TECO Research Group

Marcel Köpke
Matthias Budde
Till Riedel

Qualitätssicherungsbericht

Version 1.0

Visualizing & Mining of Geospatial Sensorstreams with Apache Kafka

---

Jean Baumgarten
Thomas Frank
Oliver Liu
Patrick Ries
Erik Wessel

2. September 2018

Inhaltsverzeichnis

1 Einleitung

Dieser Qualitätssicherungsbericht dient als Übersicht der Fehlerbehebungen und Verbesserungen an dem Code des Projekts während der Qualitätssicherungsphase.

Es wurden verschiedene Werkzeuge verwendet um uns in der Qualitätssicherung zu unterstützen:

- **JUnit**: für Java Unit-Tests
- **Jasmine**: für JavaScript Unit-Tests
- **EclEmma** (in Eclipse): zur Visualisierung der Code-Überdeckung durch die Unit-Tests
- **KafkaStreamTestUtils**: zum Testen von Klassen, die mit Kafka kommunizieren müssen
- **Sonarqube** (auf sonarcloud.io): zum Untersuchen des Quellcodes nach Bugs, Code Smells und Vulnerabilities.

Nachfolgend wird eine Übersicht über Bugfixes und Tests gegeben. Einige Tests wurden bereits in der Implementierungsphase verfasst.

2 Bugfixes

2.1 Jean

Servlet ging nicht

- Fehlersymptom: Servlet wird von Tomcat nicht gestartet.
- Fehlerursache: Kafka und JSON dependency vertragen sich anscheinend nicht mit Tomcat.
- Fehlerbehebung: Export-Funktionalität die Kafka benötigt wurde als separate jar ausgelagert.

usr.home Ordner

- Fehlersymptom: Servlet und Export jar benutzen nicht die selben Dateien zur persistenten Datenspeicherung.
- Fehlerursache: User Home Ordner ist aus Sicht eines Servlets innerhalb des Tomcat Ordners, während eine jar einen anderen Ordner betrachtet.
- Fehlerbehebung: Hardcoded Verzeichnisse innerhalb des Dockers.

Export geht nicht

- Fehlersymptom: Export bricht nach Start sofort ab und gibt leere Datei zurück.
- Fehlerursache: Export wurde abgebrochen, wenn von Kafka angefragte Daten außerhalb des für den Export erwünschten Zeitraums liegen. Insbesondere auch wenn sie davor liegen.
- Fehlerbehebung: Es wird nun nur beendet, wenn der Zeitraum der erhaltenen Daten nach den erwünschten Daten liegt.

Kafka Pollzeit

- Fehlersymptom: Es wird nur ein Bruchteil der Daten exportiert.
- Fehlerursache: Poll Timeout Zeit an Kafka ist zu kurz.
- Fehlerbehebung: Poll Timeout Zeit wird erhöht.

Millisekunden im TimeFrame

- Fehlersymptom: Exporter stürzt bei Ausführung mit manchen Daten ab.
- Fehlerursache: Exporter kann nur Zeiten nach ISO 8601 ohne Millisekunden verarbeiten.
- Fehlerbehebung: Exporter unterstützt nun auch Millisekunden.

@iot.id

- Fehlersymptom: Es sind keine `@iot.id`s für Observations im Export enthalten. Stattdessen steht überall `null`, wodurch der Import dieser Daten unmöglich ist.
- Fehlerursache: Es wurde ein leeres Feld aus den Kafkadaten ausgelesen.
- Fehlerbehebung: Wechsel auf das richtige Feld.

2.2 Thomas

Bug 1

- Fehlersymptom: Blockieren von Http-Anfragen bei Webinterface-internem JavaScript Code
- Fehlerursache: Cross-Origin Resource Sharing (CORS) zwischen Webinterface und Server nicht möglich
- Fehlerbehebung: Konfigurieren der Server-lokalen Tomcat Konfigurationsdatei

Bug 2

- Fehlersymptom: Fehlerhafte Synchronisierung der Zustände bei Export-, Sensortyp-, Favoriten-, und Einstellungs-Modal-Fenstern
- Fehlerursache: Veränderung der Auswahl/Parameter in einem Modal-Fenster werden nicht an anderes Modal-Fenster weitergegeben
- Fehlerbehebung: Korrekte Beobachter Struktur zwischen den Komponenten aufbauen

