zweig bzw. des Feature/Bug-Zweig) erstellt.
3. Der AN modifiziert oder erstellt das seitens eines AG gewünschte Produkt, testet dieses und liefert es an den AG (z. B. für ein AG spezifisches Projekt)
4. Der AG prüft das gelieferte Produkt, nimmt dieses ab und zertifiziert dieses ggf.
5. Das geänderte / neue Produkte wird an die FTB zur Übernahme in das GIT-Repository übergeben (Übernahmeanfrage)
6. Die FTB übernimmt die Änderungen in den Develop-Zweig. Die Änderungen werden dort formal geprüft und nach erfolgreicher Prüfung in den Master-Zweig des zentralen GIT-Repository „gemerged“ und mit einer neuen Version gekennzeichnet.
7. Die FTB erstellt aus den geänderten Quelldaten auf Basis der Produktdefinition (build.gradle) mit dem Buildtool (Gradle) die eigentlichen Ausgangsprodukte (Distributionspakete, Update-seiten, …) und stellt diese auf dem allgemein zugänglichen Produktserver zur Verfügung. Zudem wird das versionierte Produkt auf einem Server zur Verwaltung der Abhängigkeiten in einer spezifischen Form (Maven-Repository) bereitgestellt.
8. Anwendern stehen auf dem Produktserver die unterschiedlichen Produkte zum Download bereit.



Abläufe - Praxis

- Randbedingungen
  - Es gibt einen *Master-Zweig*, dessen Inhalt jederzeit *deploybar* sein muss (aus dem sich jederzeit stabile Produkt-Versionen erstellen lassen müssen).
  - Parallel dazu gibt es einen *Develop-Zweig*, in den alle Änderungen zeitnah durch den Entwickler übernommen werden.
  - Der *Develop-Zweig* enthält alle aktuellen Änderungen und Bugfixes.
  - Es kann jederzeit eine SNAPSHOT Version aus dem *Develop-Zweig* erzeugt werden.



Beispiel
Distributionspaket erstellen

Build-Skripte (GRADLE)

build.gradle 2.28 KB

```
1  //-------------------------------------------------------------------
2  // NERZ-SWE-Plugin
3  //-------------------------------------------------------------------
4  plugins {
5    id "de.bsvrz.gradle.nerzswe" version "0.12.0"
6  }
7
8  //-------------------------------------------------------------------
9  // SWE-Eigenschaften
10 //-------------------------------------------------------------------
11 group = 'de.bsvrz.ars'
12 version = '4.2.1-SNAPSHOT'
13 description = 'Archivsystem'
14
15 // Properties des NERZ-SWE-Plugins:
16 nerzswe {
17   sweStatus = 'BETA'
18   sweDatum = ''
19   mainClassName = 'de.bsvrz.ars.ars.mgmt.ArchiveManager'
20 }
21
22 //-------------------------------------------------------------------
23 // Abhängigkeiten
24 //-------------------------------------------------------------------
25 String kernsoftware_version = '3.12.0'
26 dependencies {
27   compile group: 'de.bsvrz.dav', name: 'de.bsvrz.dav.daf', version: kernsoftware_version
28   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.losb', version: kernsoftware_version
29   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.operatingMessage', version: kernsoftware_version
30   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.debug', version: kernsoftware_version
31   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.dataIdentificationSettings', version: kernsoftware_version
32   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.dataSerializer', version: kernsoftware_version
33   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.commandLineArgs', version: kernsoftware_version
34   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.communicationStreams', version: kernsoftware_version
35   compile group: 'de.bsvrz.sys', name: 'de.bsvrz.sys.funclib.kappich', version: kernsoftware_version
36   compile group: 'com.google.guava', name: 'guava', version: '21.0'
37
38   // Abhängigkeiten Integrationstests
39   testCompile group: 'de.kappich.pat', name: 'de.kappich.pat.testumg', version: kernsoftware_version
40
41   testCompile group: 'junit', name: 'junit', version: '4.12'
42   testCompile group: 'org.hamcrest', name: 'hamcrest-library', version: '1.3'
43 }
```

