FAKIN Projektdokumentation

Hauke Sonnenberg

2018-06-04

Contents

Part I

FAKIN-Dokumentation

Chapter 1

Einführung

Hier wollen wir die Ergebnisse unseres Forschungsprojekts FAKIN (Forschungsdatenmanagement an kleinen Instituten) veröffentlichen.

Da wir viele, zum Teil ganz unterschiedliche Projekte bearbeiten, sind auch unterschiedliche Bereiche des Forschungsdatenmanagements betroffen.

Wir wollen hier zunächst versuchen, Fallbeispiele herauszuarbeiten, in die sich die verschiedenen Projekte einordnen lassen. Für jedes Fallbeispiel wollen wir dann Gemeinsamkeiten in Bezug auf Anforderungen und Problemstellungen herausarbeiten und für diese dann existierende Lösungen einsammeln oder gezielt neue Lösungen entwickeln. Diese Lösungen sollen dann in zukünftigen Projekten, die sich den jeweiligen Fallbeispielen zuordnen lassen, angewendet werden.

Dazu brauchen wir Eure Mitarbeit! Wer kann Beiträge zur Beschreibung von Problemfällen und eigenen Lösungen liefern?

Chapter 2

Projektbeschreibungen

2.1 Projektgruppen

Nach welchen Kriterien können wir unsere Projekte gruppieren, wie können wir sie grob klassifizieren?

Was folgt daraus in Bezug auf das Datenmanagement?

Wenn wir Ähnlichkeiten finden, können wir in Zukunft gezielt auf die Lösungen, die in ähnlichen Projekten erzielt (und beschrieben!) wurden, zurückgreifen.

Mögliche Gruppierung nach:

- Abteilung: Grundwasser, Oberflächengewässer und Kanal, Abwasser
- Finanzierung: Sponsoring, EU-Projekt, BMBF-Projekt, Auftrag
- Anzahl Partnern: große Projekte mit vielen Partnern, kleine Projekte mit wenigen Partnern

Im folgenden fangen wir einfach mal an und versuchen, die Projekte, an denen wir mitgearbeitet haben, kurz zu beschreiben und zu charakterisieren. Wenn wir im Rahmen der Projekte bestimmte Lösungen erarbeitet haben, beschreiben wir diese kurz unter Problembereiche oder Konkrete Aufgaben und Lösungen.

Aufbauend auf dieser ersten Aufstellung von Projekten, Aufgabenbereichen und Lösungen wollen wir dann mit Euch zusammen weitere Projekte in die Betrachtung mit einbeziehen. Wir wollen gemeinsam die Charakterisierung und Gruppierung und natürlich insbesondere die Aufgabenbeschreibungen und Lösungen erweitern und verbessern.

2.2 Projekt: Spree2011 (2007)

- Meine Diplomarbeit
- eigene Messungen über einen begrenzten Zeitraum
  - Wasserstand und Durchfluss in einem Kanalschacht -> Text/CSV
  - Regen -> Text/CSV
- kontinuierliche Messungen der Berliner Wasserbetriebe
  - Fördermengen der Pumpwerke
  - Wasserstände in den Pumpwerken
  - Regen an benachbarten Regenschreibern

Aufgaben und angewandte Methoden nach Themen

- Modellierung/Kanalnetz/Kalibrierung:
  - InfoWorks: Erzeugen von Regen-Inputdateien

  - InfoWorks: Erzeugen von Real Time Control (RTC)-Dateien

Aufgetretene Fragen:

- Wo werden Präsentationen abgelegt?
- Wo werden die Rohdaten abgelegt? Z.B. Persönlicher Ordner “Diplomarbeit”
- Worauf beziehen sich die InfoWorks-Zeitstempel? -> Themenbereich Metadaten

Best of aus meiner persönlichen Logdatei:

2.3 Projekt: MIACSO

2.4 Projekt: KURAS

2.5 Projekt: Flusshygiene

Chapter 3

Projektbausteine

Wir können "Projektbausteine" definieren, die in verschiedenen Projekten vorkommen. Diese werden im Folgenden einzeln betrachtet.

3.1 Monitoring

In vielen Projekten führen wir eigene Messungen durch. Dazu verwenden wir eigene Messgeräte.