Bug 3

- Fehlersymptom: Unresponsivität der Anpassung bei Veränderung der vertikalen Fenstergröße
- Fehlerursache: Da der Container der Karte initial eine fest definierte Höhe braucht, kann man das nicht in CSS in Prozent, also dynamisch zur Fenstergröße festlegen
- Fehlerbehebung: Dynamisches Styling der Höhe über JavaScript Beobachter

2.3 Oliver

2.3.1 Bridge

Main-Klasse

- Fehlersymptom: Main-Klasse mit main-Methode konnte nach kompilieren ins jar-Format bei der Ausführung nicht gefunden werden.
- Fehlerursache: Einige Einträge in der pom.xml fehlten.
- Fehlerbehebung: Hinzufügen einiger Einträge in der pom.xml.

Source-Ordner

- Fehlersymptom: src/main/java wurde anfangs nicht als Java Source-Ordner erkannt.
- Fehlerursache: Eclipse wurde falsch eingestellt.
- Fehlerbehebung: Eclipse wurde richtig eingestellt, Importe und Paketdefinitionen wurden entsprechend angepasst.

`null`-Checks

- Fehlersymptom: In einigen Methoden wurde nicht nach `null`-Objekten geprüft.
- Fehlerursache: Fehlende Checks.
- Fehlerbehebung: Checks wurden implementiert.

Logger

- Fehlersymptom: Konfigurationsprobleme beim Logger, nicht-einheitlicher Logger mit dem Rest von PaVoS.
- Fehlerursache: `java.util.logging.Logger` wurde verwendet.
- Fehlerbehebung: der Logger für Klassen der Bridge wurde von `java.util.logging.Logger` zu `org.slf4j.Logger` geändert.

`System.out` und `System.err`

- Fehlersymptom: Fehler- und Informationsnachrichten wurden über `System.out` oder `System.err` ausgegeben.
- Fehlerursache: Fehler- und Informationsnachrichten wurden über `System.out` oder `System.err` ausgegeben.
- Fehlerbehebung: Fehler- und Informationsnachrichten werden über den Logger ausgegeben.

MqttConsumer

- Fehlersymptom: Verbindung zu MQTT wird bei Verbindungsverlust nicht wiederaufgebaut.
- Fehlerursache: Die `connectionLost()` Methode wurde falsch implementiert.
- Fehlerbehebung: Der Verbindungsaufbau wurde nun in eine zusätzliche private Methode geschoben, die sowohl im Constructor als auch bei Verbindungsverlust ausgeführt wird.

Properties

- Fehlersymptom: Die Bridge startet auch bei fehlerhaften Properties, oder `PropertiesFileReader.init()` beendet die Bridge.
- Fehlerursache: `PropertiesFileReader.init()` führt `System.exit(-1)` bei fehlerhaften Daten aus.
- Fehlerbehebung: Die Main-Methode prüft ob die Initialisierung erfolgreich war und beendet dann dort die Bridge.

Sonarqube

- Nach der Analyse der Bridge mit Sonarqube wurden 35 von 54 Code Smells und 2 von 2 Vulnerabilities gefixed.

2.3.2 Database

- Mehrere Code Smells wurden mit Sonarqube entdeckt und durch Erik gefixed.

2.4 Patrick

Bug 1

- Fehlersymptom: Topics wurden in Kafka nicht erstellt, welches dann teilweise zu Fehlern in den einzelnen Nutzern der Kafka Stream Funktionalität führte.
- Fehlerursache: Die Input Topics waren nicht in Kafka enthalten
- Fehlerbehebung: Ich habe eine Klasse hinzugefügt welche alle Input Topic generiert und somit garantiert dass das Kafka System alle Topics die benötigt werden enthalten sind.

Bug 2

- Fehlersymptom: Die Daten wurden nicht von Frost auf Kafka übertragen.
- Fehlerursache: Es gab Null Values bei den Zeiten welche nicht erlaubt waren
- Fehlerbehebung: Im Importer wurde einfach gefixt, dass die Zeit nicht null sein darf.