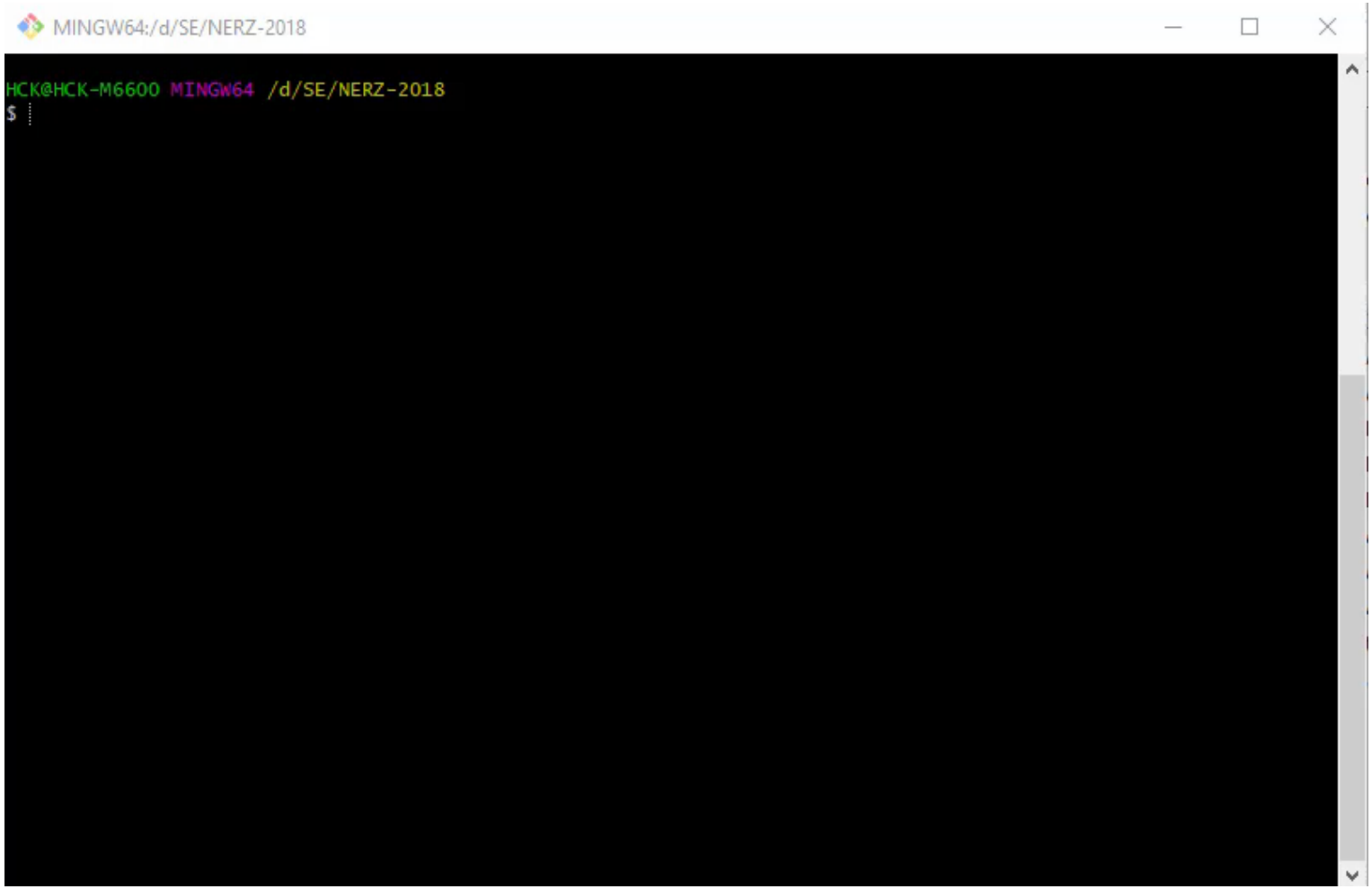
Beispiel

Distributionspaket erstellen

```
1. git clone http://gitlab.nerz-ev.de/ERZ/SWE_de.bsvrz.ars.ars.git de.bsvrz.ars.ars
2. cd de.bsvrz.ars.ars
3. git checkout develop                    // oder (master…)
4. ./gradlew clean build
```