In Monitoringprojekten spielen Metadaten eine besonders wichtige Rolle. Wir betreiben viele Messgeräte mit Akkus. Der Zeitpunkt des Austauschs des Akkus ist eine wichtige Metainformation. Wenn Akkus zu spät gewechselt werden können wichtige Daten verloren gehen bzw. verpasst werden! Auch die Reinigung von Messgeräten ist wichtig, z.B. von Sonden mit optischen Messfenstern. Diese muss einerseits regelmäßig erfolgen. Andererseits ist es wichtig zu wissen, wann Reinigungen erfolgten, damit Messwerte entsprechend interpretiert werden können.

Im Projekt MIA-CSO haben wir erst angefangen, diese Informationen in einer Exceldatei zu erfassen. Angesichts der vielen Messgeräte, Pumpen, Kompressoren und anderer Geräte wurde das aber bald unübersichtlich. Später habe ich eine MS Access Anwendung dafür entwickelt, die seither in einigen Projekten zum Einsatz kam.

Problemstellungen:

- verschiedene Geräte verschiedener Hersteller -> verschiedene, meist nicht standardisierte Dateiformate
- Zeitstempel der Messung
  - Stellt das Gerät auf Sommerzeit um? -> Problembereich Zeitstempel
  - Geht die interne Uhr immer richtig? Wie gewährleiste ich das?

Beispielprojekte: MIACSO, KURAS, Flusshygiene

3.2 Kanalnetzsimulation

3.2.1 Datenaustausch mit BWB

Bei der Kanalnetzsimulation arbeiten wir eng mit den Berliner Wasserbetrieben zusammen. Diese haben und pflegen die Modellnetze.

Eine Schwierigkeit besteht im Austausch von Netzen und in der Versionsverwaltung. In welcher Version wurde ein Netz geschickt?

Gibt es dafür ein halbwegs vernünftiges Vorgehen? Wie ich es sehe, besteht eine große Gefahr, dass beim Hin- und Herschicken von Modellnetzen zwecks Abstimmung versehentlich wieder auf eine ältere Version zurückgegriffen wird und neueste Änderungen auf einer alten Version aufsetzen anstatt auf der aktuellen.

Neben Modellnetzen werden auch andere Bausteien, z.B. Regenreihen oder RTC-Bausteine ausgetauscht.

Fragen:

- Wie wird der Datenaustausch dokumentiert?
- Wie werden die ausgetauschten Objekte versioniert? Das müsste einheitlich auf beiden Seiten des Austauschs geschehen.

3.2.2 Modellkalibrierung

Trockenwetterkalibrierung

Die Kalibrierung der Abflüsse bei Trockenwetter wird anhand einer mittleren Trockenwetterganglinie gemacht. Dafür müssen Trockenwettertage aus den Messdaten ausgewählt werden. Wie wird das gemacht? Ist nachvollziehbar, nach welchen Kriterien die Auswahl erfolgte? Wie wird der Mittelwert über die Ganglinien berechnet? Was geschieht mit Ausreißern, was mit fehlenden Werten? Ist die Auswahl und Berechnung reproduzierbar und lässt sie sich leicht auf neue Daten anwenden? Das wäre eine typische Problemstellung, die sich mit Programmierung lösen lässt. Ggf. gibt es aber auch funktionierende Excellösungen dafür. Wie macht ihr das?

Bei der Modellkalibrierung wird versucht, die Modellparameter, die die Abflussbildung auf der Oberfläche beschreiben, so einzustellen, dass die simulierten Abflüsse und Wasserstände den gemessenen Abflüssen und Wasserständen entspricht. Dazu müssen die gemessenen Regenmengen so aufbereitet werden, dass sie in das Modell importiert werden können. Die erste Aufgabe besteht darin, die vollständige Regenmessreihe in Ereignisse zu zerlegen. Das ist wichtig, da InfoWorks ereignisbezoge z.B. die Evaporation berechnet. Nach der Zerlegung in Ereignisse müssen die Ereignisdaten noch in das richtige Format gebracht werden.