Bug 3

- Fehlersymptom: Die Kafka Topics von den Prozessen wurden nicht erstellt.
- Fehlerursache: Ich ging davon aus, dass FeatureOfInterest und Observation eine bijektive Verbindung zu einander haben.
- Fehlerbehebung: Ich habe das so gefixed, dass es eine surjektive Verbindung wird durch das Verwenden von einem leftJoin.

Bug 4

- Fehlersymptom: Der Export Prozess produzierte fehlerhafte Messages im Export Topic
- Fehlerursache: Gleicher Fehler wie bei Bug 3.
- Fehlerbehebung: Gleicher Fehler wie bei Bug 3.

2.5 Erik

Bug 1

- Fehlersymptom: Maven auf Debian kann Pavos-Core nicht erstellen `ClassNotFoundException: LoggerFactory.java`
- Fehlerursache: Benötigte Dateien sind nicht im Classpath
- Fehlerbehebung: Maven dependency plugin fügt dependencies zum Classpath hinzu

Bug 2

- Fehlersymptom: Maven auf Debian kann Pavos-Core nicht erstellen Avro has no Manifest
- Fehlerursache: Avro.jar enthält keine Manifest-Datei
- Fehlerbehebung: Aus Maven dependency plugin ausschließen

Bug 3

- Fehlersymptom: Erstellter GeoGrid akzeptiert Sensordaten.
- Fehlerursache: `PointNotOnMapException`
- Fehlerbehebung: In der Erstellung wurde x und y als Breite und Höhe interpretiert, weshalb die Kartengröße nur 25% der gewollten Größe betrug

Bug 4

- Fehlersymptom: MultiGradient toString() wirft eine `NullPointerException`
- Fehlerursache: Farbwerte hatten einen Index-Shift um 1, weshalb der erste Farbwert übersprungen wurde und der letzte null war
- Fehlerbehebung: Index-Shift korrigiert und Absicherung gegen NullPointerException in ColorUtil erstellt

Bug 5

- Fehlersymptom: Kafka-Tests schlagen bei mvn clean install fehl
- Fehlerursache: JUnit tests reichen nicht aus, da Kafka zur Kommunikation benötigt wird.
- Fehlerbehebung: Dependency für Kafka-Streams-Test-Util für Version 1.1.0 hinzugefügt, da 1.0.1 nicht existiert

Bug 6

- Fehlersymptom: WebServer soll immer nach den KafkaProzessen gestartet werden und trotzdem soll Webserver auch selbstständig gestartet werden können. Beide haben main(String[] args)
- Fehlerursache: Zwei main Methoden können nicht ausgeführt werden, da keine zwei Main klassen existieren können.
- Fehlerbehebung: WebServer implementiert Runnable und ruft run() in seiner main Methode auf. KafkaProzesse, die Main Klasse startet WebServer auf einem Thread.

Bug 7

- Fehlersymptom: WebServer wirft Fehlermeldungen und lässt die Verbindung abstürzen.
- Fehlerursache: http-Anfrage ist falsch formuliert oder angefragte Komponenten existieren nicht.
- Fehlerbehebung: Der http-Status des WebServer wird entsprechend geändert. Meistens auf Bad-Request und wird gesendet.

Bug 8

- Fehlersymptom: Die Datenbank kann bei der Übertragung nicht erreicht werden.
- Fehlerursache: Die Verbindung zur VM wurde nicht eingestellt.
- Fehlerbehebung: Host wurde von localhost zur VM-Host-Adresse geändert.

Bug 9

- Fehlersymptom: Daten zur Überprüfung konnten im Test nicht überprüft werden oder wurden falsch ausgewertet.
- Fehlerursache: Die Überprüfung verwendete einen Regex-Ausdruck, der Zeichen enthielt, die falsch interpretiert wurden.
- Fehlerbehebung: Zeichen wie { und [wurden mit einem doppelten \ versehen um sie erkenntlich zu machen.