- **fertig!**
- **Ergebnisse im Order „build“**

Distributionspaket erstellen (Archivsystem)

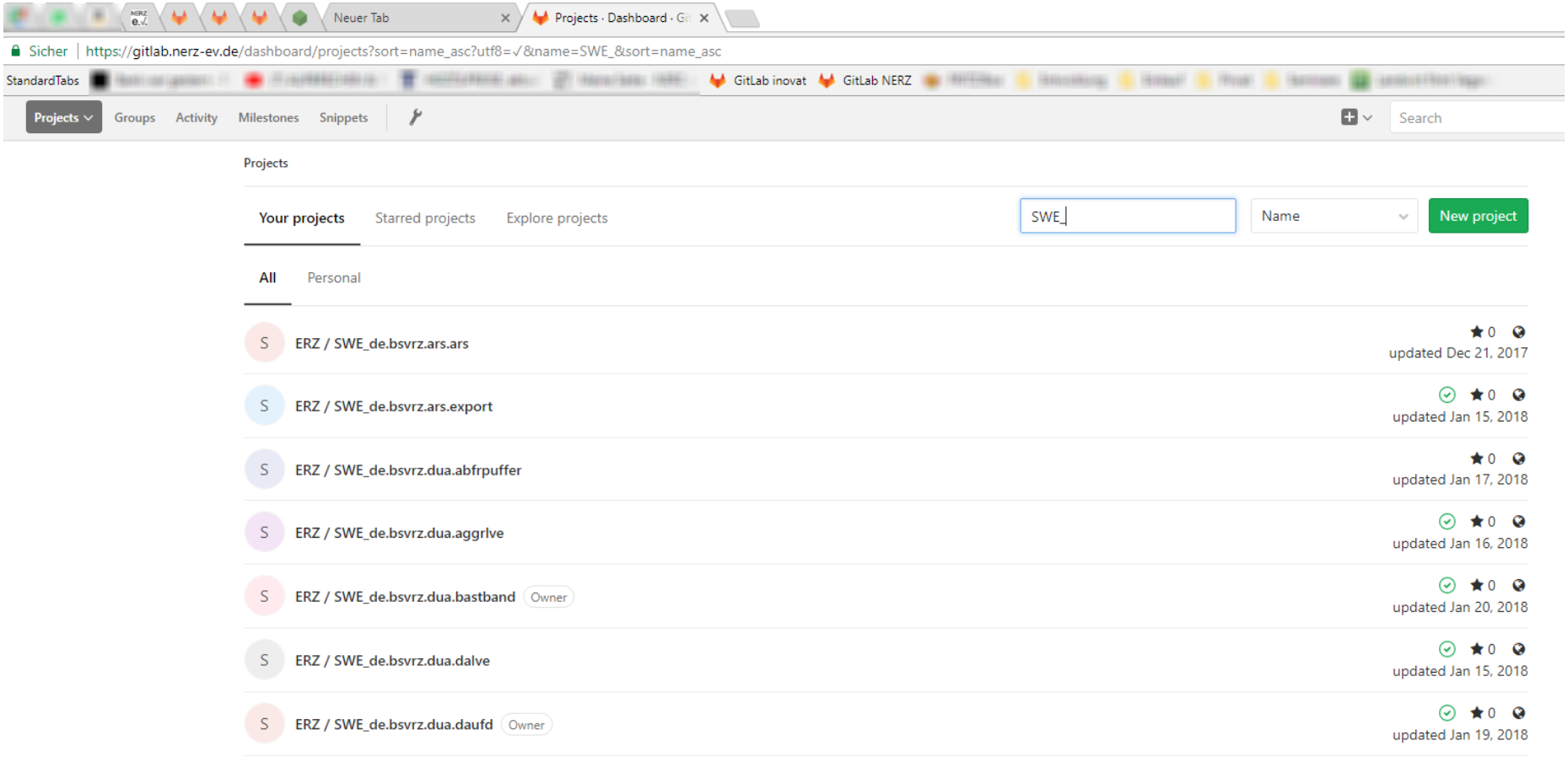


Sonstiges…

Sonstiges

- Checkstyle
  - Formale Prüfung der Sourcen (Formatierung, Kommentierung, …)
- FindBugs
  - Erkennen potentieller Fehler im Source
- CI (Continuous integration)
  - gitlab-ci.yml Dateien
  - automatischer Build beim Commit / Push

