MonVerifyTools - Dokumentation

Andreas Nicolai, Stefan Vogelsang, Andreas Söhnchen
Institut für Bauklimatik, TU Dresden
stefan.vogelsang@tu-dresden.de

25. Mai 2020

Zusammenfassung

Die *MonVerifyTools* sind Python-Scripte für eine automatische Prüfung von Messdaten hinsichtlich Korrektheit und Vollständigkeit. Sie dienen in erster Linie dazu, selbst bei einer Vielzahl von Sensoren und Messdaten Fehler, z.B. Sensorausfall oder fehlerhafte Messwerte, schnell zu erkennen. Zudem beinhalten die Scripte halb-automatische Korrekturmechanismen, um fehlerhafte/unvollständige Messdaten sinnvoll zu korrigieren und ergänzen, sodass eine nachfolgende Auswertung möglich ist.

Inhaltsverzeichnis

1 Einführung

1.1 Konzept

Wenn Messdaten automatisch erfasst werden, besteht immer auch die Möglichkeit des Ausfalls einzelner Sensoren, bzw. Erfassung von fehlerhaften Daten. Bei automatischer Langzeiterfassung können nicht alle Sensorwerte manuell überprüft werden, bzw. ist eine manuelle Prüfung sehr zeitaufwändig.

Daher ist nachfolgend eine Prüfmethodik und ein Programm (Python-Skript) beschrieben, welches diese Prüfung automatisiert ermöglicht und den Zeitaufwand dafür drastisch reduziert.

Weiterhin ist es notwendig, fehlerhafte oder fehlende Werte für die nachfolgende Auswertung der Daten sinnvoll zu ergänzen bzw. zu korrigieren. Auch hier ist eine manuelle Korrektur mitunter sehr zeitaufwändig. Auch sind bestimmte Korrekturmechanismen, wie z.B. das lineare Verschneiden von Daten, manuell kaum möglich. Auch für diese Arbeitsschritte bietet das nachfolgend beschriebene Programm die notwendige Funktionalität.

Voraussetzung für die automatische Prüfung ist ein einheitliches Messdaten-Dateiformat. Dieses *Monitoring File Format*[1] wird vom MonVerifyTools-Skript verwendet und muss daher von allen Messerfassungssystemen bereitgestellt werden.

Eine Datei hat im Prinzip folgendes Format:

```
SensorID,G1,SensorName1,SensorName2,G2,SensorName3,SensorName4,SensorName5
Unit,,T,%,,%,T,Pa

2016-03-04 00:00:00,G1,11.41,44.3,G2,55.3,-5.2,101343
2016-03-04 00:00:05,G1,11.41,44.3,G2,55.3,-5.2,101343
2016-03-04 00:00:10,G1,11.41,44.3,G2,55.3,-5.2,101343
.
.
.
```

Mit den folgenden Eckdaten der Formatspezifikation:

- Header und Datensektion sind durch *exakt* eine Leerzeile getrennt.
- Alle Spalten in der Datei sind mit Komma getrennt, Zahlen sind im englischen Format (221.21 bzw. 1.32e-4) abgelegt.
- Zeitstempel haben das Format: `yyyy-mm-ss HH:MM:SS`.
- Die Kopfzeile `SensorID` ist Pflicht, die `Unit`-Zeile mit den dazugehörigen IBK-Einheiten ist optional. Weitere Kopfzeilen können beliebig hinzugefügt werden.
- Es gibt beliebig viele Sensor-Namen, welche Sensorgruppen (im Beispiel oben G1 und G2) mit beliebigem Namen zugeordnet sein können. Die Spalten mit den Sensorgruppen haben keine Einheit und keine Zahlen bei den Datensamples (werden beim Lesen ignoriert, d.h. in den Datensample-Zeilen können in den Sensorgruppen Spalten beliebige Werte/Zeichenketten stehen, wie G1 und G2 im obigen Beispiel).
- Die Anzahl der Spalten in den Sample-Zeilen (Datensektion) muss *exakt* mit den Spalten in der `SensorID`-Kopfzeile übereinstimmen. Jede Sensor-Spalte muss einen Wert enthalten.

1.2 Workflow

1.2.1 Datenerfassung

Auf Messrechnern laufen Skripte/Programme, die automatisiert die angeschlossenen Datenlogger abfragen und die resultierenden Daten in das *Monitoring File Format* konvertieren. Alternativ kann die Datenerfassung auch manuell erfolgen, z.B. mit Temperatur/Luftfeuchte-Datenloggern, deren Daten ausgelesen und in das *Monitoring File Format* konvertiert werden.

1.2.2 Datentransfer

Die Messrechner sind mit dem Internet verbunden. In regelmäßigen Abständen verbinden sich die Clients mit dem Messdatenserver und kopieren neue Messdaten in das Eingangsverzeichnis. Bei Datenerfassung im Feld können verschiedene Mess-Clients (z.B. Raspi-Computer) individuell Messdaten an den Messdaten-Server schicken. Oder es kann

[1] Vogelsang, S. und Söhnchen, A.; *Monitoring Tools File Specification*, Technischer Bericht, 2016, TU Dresden, https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa-199034

ein zentraler Mess-Client-Rechner vor Ort eingerichtet werden, der die Messdaten der einzelnen Erfassungsrechner einsammelt und zentral an den Mess-Server schickt.

Alternativ können Messdaten auch manuell in das Eingangsverzeichnis auf dem Messdatenserver kopiert werden.

1.2.3 Datenprüfung und Archivierung

Auf dem Messdatenserver läuft nun in regelmäßigen Abständen das Prüfprogram, das *MonVerifyTool*. Dieses Programm prüft neu eingegangene Daten auf Korrektheit und Vollständigkeit. Weiterhin wird geprüft, ob Daten auch erwartungsgemäß geliefert wurden. Das Programm wird mit folgender Kommandozeile gestartet:

```
> python MonVerifyTools.py /path/to/projectRoot
```

Abbildung 1 zeigt den grundsätzlichen Ablauf des Testprogramms. Falls Fehler bei der Prüfung auftreten, werden fehlerhafte Dateien zur Prüfung beiseite gelegt und die Projektbearbeiter informiert. Die Unterprogramme sind im Kapitel 5 bzw. im Abschnitt 5.2 näher erläutert.

