College of Engineering, Design and Physical Sciences

Department of Electronic & Computer Engineering

MSc Distributed Computing Systems Engineering

Brunel University London

Conception and Implementation of a Single Window Harmonization System for acquisition and provision of Waste-Water Treatment-Plant data

StudentId: 1644612

Supervisor: Dr. Alireza Mousavi

March/2018

A Dissertation submitted in partial fulfillment of the requirements for the degree of Master of Science

College of Engineering, Design and Physical Sciences

Department of Electronic & Computer Engineering

MSc Distributed Computing Systems Engineering

Brunel University London

Conception and Implementation of a Single Window Harmonization System for acquisition and provision of Waste-Water Treatment-Plant data

StudentId: **1644612**

Signature of student:_____

Abstract

This dissertation presents the research on the topic of data harmonization and the waste-water treatment process, leading to the design and implementation of a "Single Window Harmonization System". The project is supposed to act as a proof of concept for a system, capable of gathering data from various waste-water treatment-plants, as well as providing the possibility to enter historical data – harmonizing the data and provide it in a unified form to 3rd party systems and users.

The first two chapters of this dissertation contain the introduction for the topic, as well as a description of the desired outcome. Furthermore describe the software development process with a focus on the chosen methodologies, tools and frameworks and the reasoning behind the choice. The second part focuses on the literature review – which aims at the creation of a knowledge base on which the harmonization system will be designed. The main subjects of the literature review are *waste-water treatment-plants*, *data harmonization* and taking a look on projects, which attempted to build a harmonization system. The third part deals with the design and implementation phase of the project. It describes the architecture and the workflow of the system, as well as the relations of the single components and the interfaces of the system with the outside world. The last part contains the results of the dissertation. This includes a description of the created system with focus on the requirements from the first chapter. Additionally the chapter contains a description on which work still needs to be considered, to make the system usable in the real world.

Acknowledgements

This Master's Thesis would not have been possible without the help of several people, which I want to thank in form of the following acknowledgements.

Thanks to Dr. Alireza Mousavi for the idea leading to this dissertation, as well as the support during the creation phase. Furthermore I want to thank the two PhD students Vasileia Vasilaki and Evina Katsou for providing me example data from existing water-plants, as well as interesting sources on the topic.

I also want to thank my girlfriend Julia, for taking care of our daughter, which decided to get her first two teeth right in the time I took holidays to work on this dissertation. Additionally I want to thank her mother Sabine and my mother Malgorzata, for taking care of Julia, who got sick while taking care of our daughter.

List of Figures

List of Tables

List of Abbreviations

WWTP	**W**aste-**w**ater **t**reatment-**p**lant
UK	**U**nited **K**ingdom
UNECE	**U**nited **N**ations **E**conomic **C**ommission for **E**urope
UNNEXT	**U**nited **N**ations **N**etwork of **Ex**perts for Paperless **T**rade in Asia and the Pacific
ESCAP	**E**conomic and **S**ocial **C**ommission for **A**sia and the **P**acific
FCTC	**F**ramework **C**onvention on **T**obacco **C**ontrol
WHO	**W**orld **H**ealth **O**rganization
ID	**I**dentifier
API	**A**pplication **P**rogramming **I**nterface
EEA	**E**uropean **E**nvironment **A**gency
DB	**D**ata **B**ase
VS	**V**isual **S**tudio
AWS	**A**mazon **W**eb **S**ervices
IDE	**I**ntegrated **D**evelopment **E**nvironment
SQL	**S**tructured **Q**uery **L**anguage
JSON	Java**S**cript **O**bject **N**otation
XML	**Ex**tensible **M**arkup **L**anguage
DEFRA	**D**epartment for **E**nvironment, **F**ood and **R**ural **A**ffairs
SWH	**S**ingle **W**indow **H**armonization
CSV	**C**omma **S**eparated **V**alues
XLS	**Ex**cel **S**preadsheet
RAM	**R**andom **A**ccess **M**emory
Ms	**M**ili**s**econds
URI	**U**niform **R**esource **I**dentifier
CPU	**C**entral **P**rocessing **U**nit
UML	**U**nified **M**odeling **L**anguage

List of Terms

WWTP / Waste-water treatment-plant / water-plant	A **Waste-water treatment-plant** is a technical system build for the purpose of water-cleaning.
3rd Party System	A system, the system under design has no knowledge about, but is still able to interact with.
DD / MM / YY	In the context of a date, those letters represent the **D**ay, **M**onth and **Y**ear, as well as the number of positions they require.