GitLab Server (https://gitlab.nerz-ev.de)



GitLab Server

GitLab Server - Rechte

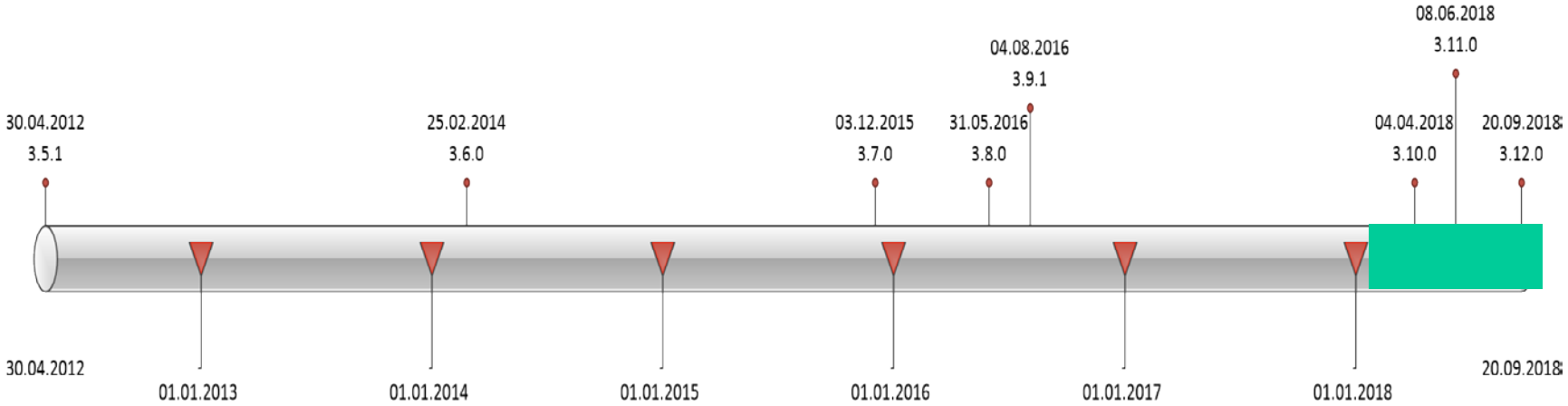
- **https://gitlab.nerz-ev.de**
- **Zugriffsrechte**
  - Keine Rechte oder Login notwendig für
    - Repositories im Browser ansehen
    - Download der Daten
    - „git clone“ der Daten
    - Anlegen von Issues (Eintrag von Fehlern / Änderungswünschen / ToDo‘s etc.)
  - Benutzerkennung (Einrichtung durch FTB) als Developer
    - Eigene Entwicklungszweige anlegen (branch)
    - push auf Entwicklungszweige
    - Anfordern eines Merge-Request (mit develop-Zweig)