Korrekte Daten werden archiviert und können durch nachfolgende Post-Processing-Programme weiter verwendet werden. Damit die Originaldaten nicht verloren gehen (z.B. im Fall einer fehlerhaft angewendeten Korrekturvorschrift), werden die Rohdaten in diesem Fall zusätzlich noch abgelegt, wobei die Rohdaten mit verändertem Dateinamen (angehängtem `.original`) abgelegt werden.

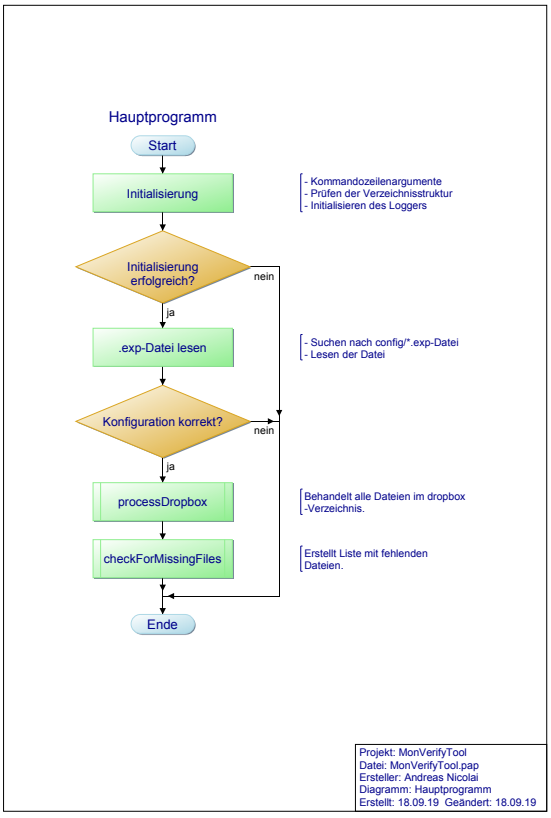

Abbildung 1: Programmablaufdiagramm für MonVerifyTool-Script

1.3 Grundlegende Annahmen und Festlegungen

- Server und Clients laufen in der gleichen Zeitzone mit gleich konfigurierten Zeiteinstellungen (ohne Sommerzeit etc.).
- Messdaten werden in Messprojekten verwaltet. Es kann mehrere (Messdatenerfassung-)Projekte auf dem Datenerfassungsserver geben, welche unabhängig voneinander bearbeitet, konfiguriert, reviewed werden können.
- Je Messprojekt werden Projektbearbeiter mit Zugriffsrechten ausgestattet (siehe Abschnitt 1.4).

- Je Messprojekt sollte es einen Mess-Client-Nutzer-Account. Werden mehrere externe Clients für ein Projekt verwendet, nutzen diese den gleichen Nutzeraccount (siehe Abschnitt 1.4). Es ist jedoch möglich, mehrere Mess-Client-Nutzer anzulegen, allerdings erhöht das den Verwaltungsaufwand, bringt keine weiteren Vorteile und ist deshalb nicht zu empfehlen.
- Das *MonVerifyTool*-Script läuft regelmäßig zu einer definierten Tageszeit, z.B. via `cron`-Job oder als Jenkins-Slave. Letzteres bietet den Vorteil, dass Fehler sofort im Jenkins-Dashboard erkennbar sind.
- Neue Messdaten werden vom Client auf den Server via `scp` kopiert. Damit während der Kopie nicht bereits lesend auf (teilweise) kopierte Daten zugegriffen wird, muss das Kopieren und Auswerten zeitlich getrennt erfolgen. Entprechend sind unterschiedliche Zeiten für den Datentransfer des Clients und das Ausführen des Prüfprogramms auf dem und Server einzustellen.
- Die erfassten Rohdaten werden unverändert aufbewahrt. Korrigierte Dateien werden entsprechend gekennzeichnet und Details der Korrekturen/Ergänzungen in einer Log-Datei vermerkt.

1.4 Datensicherheit und Zugriffskonzept

Messdaten müssen vertraulich behandelt werden. Auf den Messrechnern/Mess-Clients vor Ort haben nur Projektbearbeiter (d.h. Admins dieser Geräte) lokalen und entfernten Zugang.

Auf dem Messdatenserver werden alle projektspezifischen Daten in einer zugriffsgeschützten Verzeichnisstruktur verwaltet (siehe Abschnitt 2 zu Details zur Einrichtung dieser Verzeichnisstruktur und Vergabe von Zugriffsrechten).

Projektmitarbeiter erhalten einen Login für die Bearbeitung von Projekten, d.h. Erstellung und Anpassung von Konfigurationsdateien, Bearbeitung von zu prüfenden Daten und manueller Korrektur von fehlerbehafteten Dateien, sowie den Zugriff auf die archivierten Daten.

Für Mess-Clients werden je Projekt ein Login vergeben, welches jedoch nicht für SSH-Logins verwendet werden kann (keine Login-Shell). Mit diesen Zugangsdaten können lediglich neue Messdaten in das Projekteingangsverzeichnis `dropbox` kopiert werden (siehe Abschnitt 1.5). Werden mehrere Clients für das gleiche Projekt verwendet, so teilen sich diese diesen Zugang.