Content

1. Introduction

Access to clean water is the most basic and fundamental type of the human infrastructure. The quality of life highly depends on the accessibility to clean water. As human, we require water not only for drinking, but also for cooking, and washing. Additionally, various professions and commercial establishments, like farms or restaurants, could not exist without certain quality and quantity of water. The quantity of clean water in most cases, depends on collecting water and sewage from rivers and lakes, cleaning it in dedicated **waste-water treatment-plants** and thus bringing it to a specific quality standard, and then distributing it.

A software groundwork for acquisition, analysis and modelling of historical and real-time data of water-plants could become the first step to provide an infrastructure capable of monitoring the water on a national level – its amount and quality, as well as making forecasts and statistical reports easier. Therefore, the topic of this thesis is to design and create a prototype of a system to acquire, harmonize and provide waste-water treatment plant-related historical and real-time data from different sources.

1.1 Context of the Project

To understand how important waste-water treatment is, one has to under-
stand the dimensions of its effect. On average, a UK water-customer pays 1
pound a day to be able to enjoy high quality water. This money is highly
needed and is used for the treatment of around 16 billion litres of wastewater,
gathered in around 347 thousand kilometres of culvert and cleaned in about
9000 wastewater plants – every day, as well as developing the water infra-
structure in the country. [2, pp. 2,3]

Every country has its own way of dealing with the regulations of the water
treatment process: Taking into consideration the 4 Steps:

- Collecting wastewater

- Primary treatment[1]

- Secondary treatment

- Tertiary treatment

The statistics reported by the European Environment Agency [3] show, that
the trend in Europe over the past decades was to connect more and more
population to waste water treatment plants. In northern countries 80% have
already been reached in 1995, with over 70% of the water receiving tertiary
treatment. In central Europe (this includes the UK) on average over 95% of
the population enjoy treated water.

---

[1] The 3 treatment stems will be explained later in detail

The trend also shows increasing amount of tertiary treating among all countries:

Figure 1-1: Waste-water treatment in Europe [3, p. Fig. 3]

In the UK, the entire population is connected to at least one WWTP. Also, all of the collected water receives at least a secondary treatment (Data from 2015). What UK still lacks, is tertiary treatment which is close to 100%, like in other countries including Germany, Netherlands, Denmark and Austria. The

trend shows improvement over the past years, but it seems rather slow in the progress. The tertiary treatment usually significantly reduces nitrogen and phosphorus pollution and might not always be required, but is still recommended by the Urban Waste Water Treatment Directive. This evolution does not only show how important waste-water treatment is, but also that the improvement of its process is an ongoing topic.

On a big scale, the UK can compare its achievements to other countries to find out what causes their good and bad results. The comparison on a small scale includes comparing the single water-plants within the UK to see which treatments steps are lacking, where the water quality is better, the treatment more effective and efficient and why.

The UK-wide water supply regulations are set by the government and regulate the water treatment process of every water provider whose area is wholly or partially in the United Kingdom. The list of indicator parameters is long and contains minimum, maximum values and ranges within which values are allowed to lie. Only if all regulations apply, the water may be called drinking water.

With all the regulations and monitoring organizations the quality of UKs water might seem assured – yet the process of doing so is very troublesome and laborious. [3] [4]