Bug 10

- Fehlersymptom: Neuere Java Funktionalitäten können nicht verwendet werden
- Fehlerursache: Es wurde die falsche Java-Version im Classpath verwendet
- Fehlerbehebung: Update auf JavaSE-1.8 mit JDK

Bug 11

- Fehlersymptom: Verbindung des Graphite-Sender wird zu häufig geschlossen
- Fehlerursache: Bei jeder Nachricht wird das Socket überschrieben
- Fehlerbehebung: SocketManager Klasse wurde erstellt

Bug 12

- Fehlersymptom: Mehrere Einheiten einer Klasse werden erstellt obwohl sie die gleichen Daten beinhalten müssen
- Fehlerursache: Offener Konstruktor und eigene Daten pro Instanz
- Fehlerbehebung: Singleton

Bug 13

- Fehlersymptom: Schlecht möglich die Reihenfolge von Operationen eines Grids beim Updaten einheitlich zu ändern
- Fehlerursache: update() wurde intern im Grid ausgeführt
- Fehlerbehebung: GeoGridManager erstellt, der Update-Prozess zentralisiert

3 Testübersicht

3.1 Jean

Manuelle Tests:

- Import von Daten und anschließendes prüfen, ob Daten in Frost angekommen sind.
- Export von Daten und anschließendes prüfen, ob Daten in Datei angekommen sind.

Mit JUnit getestete Klassen:

- Import
  - CSVReaderStrategy
  - DataTable
  - FileImporter
  - FrostSender
  - ReaderType
- Export
  - CSVWriterStrategy
  - ExportProperties
  - FileType
  - TimeIntervall

3.2 Thomas

Manuelle Tests:

- Bedienen der Karte (Bewegen, Zoomen und Vollbild aktivieren)
- Bedienen des Graphen (Datenquelle und Darstellung ändern)
- Öffnen, Bedienen und Schließen der Modal-Fenster
- Speichern und Aufrufen eines Favoriten
- Melden eines Sensors

3.3 Oliver

Mit JUnit getestete Klassen:

- `server.database`
  - `ObservationDataToStorageProcessor`
- `bridge`
  - `FrostIotIdConverter`
  - `JmkbKafkaProducer`
  - `JmkbMqttConsumer`
  - `Main`
  - `MqttMessageConverter`
  - `PropertiesFileReader`
  - z.T. auch alle Avro-generierten Klassen

3.4 Patrick

- Test 1: Ich habe das ganze System Lokal ausführt und mit den Daten gefüttert welche wir von unseren Betreuern bekommen haben und dann alle Prozesse drauf gestartet. Zum weiteren Testen davon haben wir unseren Importeur genutzt, welcher die Daten von SensorUp einfach in unserem Frost pumpt. Dann wurden manuell die Topics kontrolliert und ob sie sinnvoll befüllt sind. Dadurch wird die Funktionalität der Bridge festgestellt.
- Test 2: Ich habe dann wieder meine Prozesse auf diesen Daten ausgeführt und überprüft ob die Topics erstellt wurden und sinnvoll waren. Dieser Test wurde sehr oft ausgeführt.
- Test 3: Wir haben die Bridge laufen gelassen und dabei Kafka neugestartet.
- Test 4: Jetzt wurde ein größerer Integrationstest gemacht. Es wurden Sachen in Frost importiert und diese dann von den Prozessen verarbeitet und diese Daten dann aus dem System exportiert.
- Test 4: Hier wurden alle Elemente vom Core und Importeur und Exporteur parallel ausgeführt um zu kontrollieren ob die Daten verarbeitet werden und auch richtig verarbeitet werden und zu anderen Modulen weiter geben werden. Dabei wurden die Schnittstellen getestet.

3.5 Erik

Mit JUnit getestete Klassen:

- Tuple2D & Tuple3D
- GeoJsonConverter & GeoJsonBuilder
- GeoPolygon & GeoRectangle
- TimeUtil
- GraphiteConfig
- GeoGrid & GeoRecRectangleGrid
- GeoGridManager
- GraphiteConverter & PythonMetricUtil
- SocketManager
- EnvironmentUtil
- Sender & GraphiteSender
- ObservationType
- ObservationData
- ObservationDataDeserializer
- Destination & DirectUploadManager
- Connector & DirectGraphiteConnector
- (Facade & ObservationDataToStorageProcessor)