1.5 Verzeichnisstruktur

In jedem Projektverzeichnis wird folgende Verzeichnisstruktur abgebildet:

```
+ projectRoot
  + config                  - Verzeichnis für Konfigurationsdateien
  + dropbox                 - enthält Eingangsdaten (z.B. von den Mess-Clients)
  + review                  - Verzeichnis für Dateien, welche der Nutzer prüfen/korrigieren muss
  + log                     - Verzeichnis mit projektspezifischen Logdateien
  + archive                 - Verzeichnis mit archivierten Daten
  + bypass                  - Verzeichnis mit ungesehen/unverändert archivierten Daten
```

`config` Enthält projektspezifische Konfigurationsdateien (`*.exp`, `*.ref` und `*.phy`)

`dropbox` Eingangsdatenverzeichnis; hier schreiben die Mess-Clients die Rohdaten hinein. Es können Unterverzeichnisse zur Strukturierung der Daten angelegt werden. *Dateinamen müssen innerhalb der gesamten Verzeichnisstruktur unterhalb von* `dropbox` *eindeutig sein.*

`review` Verzeichnis, in dem fehlerhafte Datendateien abgelegt werden (unter Beibehaltung der Verzeichnisstruktur des `dropbox`-Verzeichnisses), sodass manuelle Korrekturen möglich sind

`log` Verzeichnis mit Logdateien (siehe Abschnitt 6.2)

`archive` Verzeichnis mit Archivdateien; unter Beibehaltung der Verzeichnisstruktur des `dropbox`-Verzeichnisses werden hier alle geprüften/konvertierten/korrigierten Daten abgelegt.

`bypass` Verzeichnis für Dateien, welche ohne Sichtung entsprechend definierter Bypass-Regeln direkt von `dropbox` nach `bypass` verschoben werden.

1.6 Namenskonventionen für Messdatendateien

Entsprechend der Spezifikation wird folgendes Dateinamenschema zwingend vorgeschrieben:

```
<prefix>_<yyyy-mm-dd>_<hh-MM-ss>.csv
```

Jede Datei hat also exakt zwei Unterstrich-Zeichen '_' im Namen. Der Datums und Zeitstempel hat einschließlich des Unterstrich-Zeichens exakt die Länge von 19 Zeichen. Beispiele:

```
demo2_2019-08-08_15:44:00.csv
zvw-hh-wg1-2s_2015-07-22_00-00-00.csv
```

Alles vor dem ersten Unterstrich muss ein-eindeutig sein und wird zur Identifikation der Datei-Regeln verwendet (siehe Abschnitt 4.1). Im obigen Beispiel sind das folgende Dateinamenspräfixe:

```
demo2_
zvw-hh-wg1-2s_
```

Der nachfolgende Unterstrich dient der Unterscheidung von Namen, welche in anderen Präfixen enthalten sind, z.B. um folgende Dateien eindeutig zu unterscheiden:

```
zvw-hh-wg1-2s_
zvw-hh-wg1_
```

2 Einrichtung des Messdaten-Servers

2.1 Zentrale oder dezentrale Einrichtung

Unter *zentraler* Einrichtung wird verstanden, dass es einen zentralen Messdatenerfassungsserver gibt, auf dem mehrere Messprojekte gleichzeitig verwaltet werden. Auf diesem Server verwaltet der Admin die Projekterstellung, Gruppen und Nutzer. Eingerichtete Nutzer (Manager oder Messclients, siehe unten) arbeiten auf diesem Server mit den vom Admin bereitgestellten Verzeichnissen.

Unter *dezentraler* Einrichtung wird eine lokale Installation, z.B. im eigenen Home-Verzeichnis verstanden. Diese kann ohne Hilfe des Admins erfolgen, da keine Gruppen oder Nutzer angelegt werden müssen (entspricht der typischen Einrichtung unter Windows).

2.2 Einrichtung eines neuen Projekts - dezentrale Einrichtung

Die dezentrale Einrichtung ist sehr einfach. Man führt das Script `createMonToolProject.sh` aus und gibt ein Unterverzeichnis in einem vom Nutzer schreibbaren Verzeichnis an. Die Kommandozeilenargumente des Scripts sind:

```
>./createMonToolProject.sh <Nutzer-Gruppe> <Nutzer-Gruppe> <Nutzer-Login> <Projektbasisverzeichnis>
```

also beispielsweise:

```
>./createMonToolProject.sh andreas andreas andreas /home/andreas/messprojekt1
```

Hinweis: Unter Linuxdistributionen wie Ubuntu wird zu jedem Loginnamen eine gleichnamige Gruppe angelegt, also im obigen Beispiel gibt es im System eine Gruppe `andreas` und einen Login-Nutzernamen `andreas`.

2.2.1 Dezentrale Einrichtung unter Windows

Es wird einfach die Verzeichnisststruktur aus Abschnitt 2.3.4 erstellt, allerdings ohne irgendwelche Zugriffsrechte anzupassen.

2.3 Einrichtung eines neuen Projekts - zentrale Einrichtung

Die nachfolgend erläuterte Einrichtung der Gruppen und Nutzer ist nur für dezentrale Einrichtung notwendig.

2.3.1 Gruppen

Um den Zugriff auf die Verzeichnisse sinnvoll zu beschränken, werden für jedes Messprojekt zwei Gruppen eingerichtet:

- Gruppe für Projektmanager (mit Zugriff auf Konfigurationsdateien und Messdaten, können Fehler korrigieren, Logdateien einsehen etc.), beispielsweise `GrpMessProjekt1Manager`
- Gruppe für Messclients (haben kein Home-Verzeichnis, dürfen ausschließlich via scp Daten in das Projekt-Eingangsverzeichnis `dropbox` kopieren), beispielsweise `GrpMessProjekt1Client`

2.3.2 Nutzeraccounts

Es werden typischerweise 3 Nutzer eingerichtet:

1. Nutzer für die Messclients : nur Mitglied ihrer eigenen Gruppe und der Messprojekt-Client-Gruppe (also bspw. `GrpMessProjekt1Client`)
2. Nutzer für die Projektmanager (einer oder mehrere Projektverwalter) : sind Mitglied der Gruppen für Manager und Clients (also bspw. `GrpMessProjekt1Manager` und `GrpMessProjekt1Client`)
3. Nutzer für den Dienst, der automatisiert das Prüfskript ausführen soll (dafür kann auch ein Projektmanageraccount verwendet werden) : ist ebenfalls Mitglied beider Gruppen

2.3.3 Beispiel

Nachfolgend werden die vom Systemadmin auszuführenden Schritte zum Einrichten der Gruppen und Nutzer eines Projekts mit dem Namen `MessProjekt1` aufgeführt:

Gruppen einrichten

```
> sudo groupadd GrpMessProjekt1Manager
> sudo groupadd GrpMessProjekt1Client
```

Nutzer einrichten

Einrichten eines neuen Mess-Client-Nutzers ohne eigenes Home-Verzeichnis:

```
> sudo adduser --system --no-create-home mp1client
```

Hinweis: Unter Ubuntu wird zusätzlich noch eine Gruppe `mp1client` eingerichtet. Man könnte diese Gruppe auch anstelle der im Beispiel verwendeten Gruppe `GrpMessProjekt1Client` nehmen.