Siehe

- Erzeugen einer Ereignisliste
- InfoWorks: Erzeugen von Regen-Inputdateien

Beispielprojekte: MIACSO, KURAS, Flusshygiene

3.3 Datenaustausch

Es gibt internen Datenaustausch und Austausch mit Projektpartnern. Oft werden Dateien intern per E-Mail verschickt. Das sollten wir nicht tun, da wir unnötig Kopien von Dateien anlegen. Dateien sollten im besten Fall nur an einem Ort abgelegt sein. Dort sollten sie allerdings durch ein Backup-System gesichert sein. Natürlich muss der Zugriff auf diesen Ort gewährleistet sein. Einem Studenten mit eingeschränkten Rechten eine Datei per mail zuzuschicken, weil sie an dem Ort, an dem sie gespeichert ist, für ihn nicht zugänglich ist, ist meines Erachtens nicht die richtige Lösung. Es muss dann ein Ort geschaffen werden, an dem der gemeinsame Zugriff möglich ist. Laut Bodo ist der Ordner `Exchange` im Projektordner ein solcher Platz. Aus dem Grund der allgemeinen Zugänglichkeit wir er auch in manchen Projekten als allgemeiner Datenablageplatz “misbraucht”. Es kann aber niemandem ein Vorwurf gemacht werden, da ja nirgendwo dokumentiert ist, was in diesem Ort abgelegt werden soll.

Das können wir nun ändern:

Laut Bodo sind Dateien im Exchange-Ordner nur kurzfristig abzulegen, um z.B. Studenten Zugriff darauf zu geben. Die Dateien sollen nach dem Gebrauch wieder aus dem Ordner gelöscht werden. Demnach sollte der Ordner im Normallfall leer sein. Dies ist in einigen Projekten nicht der Fall.

Wir brauchen eine einheitliche Definition der Bedeutungen von Ordnern.

Wie die zu dokumentieren sind sollten wir in Metadaten beschreiben.

3.4 Datenimport

3.5 LCA

LCA steht für Life Cycle Assessment, bedeutet also soviel wie Lebenszyklusbewertung.

Chapter 4

Problembereiche

4.1 Metadaten

Beispiel:

Wenn wir Zeitreihen von den Beliner Wasserbetrieben bekommen, dann haben diese entweder eine Spalte mit Zeitstempeln oder zwei Spalten mit Zeitstempeln, wobei die erste den Anfang und die zweite das Ende des Zeitraums enthält, auf den sich die Messwerte in der entsprechenden Zeile beziehen.

Wenn nur eine Spalte Zeitstempel enthält, dann ist nicht klar, ob der Zeitstempel den Anfang oder das Ende (oder gar einen anderen Zeitpunkt innerhalb) des Zeitraums angibt.

Bei einer Zeitreihe mit fünfminütigen Messwerten, z.B.

```
Zeit                Wert
2018-06-05 16:20:00    0
2018-06-05 16:25:00    1
2018-06-05 16:30:00    2
2018-06-05 16:35:00    3
2018-06-05 16:40:00    4
```

ist nicht klar, ob sich der Wert 1 auf den Zeitraum `16:20 - 16:25` oder auf den Zeitraum `16:25 - 16:30` bezieht.

Dies ist ein Beispiel für eine sehr konkrete Metadatenanforderung. An dieser Stelle gehen wir meistens von Annahmen aus oder hat jemand von Euch schon mal eine Aussage darüber von einem BWB-Mitarbeiter erhalten?

Die Kenntnis über den Bezugspunkt des Zeitstempels ist zum Beispiel wichtig beim

- Erzeugen von Regen-Inputdateien für die Software InfoWorks

In der obigen Beispielzeitreihe ist auch nicht klar, auf welche Zeitzone sich die Zeitstempel beziehen. Die Zeitzone bzw. die Angabe des UTC-Offsets ist eine wichtige Metainformation. Am besten wäre es, wenn alle Zeitstempel diese Information enthielten, z.B.

```
2018-06-05 16:20:00+02
2018-06-05 16:25:00+02
```

Dann bräuchte man diese Information nicht extra zu führen.

Metadaten werden oft erst bei Bedarf erhoben, nämlich genau dann, wenn man sich z.B. die obigen Fragen nach Zeitstempelbezug und Zeitzone stellt. Dann wird nachgefragt und im besten Fall wird eine Antwort darauf gegeben, z.B. in Form einer E-Mail. Die Metadaten sind dann in der E-Mail enthalten.