Praxiserfahrungen…

Vergleich Vorher-Nachher

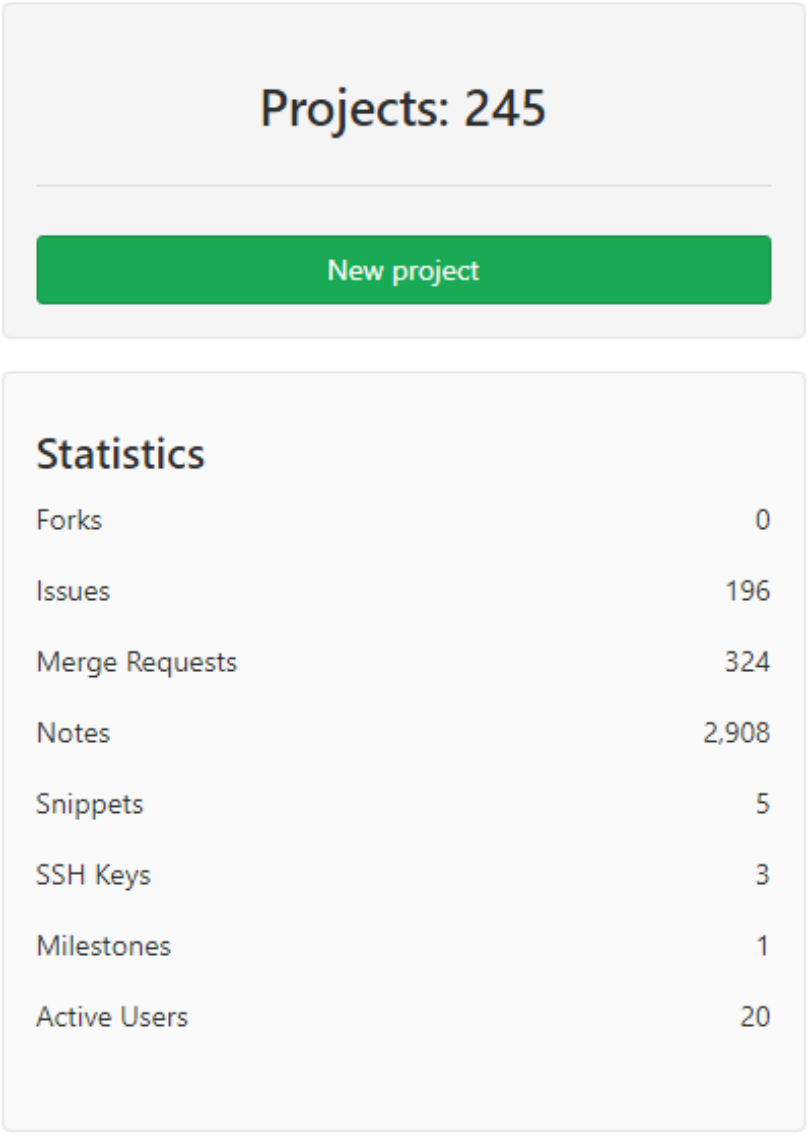
Release Zyklen

- Beispiel Kernsoftware
- nur Releases mit wesentlichen Funktionserweiterungen

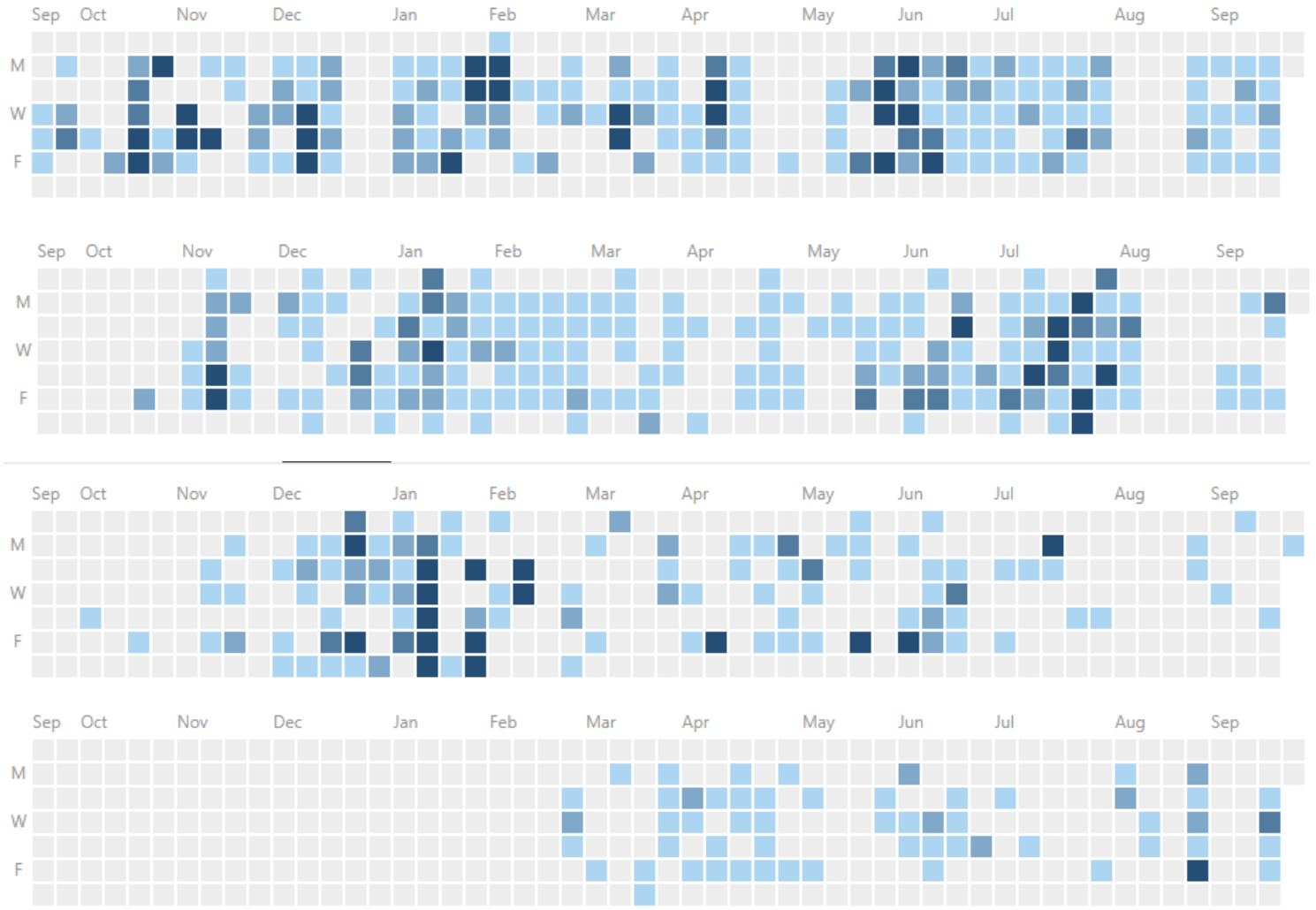


Aktivitäten

- Zeitraum (ca. letzten 10 Monate)
- Einträge (Issues): 196
- Merge Requests: 324
  - Bereitstellung aktueller Änderungen
- Anmerkungen: 2.908
  - i.d.R. Anmerkungen zu einzelnen Änderungen (Commits)



Nutzung KM durch Entwickler (Sept. 2017- Sept. 2018)