Einrichten der Projektmanager und Projekt-Service Nutzer:

```
> sudo adduser mp1manager
> sudo adduser mp1service
```

Gruppen zuordnen

Nun wird die Datei `/etc/group` vom Admin editiert, und in der Gruppe `GrpMessProjekt1Client` alle drei neu erstellten Nutzer eingetragen. In der Gruppe `GrpMessProjekt1Manager` stehen nur die Nutzer `mp1manager` und `mp1service` drin.

Die `/etc/group` enthält nach diesen Schritte beispielsweise folgende Zeilen:

```
.....
GrpMessProject1Manager:x:1006:mp1manager,mp1service
GrpMessProject1Client:x:1007:mp1manager,mp1service,mp1client
mp1client:x:999:
mp1manager:x:1001:
mp1service:x:1002:
....
```

2.3.4 Verzeichnisstruktur und Zugriffsrechte

Automatische Erstellung des Projektverzeichnissen Für das Einrichten der Projektverzeichnisstruktur und Setzen der Zugriffsrechte kann das Script `createMonToolProject.sh` ausgeführt werden. Syntax:

```
>sudo ./createMonToolProject.sh <Projektbearbeiter-Gruppe> <Mess-Client-Gruppe> <SRV-User> <
    ↪ Projektbasisverzeichnis>
```

Also beispielsweise:

```
>sudo ./createMonToolProject.sh GrpMessProjekt1Manager GrpMessProjekt1Client mp1service /srv/data/
    ↪ MessProjekt1
```

dabei muss das übergeordnete Verzeichnis `/srv/data` erstellt sein und von normalen Nutzern erreichbar sein. Typischerweise reicht:

```
>sudo mkdir /srv/data
```

Manuelle Erstellung des Projektverzeichnissen und Vergabe von Zugriffsrechten Zuerst wird das Projektbasisverzeichnisses, z.B. unter `/srv/data/MessProjekt1` erstellt, `Verzeichnis bekommt Rechte rwxr_x___` und *mp1service:GrpMessProjekt1Client* als Nutzer/Group.

Dann wird folgende Verzeichnisstruktur mit den jeweils angegebenen Nutzernamen und Gruppen sowie Rechten erstellt:

```
MessProject1/       mp1service:GrpMessProject1Client        rwx r_x ___
    archive         mp1service:GrpMessProject1Manager       rwx r_x ___
    bypass          mp1service:GrpMessProject1Manager       rwx r_x ___
    config          mp1service:GrpMessProject1Manager       rwx rwx ___
    dropbox         mp1service:GrpMessProject1Client        rwx _wx ___
    log             mp1service:GrpMessProject1Manager       rwx rwx ___
    review          mp1service:GrpMessProject1Manager       rwx rwx ___
    status          mp1service:GrpMessProject1Manager       rwx r_x ___
```

2.3.5 Einrichten des cron-Jobs/Jenkins-Slave für Starten des Scripts

Das MonVerifyTool-Skript wird regelmäßig ausgeführt, wobei die Frequenz projektabhängig eingestellt werden kann. Werden z.B. täglich neue Messdaten erwartet, sollte das Skript ebenfalls täglich laufen (häufiger wäre nicht sinnvoll).

Das Script wird wie folgt aufgerufen:

```
>python /path/to/scripts/MonVerifyTool.py /srv/data/MessProjekt1
```

Das Script sollte unter dem Nutzer `mp1service` laufen (kann aber auch manuell von jedem Projektmanager ausgeführt werden).

3 Einrichten von Messerfassungs-Clients

Es wird vorausgesetzt, dass der Admin-Nutzer einen private/public Key für den Messclient-Nutzer eingerichtet hat.

Das Kopieren der neuen Messdaten erfolgt via scp, beispielsweise der Datei `wg1-rh4-2_2019-08-09_00:00:00.csv` an einen Server unter der IP-Adresse `1.2.3.4`:

```
> scp -i /path/to/id_rsa wg1-rh4-2_2019-08-09_00:00:00.csv mp1client@1.2.3.4:/srv/data/MessProject1/
    ↪ dropbox
```

Das zusätzliche Argument `-i /path/to/id_rsa` mit dem Pfad zur privaten Schlüsseldatei ist notwendig, da Mess-Client-Nutzer üblicherweise kein Home-Verzeichnis haben und daher der Standard-Suchpfad `~/.ssh/id_rsa` für die private Schlüsseldatei nicht gültig ist. Dabei ist `/path/to/id_rsa` mit einem Pfad zu einer ausschließlich vom Mess-Client-Nutzer lesbaren privaten Schlüssel-Datei zu ersetzen.

4 Projektkonfiguration

Das MonVerifyTool-Skript führt verschiedenen Prüfungen durch. Diese werden in den unterschiedlichen Konfigurationsdateien festgelegt. Nachfolgend sind die einzelnen Konfigurationsdateien beschrieben und die darin definierten Tests.

4.1 Projekt-Eingangsprüfungs-Konfiguration / Expectation-Dateien (*.exp)

Ein Projekt hat exakt eine Konfigurationsdatei mit den Eingangs-Prüfungen. Diese hat die Endung `exp` (von *Expectations*) und liegt im `config`-Unterverzeichnis. In dieser Datei wird festgelegt, welche Mess-Dateien in diesem Projekt erwartet werden. Die Datei ist im JSON-Format geschrieben und hat beispielsweise folgenden Inhalt:

```
{
    "ExpectedFiles" : [
        ["p1.1/demo2_","IBK_TimeSeries",0,0,100,1,"p1.1/demo1.ref","p1.1/demo1.phy",120,1],
        [...],
        ["p1.5/demo2_","IBK_TimeSeries",0,0,100,1,"p1.1/demo1.ref","p1.1/demo1.phy",120,1]
    ],
    "BypassFiles" : [
        "TestHaus/ignored_"
    ]
}
```

4.1.1 Erwartete Dateien

Das Element `'ExpectedFiles'` definiert in einer Liste alle Dateien, welche regelmäßig im Projekt erwartet werden. Jeder Eintrag der Liste ist selbst eine Liste mit den in Tabelle 1 definierten Elementen.