Fragen:

- Wo und unter welchem Namen legen wir diese E-Mail ab?

Beispiel für Metadaten aus meiner persönlichen Logdatei:

> Regendaten gelesen; ca. 10 Impulse gegen 14:00; Dannecker geschrubbt, neuer Akku: Batterie defect, alte Batterie drin gelassen, Daten auslesen nicht möglich; H=55cm

Das müssen wir irgendwie besser hinkriegen, aber wie?

Die grundlegenden Fragen sind:

- Wer ist für die Ablage von Metadaten verantwortlich?
- Wo werden die Metadaten abgelegt? Direkt bei den Daten oder in einer übergeordneten Datei/Datenbank?
- In welchem Format werden die Metadaten abgelegt? Einfaches Textformat vs. XML-Format, das sich wohl einfach nur mit Hilfe eines Programms editieren lässt.

4.2 Zeitstempel

Frage:

Wie stellen wir die Messgeräte ein? Sollen sie die Uhr automatisch umstellen oder nicht?

Diskussion:

Aus Sicht der Datenverarbeitung bereitet uns die Umstellung von Winterzeit (= Normalzeit) auf Sommerzeit und von Sommerzeit auf Winterzeit immer wieder Schwierigkeiten. Warum?

Umstellung von Winterzeit auf Sommerzeit

Die Uhr wird um eine Stunde von 02:00 CET (*Central European Time*) auf 03:00 CEST (*Central European Summer Time*) **vorgestellt**. Merke: Die Stühle eines Cafés werden **vor** den Laden gestellt.

In einer Zeitreihe ergibt sich daraus am Tag der Zeitumstellung (z.B. am 25.03.2018) eine Lücke in den Zeitstempeln:

```
2018-03-25 01:45:00 CET
2018-03-25 01:50:00 CET
2018-03-25 01:55:00 CET
2018-03-25 03:00:00 CEST
2018-03-25 03:05:00 CEST
2018-03-25 03:10:00 CEST
```

Wenn die als Text vorliegenden Zeitstempel in einen numerischen Zeitstempel umgewandelt werden, gibt es, auch wenn die Angabe `CET` oder `CEST` fehlt, im Grunde kein Problem, da sie eindeutig sind. Voraussetzung ist, dass bei der Umwandlung angegeben wird, dass diese Zeitstempel in der Zeitzone “Europe/Berlin” (in der es Sommer- und Winterzeit gibt) aufgenommen wurden.

In der Programmiersprache R ist das auf unseren Rechnern das Standardverhalten, da die Zeitzone vom Betriebssystem abgefragt wird:

```
as.POSIXct("2018-03-25 01:55:00")
## [1] "2018-03-25 01:55:00 CET"
as.POSIXct("2018-03-25 03:00:00")
## [1] "2018-03-25 03:00:00 CEST"
```

Interessant an dieser Stelle ist das Verhalten, wenn ein Zeitstempel, der in der Zeitzone “Europe/Berlin” nicht existiert, angegeben wird:

```r
as.POSIXct("2018-03-25 02:30:00")
## [1] "2018-03-25 01:30:00 CET"
```

Leider tritt an dieser Stelle kein Fehler auf!

Umstellung von Sommerzeit auf Winterzeit

Die Uhr wird um 03:00 CEST wieder um eine Stunde auf die Normalzeit 02:00 CET **zurückgestellt**. Merke: Die Stühle werden wieder in den Laden **zurück** gestellt.

In einer Zeitreihe ergeben sich daraus am Tag der Zeitumstellung (z.B. am 28.10.2018) Duplikate in den Zeitstempeln:

```
2018-10-28 02:15:00 CEST
2018-10-28 02:30:00 CEST
2018-10-28 02:45:00 CEST
2018-10-28 02:00:00 CET
2018-10-28 02:15:00 CET
2018-10-28 02:30:00 CET
```

Ohne die Information `CEST` bzw. `CEST` sind die Zeitstempel (für sich genommen) nicht eindeutig. In Dateien, die wir von den BWB bekommen (z.B. Regendaten) würden die Zeitstempel zum Beispiel in dieser Form vorkommen:

```
28.10.2018 02:15:00
28.10.2018 02:30:00
28.10.2018 02:45:00
28.10.2018 02:00:00
28.10.2018 02:15:00
28.10.2018 02:30:00
```

Erst aus der Reihenfolge (erstes Auftreten ist CEST, zweites Auftreten ist CET) lässt sich die Zuordnung rekonstruieren. Dies müssen wir beachten, wenn wir die Daten bearbeiten und ich denke, dass wir dies in den seltensten Fällen tun!