Twelve big companies, responsible for water and sewerage, cover most of UKs water supply. Additionally, there are some water-only companies providing water for some of the remaining regions.
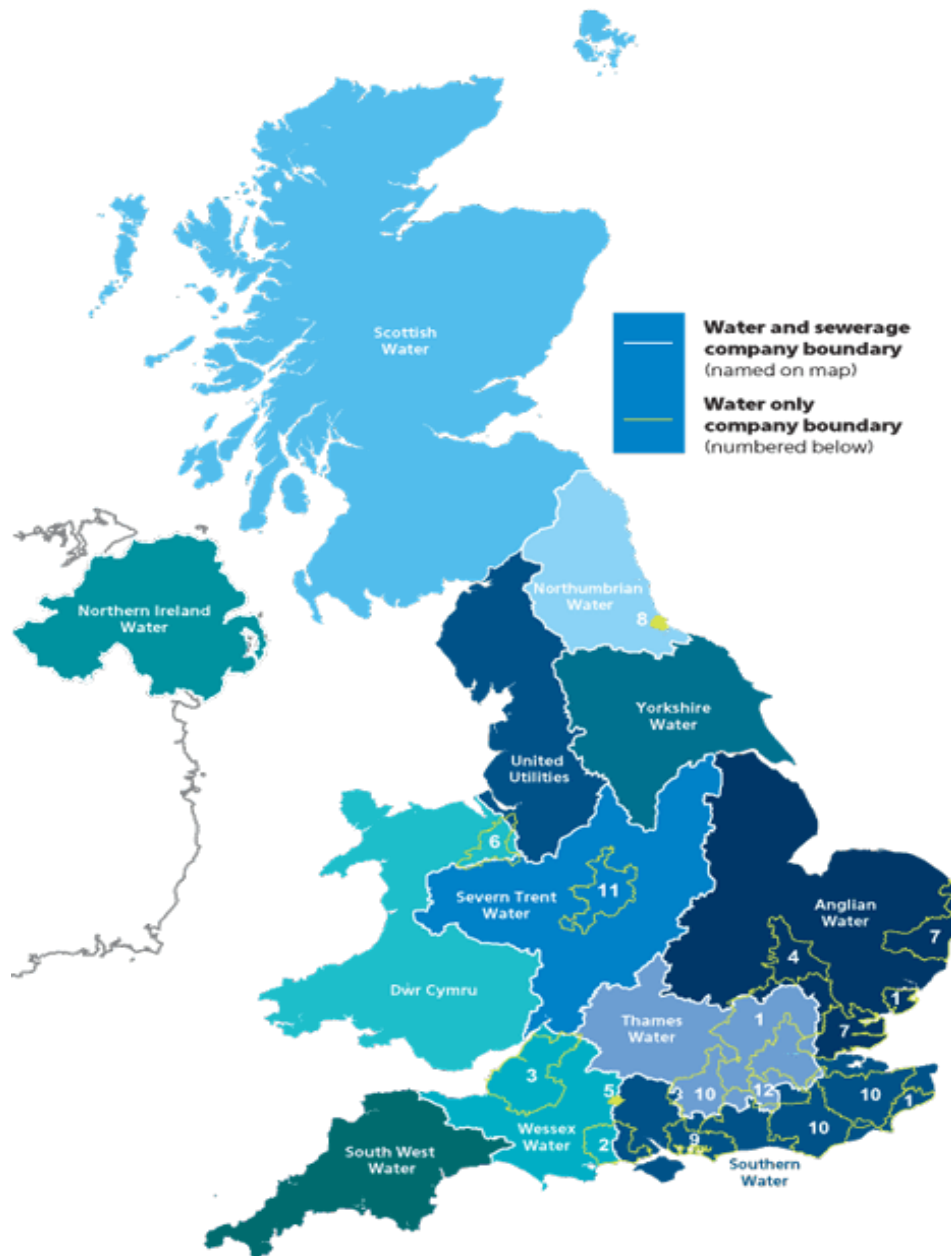
Figure 1-2: Water suppliers in the United Kingdom [5]

Water transfer and interconnection

A briefing sheet by Ben McAliden [6] provides insight on the topic of water transfer. Depending on the location, the population and the climate, some

regions of the UK have less available clean water than others. Those regions usually lie in the south and east. This is why a system for water moving was required and build already in the 17th century (*New River* to transfer water from Hertfordshire to London). This transfer and interconnection system was since then optimized and expanded to provide water to regions in need, despite other water companies being "responsible" for this region[2].

Water can be transferred treated or untreated using canals, pipes, aqueducts or rivers. Treated water is typically transported over buried pipes. Transferring the water is costly, especially when lowland reservoirs have to supply upland reservoirs and the water needs to be pumped instead of having the gravity doing most of the work. This is why water should not be transferred if not explicitly necessary. Even though water transfer – or "water trade" when referring to water transfer between companies, is costly, it is still in most cases cheaper in money and energy than water desalination, and thus promoted by the government and regulators in the UK.

1.2 Problem Description

The water quality is regulated UK-wide, yet the way the different companies ensure their quality and monitor their water treatment process is not entirely unified. This makes comparison of data between companies and water-plants, as well as getting a global picture, difficult. Reacting to lack of quality water in specific regions, or forecasting such a scenario, while still monitoring which of the remaining regions has enough "spare" quality water to help out the company in need would be a lot easier with a common information base. It would simplify the monitoring of local area changes caused by changes in

---

[2] „Intra-Company transfer"

the water and wastewater treatment regulations. To assure better forecasts or more meaningful reports, other information bases, like weather information might be taken into account – but using those external systems is not a topic in this part of the (data-gathering) system.

The advantages of a big dataset from various sources are obvious – especially in a case where the geographical location of sources also matter. Co-operating, comparing, planning, monitoring and analysing is a lot easier when all the data can be retrieved from one place in a unified format.

Several questions need to be investigated upon before the approach of gathering the water information from various water-plants can be designed and implemented. The following list shows the questions that occur and need to be researched upon. While the core questions for this dissertation are marked **fat**, there are some other questions, which will occur at some point during the development of the final system, beyond the scope of this thesis. Those questions will be treated as marginal questions during the research phase and might get answered if the research on the core questions leads to it. Most of them will be mentioned in the "Future Work" chapter.