Spaltenindex	Typ	Beschreibung
0	Dateipräfix	Der Präfix des Dateinamens, d.h. der komplette relative Pfad zum `dropbox`-Verzeichnis einschließlich des Dateinamens bis und inklusive des ersten Unterstrichs. Beispielsweise: `p1.1/demo2_` (siehe auch Abschnitt 1.6). Pfade außerhalb des `dropbox`-Verzeichnisses sind nicht erlaubt.
1	Testgruppe	Ein Bezeichner einer Testgruppe (siehe Tabelle 2 für eine Übersicht der Zuordnung von Einzeltests zu Testgruppen). Durch Angabe der Testgruppe können bestimmte Prüfungen ausgeschlossen werden. Ist das Feld eine leere Zeichenkette (d.h. `""`), wird `'IBK_TimeSeries'` angenommen.
2	Minimale Größe in Bytes	Die minimale bzw. exakte Größe der erwarteten Datei in Bytes. Ein Wert von Null deaktiviert den Größentest (die Unterscheidung zwischen Größenbereichstest und Test auf exakte Dateigröße wird durch die Zuordnung zu der Testgruppe festgelegt, siehe Tabelle 2).
3	Maximale Größe in Bytes	Die maximale Größe der erwarteten. Ein Wert von Null deaktiviert den Test auf maximale Größe.
4	Zeilenzahl	Die Anzahl der Daten-Zeilen, die in einer vollständigen Datei nominal erwartet werden. Ein Wert von 0 deaktiviert den Zeilenzahltest.
5	Zeilenzahltoleranz	Die maximal erlaubte Abweichung (nach unten oder oben) beim Test auf Zeilenzahl.
6	Kopfzeilenreferenz-Datei	Ein Pfad (relativ zum `config`-Verzeichnis) zu einer Datei mit den Kopfzeilen der erwarteten Datei. Beispielsweise: `p1.1/demo1.ref`. Pfade außerhalb des `config`-Verzeichnisses sind nicht erlaubt. Üblicherweise wird diese Datei erstellt, indem man die erste generierte Messdaten-Datei kopiert, alle Datenzeilen entfernt und als `.ref`-Datei abspeichert. Eine fehlende Pfadangabe (d.h. leere Zeichenkette `""`) deaktiviert den Kopfzeilen-Vergleichstest.
7	Datei für Inhaltsprüfungen	Ein Pfad (relativ zum `config`-Verzeichnis) zu einer Datei mit Definitionen der inhaltlichen Prüfungen. Beispielsweise: `p1.1/demo1.phy`. Pfade außerhalb des `config`-Verzeichnisses sind nicht erlaubt. Eine fehlende Pfadangabe (d.h. leere Zeichenkette `""`) deaktiviert die inhaltlichen Werteprüfungen.
8	Mess-Interval	Erwartete Messfrequenz in Sekunden. Wenn dieser Wert nicht 0 ist, werden die Zeitstempel aller Datenzeilen daraufhin überprüft, ob die erwartete Messfrequenz eingehalten wurden. *Achtung: Der zeitliche Abstand zwischen dem letzten Zeitstempel der zuletzt archivierten Datei und dem ersten Zeitstempel der aktuell geprüften Datei kann an dieser Stelle nicht überprüft werden.*
9	Mess-Interval-Toleranz	Toleranz (+/-) für Messfrequenz in Sekunden. Wenn dieser Wert nicht 0 ist, muss die Länge jedes Messintervalls im Toleranzbereich liegen (Min: Mess-Interval - Toleranz; Max: Mess-Interval + Toleranz).

Tabelle 1: Konfigurationseinträge für jede Datei im `'ExpectedFiles'` Attribut.

4.1.2 Bypass-Dateien

Liefert ein Mess-Client Daten in anderen Dateiformaten ab, oder zusätzliche Dateien mit Metainformationen, so kann man diese vom Skript direkt in das bypass-Verzeichnis verschieben lassen, ohne die Eingangsprüfungen durchzuführen. Dazu muss in der `exp`-Datei das Element `'BypassFiles'` definiert werden. Dieses Element ist eine Liste von Datei-Suchmasken. Die Dateinamen oder Pfade werden als relative Pfade zum `dropbox`-Verzeichnis interpretiert. Jede neue

Datei, welche dem Suchmuster entpricht, wird unter Beibehaltung der Verzeichnisstruktur im `dropbox`-Verzeichnis in das `bypass`-Verzeichnis verschoben.

Eine Datei gilt als korrekt erkannt, wenn die der Dateipfad mit einem der definierten Suchmuster *beginnt*.

Beispielsweise wird die Datei:

```
dropbox/TestHaus/wp2_2019-08-09_extra_daten.csv
```

durch die Suchmaske

```
TestHaus/wp2
```

gefunden und in das Verzeichnis

```
bypass/TestHaus/wp2_2019-08-09_extra_daten.csv
```

verschoben.

5 Prüfungen

5.1 Daten-Eingangsprüfungen

Jede neue Mess-Datei wird vom Skript entsprechend der Datei-Konfiguration überprüft. Wird ein Fehler festgestellt, wird die geprüfte Datei in das `review`-Verzeichnis verschoben und ein Log-Eintrag geschrieben.

5.1.1 Grundlegende Vorgehensweise

1. Es wird geprüft, ob die Datei durch eine Suchmaske in der `'BypassFiles'`-Liste gefunden wird. Falls ja, wird die Datei in das `bypass`-Verzeichnis verschoben.
2. Es wird geprüft, ob der Dateiname der Datei dem definierten Schema entspricht. Falls nicht, wird die Datei in das `review`-Verzeichnis verschoben und als „BadFilename“ klassifiziert (siehe Tabelle 3).
3. Es wird geprüft, ob für die Datei ein passender Eintrag in der `'ExpectedFiles'`-Liste gefunden wird. Falls nicht, wird die Datei in das `review`-Verzeichnis verschoben und als „NotExpected“ klassifiziert (siehe Tabelle 3).
4. Je nach Testgruppe (aus der ExpectedFiles-Definition) werden nun die für diese Gruppe gültigen Tests durchgeführt (siehe Tabelle 2).