Konfigurationsmanagement

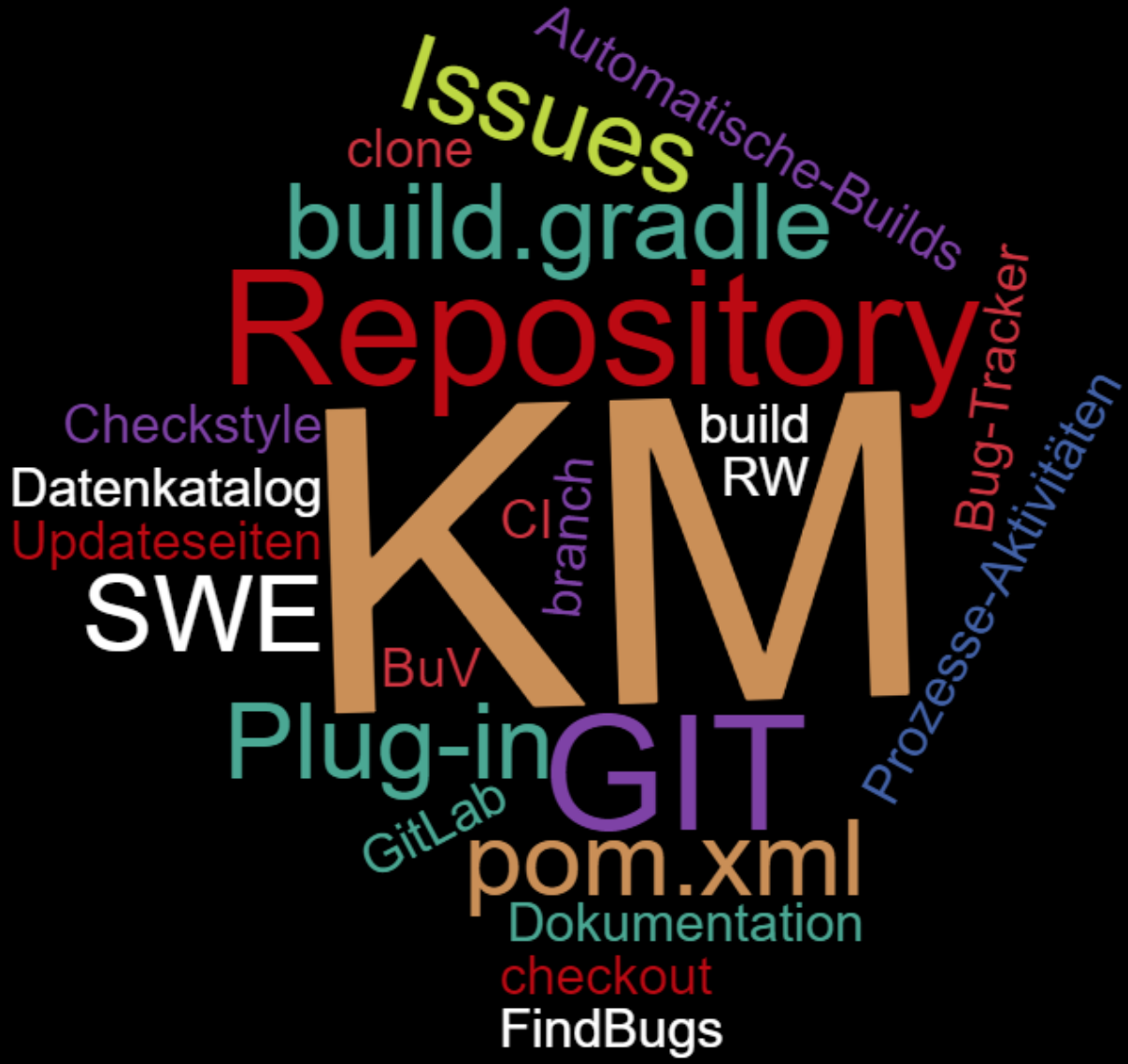
Aktueller Stand

- Bereitstellung und Verwaltung aller Produkte über ein Versionsverwaltungsprogramm
  - GIT
  - Quellcode
  - Dokumente
  - Distributionspakete
  - Datenkatalog
  - Konfigurationen (Dokumentation der Abhängigkeiten zwischen den vorgenannten Produkten)