- Which data is available for the existing water-plants
  - **Which water-quality indicators**
  - How can the data be accessed
  - How can the data be legally used
  - What needs to be done to access the data
- Which data is mandatory to create a useful platform
  - Who are the stakeholders
  - Do all / most water-plants provide this data
  - **Which format is the data provided in**
  - **Which format is optimal for this system**

- What is the best way to handle this data in terms of
  - **Storage**
  - Security
  - **Processing**
- What is the best way of harmonizing the data in terms of
  - **Flexibility** (adding new sources and components / reacting to changes in source schema)
  - **Performance**
  - **Usability** (Target-Schema)
- What is the best way to provide a system-entry point for various data types
  - **Historical data**
  - **Actual data** (real-time / near real-time)
  - Non-water-data which could be useful anyway (i.e. for forecasts or to put more context on the stored data)

1.3 Aims and Objectives

The following chapter describes the aims and objectives of the dissertation. It is separated into two parts. The first part describes the general objectives of the dissertation and the second describes the aims and objectives towards the required system, in form of technical requirements. The requirement form takes into account that the most fitting strategies and specific techniques have yet to be determined.

1.3.1 General objectives

The purpose of this dissertation is to investigate and design a knowledge and data engineering infrastructure for big amounts of water-/ and wastewater treatment-process specific data. This includes answering the questions asked in the chapter 1.2 – "Problem Description", as well as designing an effective solution capable of harmonizing data monitored within various water-plants based on the answers. Finally, a prototype of the designed system

should be implemented and act as a proof of concept for the design. Simulated and historical data should be used to test the systems harmonization capabilities, as well as its flexibility in terms of expansion by further components. The below picture shows the basic idea of the system integrated with the real world. The scope of this project lies within the Large Network Performance Collider, which represents the harmonization and storage layer. Gathered data may be of any category and be of use for any number of stakeholders.
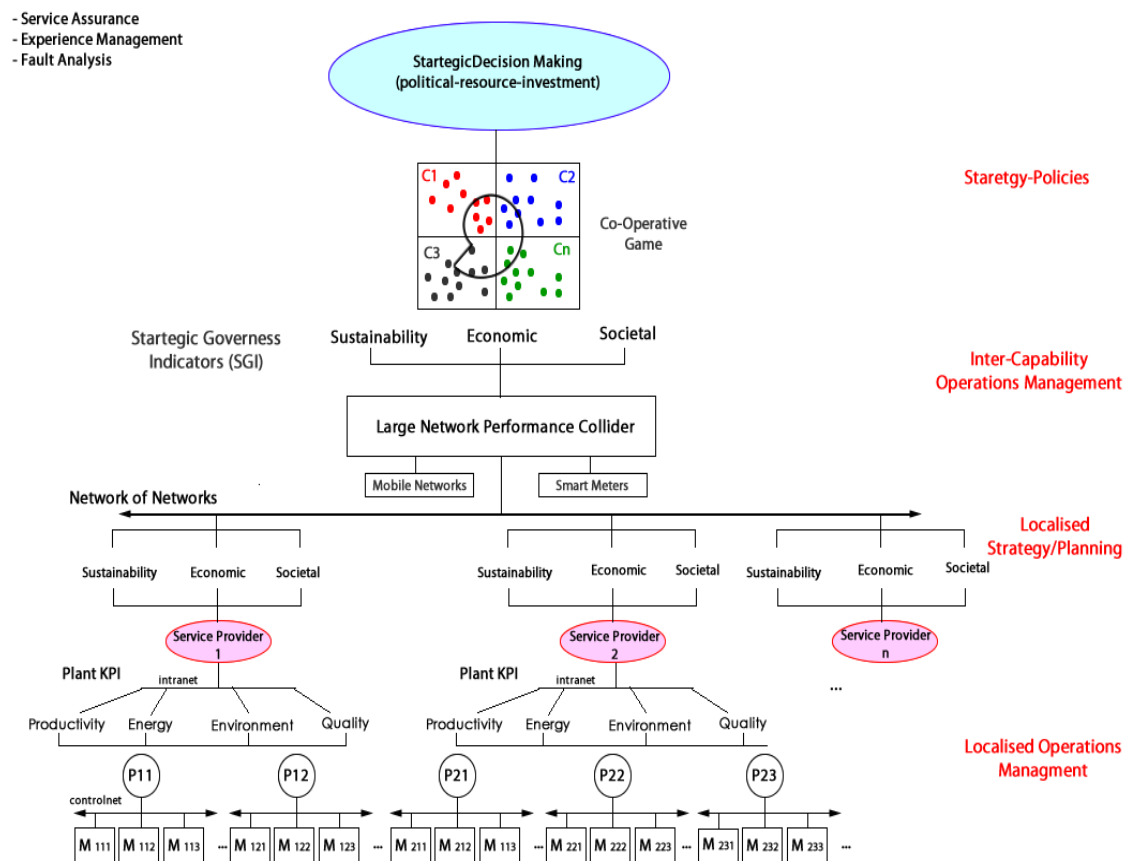
Figure 1-3: Big picture – System in context [7, p. 3]