Testgruppe	Kopfzeilen korrekt	Dateigröße stimmt exakt	Dateigröße im Toleranzbereich	Anzahl Datenzeilen wie erwartet	Mess-Interval eingehalten
IBK_Mon_DHT22	x	-	x	x	x
IBK_Mon_1Wire	x	x	-	x	x
IBK_Mon_WinStat	x	-	-	-	x
IBK_Mon_ConCom	x	-	x	x	x
IBK_TimeSeries	x	-	x	x	x
IBK_EventData	x	-	-	-	-
IBK_Custom	-	-	-	-	-

Tabelle 2: Testgruppen

Bei allen folgenden Tests wird im Fehlerfall die Datei in das `review`-Verzeichnis verschoben und entsprechend klassifiziert (siehe Tabelle 3).

5.1.2 Eingangstests

Dateinamenstest Bei diesem Test wird zunächst geprüft, ob der Dateiname dem geforderten Schema entspricht (einschließlich Format des Datums/Zeitstempels). Danach wird geprüft, ob die Uhrzeit 00:00:00 ist, was derzeit die erwartete Erstellungszeit der Datendatei ist.
Hinweis: Sollten Datendateien in anderen Zeitfrequenzen erstellt werden, beispielsweise alle 12 Stunden, muss das Skript hinsichtlich dieses Tests erweitert werden.

Kopfzeilentest Bei diesem Test wird geprüft, ob die Kopfzeilen der zu prüfenden Datei *exakt* mit den in der `ref`-Datei hinterlegten Kopfzeilen übereinstimmen. Ist keine `ref`-Datei angegeben, wird dieser Test übersprungen.

Dateigrößentest Die angegebene Dateigröße muss je nach Testgruppe entweder exakt stimmen, oder zumindest im erwarteten Toleranzband liegen, d.h. *Größe - Toleranz < Dateigröße < Größe + Toleranz*. Ist keine Größe angegeben oder = 0 wird der Test übersprungen.

Datenzeilentest Die Anzahl der Datenzeilen in der Datei wird mit der erwarteten Datenzeilenanzahl (einschließlich Toleranz) verglichen. Ist keine Zeilenzahl definiert oder = 0 wird der Test übersprungen.

Mess-Intervalltest Die Zeit zwischen jeweils aufeinanderfolgenden Datensamples wird mit dem definierten Mess-Interval verglichen. Sofern ein Mess-Interval angegeben wurde und nicht 0 ist, müssen die Messabstände *exakt* mit dem erwarteten Mess-Interval übereinstimmen. Im Fehlerfall wird in der Logdatei für jedes unstimmige Intervall der vorangegange und nachfolgende Zeitstempel angegeben.

Test	Log-Klassifizierung	Mögliche Fehlersituation
Dateiname	BadFilename	Dateiname entspricht nicht dem definierten Schema.
Dateiname	BadFilename	Zeitstempel der erwarteten Datei entspricht nicht dem erwarteten Zeitpunkt (aktuell wird Uhrzeit 00:00:00 erwartet) .
Datei erwartet	NotExpected	Für diese Datei gibt es keine Definition in der `'ExpectedFiles'`-Liste des Projekts.
Zugriffsfehler (Lesen)	AccessDenied	Kann die Datei im `dropbox`-Verzeichnis nicht lesen. Zugriffsrechte fehlen.
Schreibfehler	AccessDenied	Kann die Datei im `dropbox`-Verzeichnis nicht in ein anderes Verzeichnis verschieben. Zugriffsrechte fehlen.
Dateigröße	EmptyFile	Datei ist leer.
Dateigröße	FileSizeMismatch	Dateigröße stimmt nicht exakt, bzw. liegt außerhalb des Toleranzbereichs.
Kopfzeile	InvalidHeader	Kopfzeilen der Datei stimmen nicht mit der Referenz-Kopfzeilen-Datei überein.
Datenzeilen	SampleCountMismatch	Anzahl der Datenzeilen ist außerhalb des erwarteten Bereichs. Wird auch verwendet, wenn die Datei komplett ohne Datensamples ist (nur Header). Wird auch ausgelöst, wenn die Trennzeile zwischen Kopf- und Datensektion fehlt, und die Kopfzeilenprüfung nicht durchgeführt wurde.
Datenzeilen	ColumnCountMismatch	Anzahl der Spalten in einer Datenzeile entspricht nicht der durch die SensorID-Kopfzeile vorgegebenen Spaltenzahl.
Datenzeilen	InvalidTimeStamp	Das Format des Zeitstempels eines Datensamples stimmt nicht.
Mess-Intervall	InvalidSamplingInterval	Der Abstand zwischen zwei Datensamples/Zeitstempeln entspricht nicht dem erwarteten Mess-Intervall.

Tabelle 3: Tests und entsprechende Logeinträge.

5.2 Vollständigkeitsprüfungen

Das Skript erwartet täglich neue Datendateien, wie in der `exp`-Datei festgelegt wurde. D.h., bei korrekter Funktion müssen nach Ablauf des Skripts im `archive`-Verzeichnis alle Dateien mit Zeitstempel bis zum heutigen Tag liegen (d.h. MonVerifyTool-Skript sollte daher stets *nach* dem Empfangen neuer Daten gestartet werden).