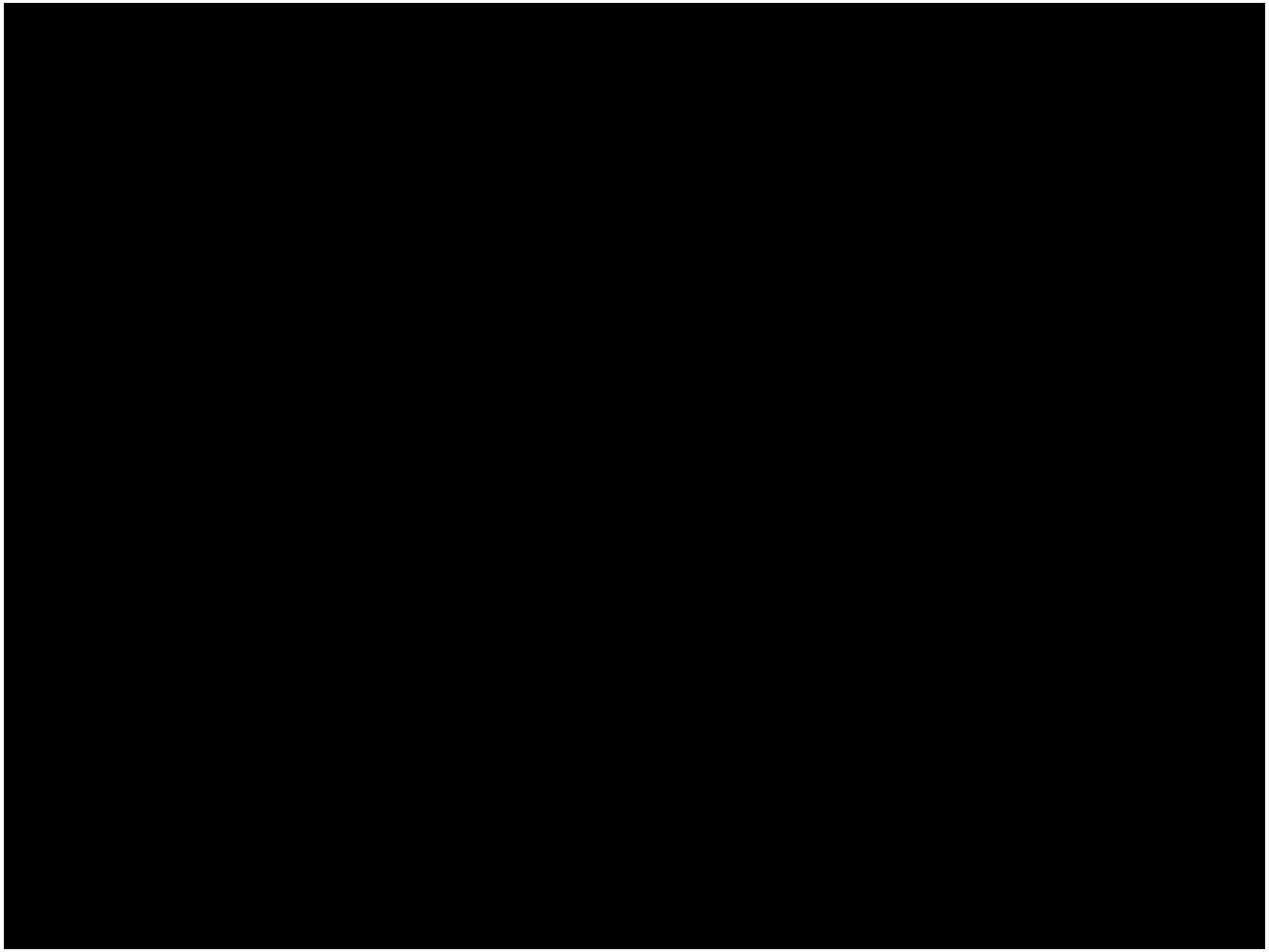
- Distributionspaketerstellung durch den NERZ e.V. (FTB)
  - konkrete Vorgaben zur Lieferung von Produkten durch AG/AN an den NERZ e.V.
  - Automatische Builds mit Maven (Client → RW, Plug-in)
  - Automatische Builds mit Gradle (Server → SWE)

Zusammenfassung

- Vollständige Versionierung für Sourcecode, Dokumente, KV etc.
- Werkzeug unterstützte Versionsverwaltung, Bugtracking, kontinuierliche Integration (CI), automatische Testausführung, Build
- Schnellere Behebung von Fehlern und insbesondere sofortige Verfügbarkeit des aktuellen Softwarestandes
- Unterstützung der Entwickler
- Dokumentation und Nachvollziehbarkeit aller Änderungen durch alle beteiligten Entwickler / AG
- …


Automatische-Builds
Issues
clone
build.gradle
Repository
Bug-Tracker
Checkstyle
build
RW
Datenkatalog
CI
branch
Updateseiten
Prozesse-Aktivitäten
SWE
KM
BuV
Plug-in
GIT
GitLab
pom.xml
Dokumentation
checkout
FindBugs

…wenn es manchmal zu technisch war…



Kontaktdaten

Fachtechnische Beratung NERZ e.V.

c/o

inovat

Dipl.-Ing. H. C. Kniß

An der Krautwiese 37

53783 Eitorf

02243 – 94 84 192

01577 – 207 44 39

christian.kniss@inovat.de

www.inovat.de