1.3.2 Requirements

The systems functionality will be defined in the following requirements. To ensure readability and a better overview the requirements will be numbered as followed:

Top-Level Requirement:	10.000
Required Use Cases:	11.000
Requirements for the Use Cases:	12.XXX
Requirements for technical functions:	13.XXX
Requirements for the quality indicators:	14.XXX
Requirements for the limitations of the solution:	15.XXX

1.3.2.1 Functional Requirements

<u>Top-Level Requirement</u>

Requirement 10000

The developed system must **gather and accept** various, defined forms of waste-water treatment-plant data, **harmonize** the data into a common format, **standardize** the harmonized data and **store** it. The system should **provide** an endpoint for the outside world to request the stored data.

<u>Requirements for the Use Cases</u>

Requirement 11000

The system should fulfil the following Use Cases:

- The user must be able to send data to the system in order to harmonize it. The administrator must be able to define endpoints and metadata of water-plants in order for the system to pull the data from the defined source and harmonize it. (Use Case: "**Harmonize Data**")

- The system must provide endpoints for the users to access the harmonized data and process the requests. (Use Case: "**Provide Data**")

Requirements for specific Use Cases:

- **Requirements for the Use Case "Harmonize Data"**

Requirement 12110

The system must provide an endpoint which accepts multiple formats commonly used for storing of data monitored by waste-water treatment-plants.

Requirement 12120

The system must have a possibility to configure endpoints and endpoint specific metadata and pull the data from the configured endpoints.

Requirement 12130

Incoming data (if in an accepted format) must be transformed (harmonized) into a predefined format.

Requirement 12140

The harmonized data must be stored by the system.

Requirement 12150

The harmonization process must exclude blacklisted properties.

- **Requirements for the Use Case "Provide Data"**

Requirement 12210

The system must provide endpoints to request available water-plants, treatment-steps and quality-indicators in a generic manner.

Requirement 12220

The system must provide an endpoint to request harmonized data with optional data filters.

Requirements for technical functions

Requirement 13100

Harmonization processes, which for some reason do not end successfully must provide a meaningful error message to the harmonization requester.

Requirement 13200

If needed, a fitting simulation should be built to test the finished system.

1.3.2.2 Non-Functional requirements

Requirements for the Software Quality Indicators:

Requirement 14100

Data consistency must be granted.

Requirement 14200

The harmonization process must be implemented efficiently and effectively.

Requirement 14300

The system must be able to handle multiple requests in parallel.

Requirement 14400

The system must be designed to be expandable during runtime[3].

Requirements for the limitations of the solution:

Requirement 15100

The endpoints should use the REST technology and be defined accordingly

to the REST guidelines.

Requirement 15200

The service must be deployable as a cloud service.

Requirements 15300

The storage must be able to handle big amounts of data efficiently and effec-

tively.

---

[3] The software does not have to be recompiled to expand the system by new quality indicators, treatment steps or water plants

2. Methodology and Project Organisation

This chapter will go into detail about the development method, tools and frameworks, which will be used to design and implement this project. It shows the combination of the methods and the project plan. Furthermore, this chapter will introduce to the time-plan describing the tasks of this project presented as a gnatt chart.

2.1 Software Development Process

The system developed within this dissertation has a clear aim. The challenges and risks can be estimated, because the objective is clearly stated and there are comparable systems and projects already existing. Based on this knowledge, the system design will be done **top-down**. Since the system will consist of components, which can be implemented separately, the approach already follows the spirit of a top-down design, and the system as a whole will be designed one component after another.