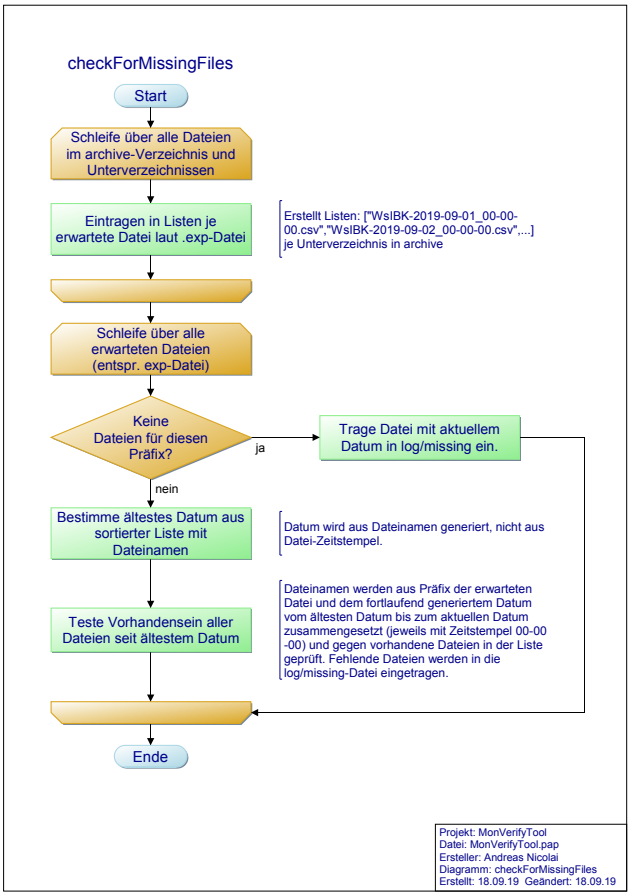

Abbildung 2: Programmablaufdiagramm für die „Fehlende Dateien“-Prüfung

Für jede in der `exp`-Datei aufgeführte erwartete Datei (siehe Abschnitt 4.1) wird eine Verzeichnisliste aus dem Inhalt des `archive`-Verzeichnisses gelesen und alle Zeitstempel extrahiert (siehe Ablaufdiagramm in Abbildung 2). Dann wird vom ältesten Zeitstempel bis zum aktuellen Datum jeder Tageszeitstempel auf Existenz geprüft. Fehlt ein Zeitstempel, so wird die entsprechende Datei in die log-Datei Der Umgang mit Encoding-Problemen ist zu diskutieren. Hier findet entweder die Anzahl der Kopfzeilen aus der exp Datei Anwendung oder die Anzahl der Headerzeilen muss ausgezählt werden. `log/missing` eingetragen.

Beispiel

```
archive/WsIBK_2019-09-15_00-00-00.csv
archive/WsIBK_2019-09-13_00-00-00.csv
archive/WsIBK_2019-09-12_00-00-00.csv
archive/WsIBK_2019-09-11_00-00-00.csv
archive/T2s/T2IBK_2019-09-15_00-00-00.csv
archive/T2s/T2IBK_2019-09-14_00-00-00.csv
archive/T2s/T2IBK_2019-09-12_00-00-00.csv
```

Die Datei `log/missing` würde nun folgende Zeilen enthalten:

```
WsIBK_2019-09-14_00-00-00.csv
T2s/T2IBK_2019-09-13_00-00-00.csv
```

Die Dateien mit Präfix `T2s/T2IBK` beginnen erst am 2019-09-12, weswegen die Datei `T2s/T2IBK_2019-09-11_00-00-00.csv` nicht als fehlend aufgeführt wird (obwohl die Datenreihe mit Präfix `WsIBK` bereits 2019-09-11 beginnt).

Sind alle erwarteten Dateien vorhanden, so wird die Datei `log/missing` gelöscht. Die Existenz der `log/missing`-Datei mit eingetragenen Fehldateien führt zu einem Rückgabewert von 1 des MonVerifyTool-Skripts.

6 Script-Ausführungsergebnisse/Ergebnisdateien

6.1 Rückgabewert

Das Skript liefert einen Rückgabewert von 0, wenn kein Fehler aufgetreten ist. Im Falle eines Fehlers wird 1 zurückgeliefert. In diesem Fall stehen im Error-Log die entsprechend aufgetretenen Fehler, und ggfs. in der Datei `log/missing` eine Liste mit fehlenden Dateien.

6.2 Log-Dateien

Es gibt drei Log-Dateitypen im Unterverzeichnis `log`.

6.2.1 Process-Log

In der Datei `log/processed` werden allgemeine Ausgaben des Skripts abgelegt. Darin wird ersichtlich, welche Dateien erfolgreich archiviert wurden.

Jede Zeile in der Logdatei hat das folgende Format:

```
<Zeitstempel>\t<Kategorie>\t<Datei>
```

wobei `\t` ein Tabulatorzeichen ist. Beispielsweise:

```
2019-08-12 15:06:15  Archiving  TestHaus/demo2_2017-10-23_23-59-00.csv
2019-08-12 15:07:41  Error      TestHaus/sw-11-we-0-00-BTM_2016-03-02_00-00-00.csv
2019-08-12 15:08:20  Error      TestHaus/demo2_2017-10-23_23-57-00.csv
2019-08-12 15:08:20  Bypassing  TestHaus/ignored_TestFile.txt
```

Die folgenden Kategorien werden verwendet:

Error Zeigt einen Fehler bei einer Datei an (Eingangsprüfung fehlgeschlagen etc.). Details dazu stehen im Error-Log.

Archiving Die Datei wurde erfolgreich archiviert (ins `archive`-Verzeichnis verschoben).

Bypassing Die Datei wurde ins `bypass`-Verzeichnis verschoben.

Corrected Eine Korrektur wurde an der Datei angewendet. Zumeist gefolgt von zwei Logeinträgen vom Typ 'Archiving', einer für die originale Datei und einer für die korrigierte Datei.

Hinweis: Die Process-Log-Datei kann in einer Tabellenkalkulation wie LibreOffice direkt eingeladen werden und dann über `Daten->AutoFilter` in eine Tabelle mit Kategorie-Filterfunktion umgewandelt werden. Auf diese Weise lassen sich übersichtliche Listen der abgearbeiteten Datendateien generieren.

6.2.2 Error-Log

In der Datei `log/errors_<Zeitstempel>` sind fehlgeschlagene Tests je Datei detailliert aufgeführt. Diese Datei kann/-sollte vom Projektbearbeiter nach erfolgreicher Korrektur des Fehlers gelöscht werden. Dadurch kann die Existenz der Datei als Indikator für Fehlerzustände genutzt werden. Gleichzeitig ist der Inhalt der Datei sinnvoll für die Darstellung auf der Mess-Projekt-Informations-Webseite (Dashboard).