Wir sollten eine einheitliche, eindeutige Vorgehensweise entwickeln.

TODO: Ggf. weitere Beschreibungen in R-Workshop, den ich mal gegeben habe…

Vorschlag:

- Nach Möglichkeit die Messgeräte so einstellen, dass der ausgegebene Zeitstempel den UTC-Offset enthält. Am besten wäre ein Zeitstempel in einem Format, das die Differenz gegenüber *Greenwich Mean Time (GMT)* bzw. *Coordinated Universal Time (UTC)* enthält, wie wir es ggf. aus unseren E-Mail-Programmen oder Terminkalendern kennen: `2018-06-01 23:18 GMT+02:00`.
- Manchmal gibt es auch die Möglichkeit, dass zusätzlich zum lokalen Zeitstempel der UTC-Offset (+01 im Winter und +02 im Sommer) als extra Spalte ausgegeben wird. Von dieser Möglichkeit sollte dann Gebrauch gemacht werden.
- Ansonsten empfehlen wir, die Geräte so einzustellen, dass sie die Zeitumstellung nicht mitmachen (also immer die Normalzeit, also Winterzeit, aufzeichnen). Dann sind die Zeitstempel eindeutig. Aufzupassen ist dann allerdings bei der Verarbeitung und Interpretation der Daten, dazu mehr unter Datenimport

Werkzeuge:

- KWB-eigenes R-Paket `kwb.datetime`. Dieses enthält unter anderem die Funktionen
  - `date_range_CEST()` zur Ermittlung der Tage, an denen die Zeitumstellung stattfindet.

TODO:

- Wichtigste Funktionen des Pakets `kwb.datetime`(z.B. zur Ermittlung der vollständigen Zeitstempel inklusive UTC-Offset aus fortlaufenden lokalen Zeitstempeln) dokumentieren und hier referenzieren.
- Sollten wir andere Pakete benutzen und wenn ja, welche? Was bietet z.B. `lubridate`? Es scheint mir allerdings, dass unsere Problematik im WWW etwas unterbeleuchtet ist.

Chapter 5

Allgemeine Methoden

5.1 Excel

Wie arbeite ich "sauber" mit Excel?

Best Practices zum Arbeiten mit Excel:

- z.B. Zellbezüge benennen, dadurch werden Formeln besser lesbar

Ihr werdet es kaum glauben, aber am Anfang meiner Zeit am KWB habe ich noch mit Excel gearbeitet. Ich habe auch komplexe Sachen gemacht und auch Excel-Makros programmiert. Ein Beweis findet sich in meiner persönlichen Logdatei:

> Fr, 14.09.07 08:15-19:30 benutzerdefinierte Excelfunktionen zur Modellgüte in Personl.xls, Modul1; TW-Kalibrierung Wochentag fertig

Ich würde heute nicht mehr empfehlen, Excel-Makros zu programmieren. Es ist umständlich, es gibt keine Bibliotheken und der Quellcode lässt sich nicht unabhängig von der Exceldatei verwalten, so dass keine ordentliche Versionsverwaltung möglich ist. Und wir wollen nicht mehr ohne Versionsverwaltung programmieren!

Wenn wir programmieren, dann sollten wir das einheitlich in R tun. R ist frei, es gibt eine großartige Community und wir haben mittlerweile eine große Expertise erlangt.

5.2 Programmierung

Wir verwenden die freie Programmiersprache R. Aller Programmcode soll unter Versionsverwaltung stehen. Das ist angesichts unserer über 1000 Skripte und komplexer Abhängigkeiten zwischen den Paketen und Skripten unabdingbar.