2.1.1 Top-Down Design

Barnette ND and McQuaid WD [8] give a technical description on this design. The top-down design starts by creating an overview over the entire system. The details are neglected at first. This approach requires a good understanding of the system (opposing to a bottom-up design, where the system is being implemented without detailed understanding of every component, and the gain on information comes in during the development). The problem as a whole, and thus the system, is broken down into smaller sub-problems. This is why the approach is also called "Divide and Conquer strategy". This breaking down continues, until the problem is small enough to be confidently

solved on its own. While in big projects dividing the problems often results in breaking the project into sub-systems, where different programmers, or programmer teams can work separately on their own part of the system. The advantage of this approach in a one-man-project like this dissertation is the fact that subsystems become easier to estimate time and risk-wise the smaller they get. They can be prioritised or stripped down of less important features if the time wouldn't be sufficient otherwise.
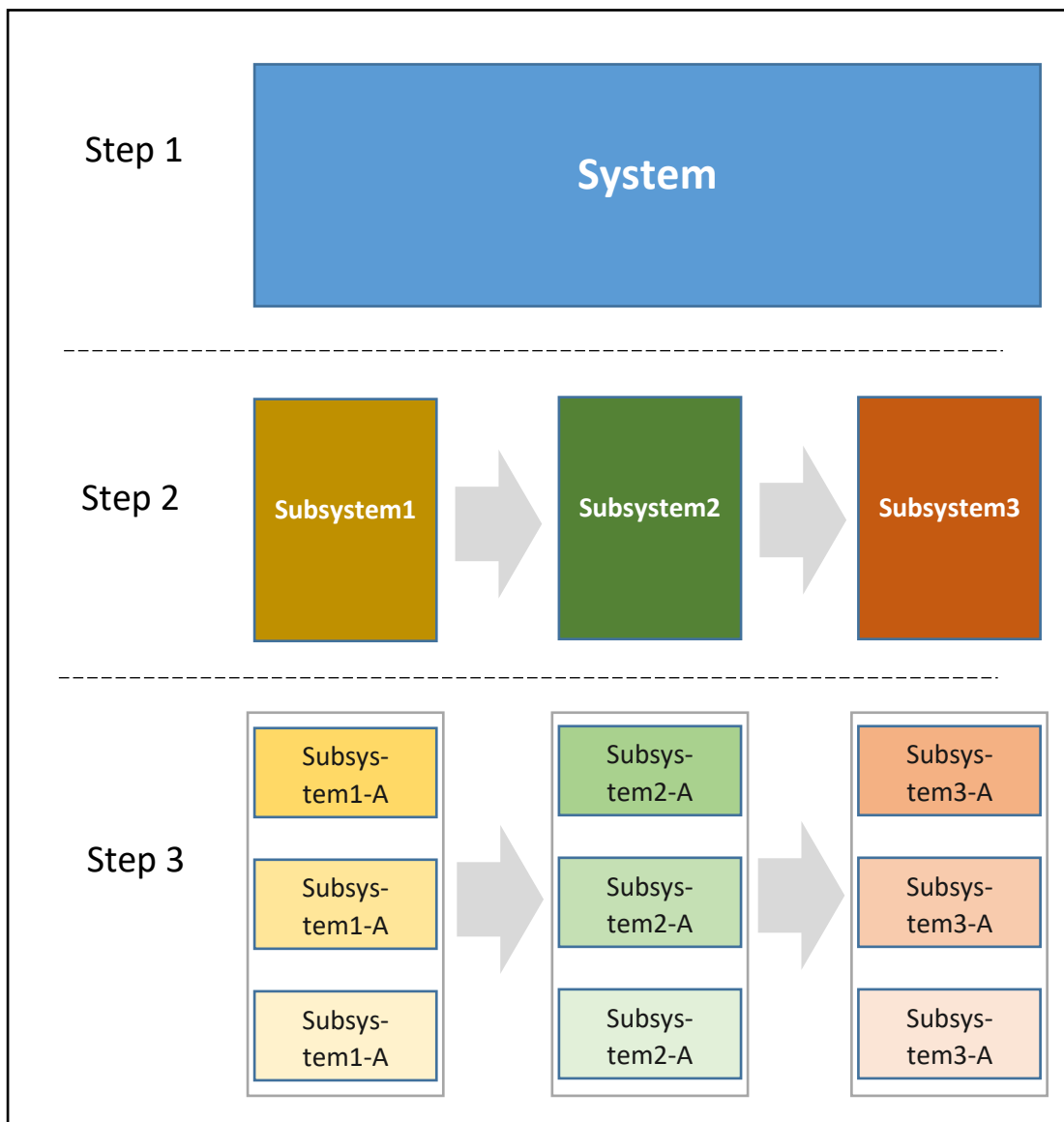
***Figure 2-4**: Top-Down design*

The advantage of dividing the system into subsystems is also the fact that each component can be tested individually if defined sufficiently. In most cases this will also assure that the subsystems become exchangeable – due to their modular character.