Jede Zeile in der Logdatei hat das folgende Format:

```
<Kategorie>\t<Datei>\t<Optionale detaillierte Beschreibung>
```

wobei `\t` ein Tabulatorzeichen ist. Beispielsweise:

```
NotExpected              TestHaus/demo2_2017.csv
InvalidHeader            TestHaus/sw-11-we-0-00-BTM_2016-03-02_00-00-00.csv
FileSizeMismatch         TestHaus/demo2_2017-10-23_23-57-00.csv            File size 65378 out of range
    ↪  [64000..65000].
InvalidSamplingInterval  TestHaus/demo3_2017-10-23_23-57-00.csv            Interval between 2019-08-09
    ↪ 12:00:00 and 2019-08-09 15:00:05 too long (10805 s but expected 120 s).
```

6.2.3 Fehlende-Dateien

In der Datei `log/missing` stehen alle Dateien, die sich eigentlich bereits im `archive`-Verzeichnis befinden sollten (siehe Abschnitt 5.2).

7 Fehlerbehandlung

Sobald eine Eingangsprüfung oder inhaltliche Prüfung fehlgeschlagen ist, wird die jeweilige Messdatendatei in das `review`-Verzeichnis verschoben. Je nach Fehlertyp sind folgende Fehlerbehandlungen sinnvoll.

7.1 InvalidHeader - Nur Zeichenkodierung

Dies ist ein Fehler bei der Konfiguration des Messclients.

1. Messclient-Fehler beheben
2. Zeichenkodierung in Dateien korrigieren. Falls UTF-8 verwendet wird (empfohlen), kann das Skript `iconv_all.sh` im `review`-Verzeichnis ausgeführt werden. Es konvertiert alle nicht-UTF-8-Dateien in UTF-8 Zeichenkodierung. Danach können alle konvertierten Dateien wieder in das `dropbox`-Verzeichnis verschoben werden.

7.2 InvalidHeader - Fehler ohne Einfluss auf Datenspalten

Dies ist ein Feher bei der (Um-)Konfiguration des Messclients.

1. Messclient-Fehler beheben/neu konfigurieren
2. Header aller betroffenen Dateien manuell korrigieren (Inhalt des Referenzheaders kopieren). Danach können alle korrigierten Dateien wieder in das `dropbox`-Verzeichnis verschoben werden.

7.3 InvalidHeader - SensorenIDs stimmen nicht

Beispiel:

```
InvalidHeader          WsIBK_2019-08-21_12-26-25.csv                    Header line mismatch:
Expected: Channel,WsIBK_SerAhl_G01,M00,M01,M02,M03,M04,,,,,,,M12,M13,M14,,,,,,,M22,,,,,,,,,,M33,M34
Current:  Channel,WsIBK_SerAhl_G01,M00,,,,M04,,,,,,,,,,,,,,,,,,,,,,,,,,,,M33,
```

Dieser Fehler liegt an einer falschen/Neu-Konfiguration des Messclients bzw. an ausgefallenen Sensoren. Hier ist eine intensivere Fehlersuche notwendig, bzw. eine Neuerstellung der Datei.

7.4 EmptyFile

Falls eine Datei komplett leer ist, wurde vermutlich die Verbindung bei der Übertragung beendet. In diesem Fall muss geprüft werden, ob die Datei zu einem späteren Zeitpunkt noch einmal übertragen werden kann (z.B. wenn der Messclient versucht, Dateien solange zu übertragen, bis die Datei komplett am Ziel angekommen ist). In diesem Fall kann die Datei gelöscht werden. Falls die Datei gar nicht existiert hat, bzw. bereits auf dem Messclient leer war, muss die Datei als *fehlende Datei* vermerkt werden.

7.5 FileSizeMismatch

Grundsätzlich können sich die Dateigrößen der zu erwartenden Dateien geringfügig unterscheiden, z.B. wenn Luftdrücke im Bereich 1003 hPa und 998 hPa gemessen wurden, so haben die Dateien je Zeile 1 Byte Größenunterschied. Ähnlich ist dies bei allen anderen Zahlen möglich. Daher ist die Dateigröße als Bereich anzugeben, sofern kein Festzahlenformat mit exakter Größe im Messclient konfiguriert ist.

Hinweis: Um einen ungefähren Überblick über die sinnvollen Bereichsgrenzen für den Größencheck zu erhalten, kann das Script `fileSizeHistogram.py` ausgeführt werden. Es liefert für alle Dateien eines Verzeichnisses ein Histogramm der Dateigrößen und die min/max-Werte der Dateigrößen.
Beispiel:

```
> python fileSizeHistogram.py /path/to/project/dropbox
```

Falls die Dateigröße dennoch außerhalb des zulässigen Bereichs liegt, fehlen vermutlich Daten. Da der Header vor der Dateigröße geprüft wird, können Größenunterschiede nicht auf unterschiedliche Header zurückgeführt werden und es muss an den Daten liegen.

Hinweis: Ein Grund für fehlerhafte Dateigrößen kann das Neustarten des Messsystems sein. In diesem Fall wird je nach Konfiguration eine neue Datei erstellt, und diese ergänzt dann die bisherige Datei mit Zeitstempel 00:00:00. Solche getrennten Messdaten können mit dem Script `mergeFiles.py` einfach zusammengefasst werden. Als Argument übergibt man den Pfad zu einer der aufgeteilten Dateien und gibt den Dateinamen bis einschließlich des Datums an, jedoch ohne Zeitstempel und Dateierweiterung.
Beispiel:

```
> python mergeFiles.py /path/to/project/review/WsIBK_2019-09-11
```

Durch diesen Aufruf werden beispielsweise die Dateien `WsIBK_2019-09-11_00-00-00.csv` und `WsIBK_2019-09-11_12-22-05.csv` zusammengefasst und in die Datei `WsIBK_2019-09-11_00-00-00.csv` geschrieben (die originalen Dateien werden umbenannt, wobei an den jeweiligen Dateinamen `.orig` angehängt wird).