5.3 Versionsverwaltung

5.3.1 Subversion

Wir verwenden einen KWB-internen Subversion Versionsverwaltungsserver, mit dem wir über die Software Tortoise SVN, die direkt in den Dateiexplorer integriert ist, kommunizieren.

Im Rahmen von FAKIN legen wir fest, dass alle Programmcodes unter Versionsverwaltung gestellt werden sollen. Das gehört rein ins QMS!

5.3.2 Git

Wir haben auch einen öffentlich zugänglichen GitHub Account:

https://github.com/KWB-R.

Auf diesem legen wir nach und nach R-Pakete ab, deren Veröffentlichung keinen Beschränkungen unterliegt, z.B. weil der Code im Rahmen eines Auftragsprojektes erstellt wurde.

Chapter 6

Konkrete Aufgaben und Lösungen

6.1 Monitoring: Erfassen von Metadaten

Zur Protokollierung von Routine- und Wartungsarbeiten an den Geräten in Messkontainern habe ich eine komplexe MS Access Anwendung geschrieben. Mit dieser können Messstellen, Geräte, Aktionen (an Geräten) und Personen (die die Aktionen durchführen) verwaltet sowie Aktionen geplant und dokumentiert werden.

Probleme und Fragen

- Wo liegt sie?
- Wie wird sie bedient? Es gibt keine Beschreibung.
- Die Software wird nicht gepflegt
- Sollten wir sie weiter verwenden? Bestimmt gibt es mittlerweile Apps (womöglich kostenfrei), die auf Tablets laufen, benutzerfreundlich und gut dokumentiert sind.

6.2 InfoWorks: Erzeugen von Regen-Inputdateien

Methode: Perl-Skript `iw_event_writer.pl`

- Wo liegt dieses Skript? Erst lokal bei Hauke, mittlerweile unter Subversion, wurde glaube ich von Mathias in ein R-Skript umgewandelt…
- Wie kann es verwendet werden? Dazu benötigt man Perl, das ist standardmäßig auf keinem Rechner installiert. Auf einem Linux-Rechner wäre das der Fall… Sollten wir einen Linux-Rechner für solche Fälle vorhalten? Nun haben wir ja Rupelton…

6.3 InfoWorks: Erzeugen von Real Time Control (RTC)-Dateien

Ziel: Steuerung der Pumpwerke im Modell, wie sie in der Realität gefördert haben.

Benötigte Daten:

- Regendaten der Berliner Wasserbetriebe

Benötigte Metadaten:

- Worauf bezieht sich der Zeitstempel in einer kontinuierlichen Zeitreihe, auf den Anfang oder das Ende des Zeitintervalls mit der Länge eines Zeitschritts?

- Auf welchen UTC-Offset bezieht sich der Zeitstempel? Meiner Erfahrung nach ist bei den Daten der Berliner Wasserbetriebe immer der Lokalzeitstempel angegeben, in ihren Datenreihen gibt es also an den Tagen der Zeitumstellung Lücken bzw. Duplikate.

Manuelle Methode:

- RTC-Modul in Infoworks-Software erzeugen
- RTC-Modul als Text exportieren
- Pumpwerksdaten nach CSV konvertieren
- Pumpwerksdaten im Texteditor mit Suchen/Ersetzen auf das von InfoWorks geforderte Format bringen
- Formatierte Pumpwerksdaten in der RTC-Textdatei an den richtigen Stellen einfügen

Automatische Methode:

- Perl-Skript `rtc_generate.pl`. Dieses macht die obigen Schritte automatisch.
- Problem: lässt sich nur auf Rechner mit Perl laufen lassen.

6.4 Erzeugen einer Ereignisliste {ereignisliste}

Es können Funktionen aus dem Paket `kwb.event` verwendet werden.

TODO: Schreiben einer Vignette, die einem einen Leitfaden gibt. Es könnte ein Ausschnitt aus einem existierenden Skript verwendet werden.

Wir brauchen unbedingt Testdaten, die wir verwenden können, um bestimmte Dinge zu veranschaulichen oder als Eingangsdaten für den Test von Funktionen.

Aber wo legen wir die Daten hin und dürfen wir das eigentlich? Oder sollten wir künstliche (generierte) Daten verwenden?