2.1.2 Spiral Model

Developing a system like this is special. The most stakeholders and contact persons knowing a lot of the subject will not have much software-technical knowledge. An iterative approach is of advantage in this case. Additionally it allows to detect risks and upcoming problems early on, which might lead to a change of the entire approach of the system.

A classical spiral model typically consists of four stages, in which the system is incrementally developed. In combination with the Top-Down design, the spiral model will help in developing each of the components during an iteration, planning the next iteration and thus the next component, and implementing the next component, as well as integrating it with the previous components.

Determination of objec-
tives, alternatives, con-
straints

Evaluate alternatives,
identify and resolve risks

Project exit
point axis

Project
entry point
axis

Prototypes

Sub-System1	Sub-System2	Sub-System3	Sub-System4

Plan next iteration

Develop and identify next
product level, Integrate
the developed system
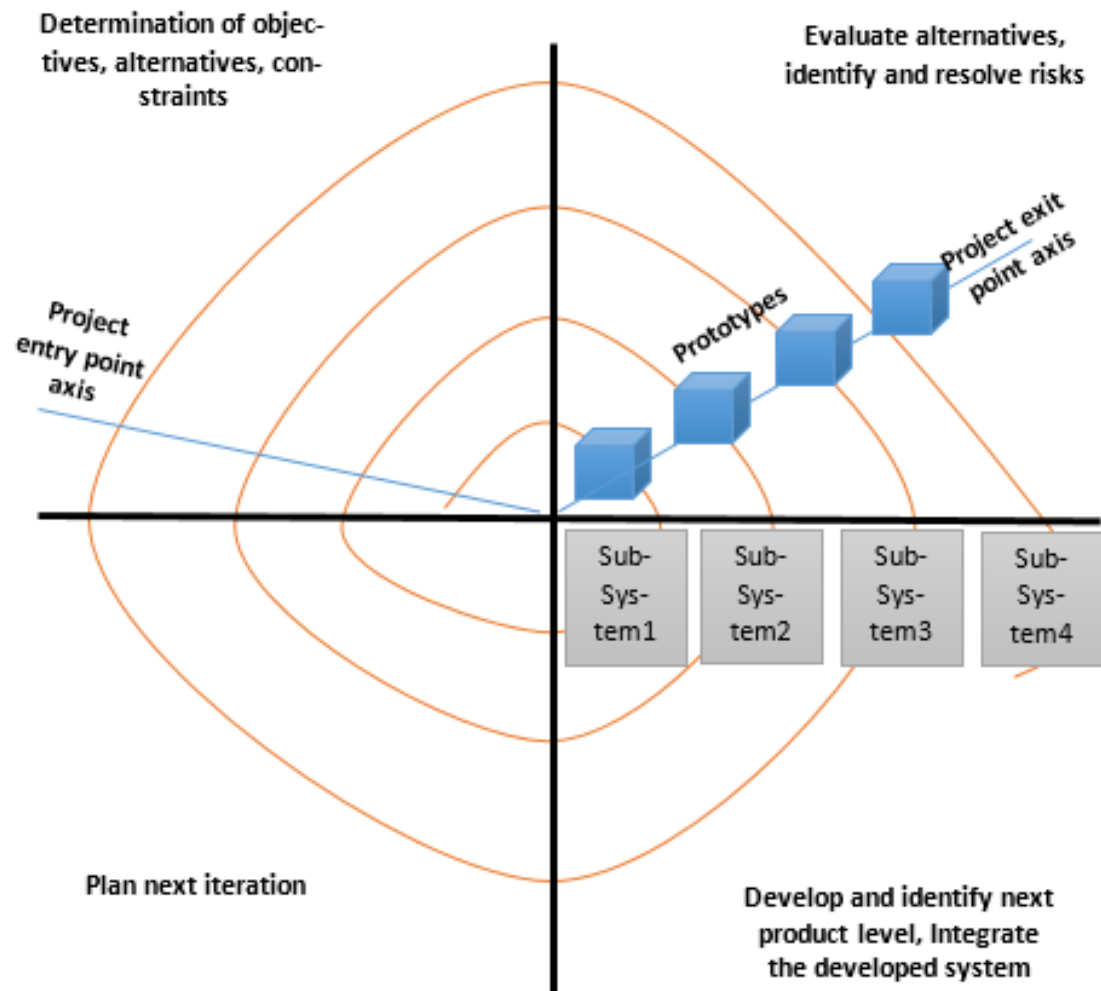
Figure 2-5: Spiral-model combined with top-down approach

Each of the development phases not only includes work on the current sub-system, but might also include changes and adjustments in the previously developed subsystems to assure their integration in the current stand of the system with all its developed components.

2.2 Tools and Architectures

This chapter deals with the tools, frameworks and architectures, which will be used to develop this system. The core components of the system will be the database, the harmonization service and the data provider. It has to be made

sure, that not only the components can fulfil the aims and objectives described in the previous chapters, but also that the various tools, frameworks and technologies are compatible with each other.

2.2.1 Visual Studio / C# .Net

The software will be developed in the programming language C#, developed by Microsoft. C# is an object oriented language based on the .Net framework. Its syntax is influenced by languages like Java and C++. C# is a language constantly developed by Microsoft and up to date on the state of art in terms of architectures, testability and support-tools and allows clean programming of back-ends, as well as front-ends. C# has lots of good documentation and can be used for building web-services for the most common cloud platforms like azure or amazon cloud. Another advantage of C# is, that even though the language itself is type-safe[4], there is the possibility to work with the *dynamic* data-type, which usually counts as bad practice, but is very fitting for a software which desires a high tolerance for data formats. Those types allow for assigning of different types to one and the same variable during runtime. The risk of this are obvious: the developer has to keep track of what could be stored in the variable at all time to prevent runtime failures.

The most common IDE (Integrated Development Environment) for C# development is Visual Studio. Visual studio is a mighty tool which provides a lot of possibilities for developers. Not only code can be developed within VS, but also databases and websites. Using additional frameworks (like Xamarin), Visual Studio even allows for development of Android and iOS applications.

---

[4] The type of a variable has to be specified before assigning a value to it and cannot be changed during runtime.

Summarized, C# in combination with Visual Studio (2017) provide for all the features which are required to develop this system with an adequate architecture.

2.2.2 Postman

Postman is a tool, which simplifies API development, by allowing the developer to create web calls, defining the call method, header, body and payload, as well as showing the call result. It also allows for easy passing of binary files.

2.2.3 Amazon Cloud

Amazon Cloud [9, pp. 11, 12, 13, 15] (Amazon Web Services or short „AWS") provides a lot of services including databases and deployment tools. Services fully deployed in the (Amazon-) cloud can take advantage of its scalability and analysis infrastructure. Although the system under design has no matter of security in the current stage, future security requirements can be added without having to make big changes on the service itself. As for the storage, the AWS provides multiple options on how the data can be stored, including the object storage called "Amazon S3" useful for modern cloud native entity frameworks, a block storage called "Amazon Elastic Block Storage" which ensures data-loss protection through replications of the data, a File System called "Amazon Elastic File System", which allows for intuitive storing and accessing files and many more. The features of AWS are by far sufficient to fulfil the requirements of this project and also provide tools to further improve and optimize the system once the aims and objectives of this disserta-

tion are finished. This includes components like an API gateway which helps with the handling of security issues and load balancing.

2.2.4 Microsoft SQL Server / MSSQL Management Studio

The previously described Amazon Cloud allows for integration of the most common SQL databases. One of the options is the Microsoft SQL Server (short MSSQL). Like .Net and Visual Studio, MSSQL is also developed by Microsoft and thus offers good support and integration into the named framework and IDE. Aside of integration with all the used technologies, MSSQL also offers a fast relational database system which supports stored procedures as well as views. Both are of advantage when trying to build and integrate a flexible and testable system. Stored procedures are functions stored on the database, which can be integrated into the web-service and allow the execution of SQL queries, without the need to write actual SQL queries in the web-service itself. Views on the other hand, allow for the implementation of a lazy-loading mechanic, as well as for creation of types based directly on the database view structure. The lazy-loading mechanic allows to create so called *Queryable* objects, which act as a tunnel between the service and the database. The objects act like usual objects, but do not actually contain any other data, than an SQL query until the data is really needed. It then only retrieves the data which is actually needed or requested, instead of iterating through and retrieving the entire view data. This is especially useful for systems with high amounts of data and parallel processing.

2.2.5 JSON

One of the main components for this system is description of the data schema. This not only includes the description of the individual data, but also the format in which this data should be provided to the system, as well as the data format the system provides its data to the outside world.

JSON (**J**ava**S**cript **O**bject **N**otation) is an object/data format. It has become a very commonly used format for client-server communication and is replacing XML in this matter. It is supported by the modern programming languages and frameworks. JSON basically consists of object identifiers followed by the objects. Those objects might be further JSON types. The advantages of JSON when compared to other formats:

Readability:

JSON formatted objects are easy to read for a human, as its syntax focuses at removing the unnecessary parts and presents the data as an object similar to programming languages syntax.

Example of JSON a formatted object "Person":

```
{
    "firstName": "Jack",
    "lastName": "Black",
    "age": 29,
    "address":
    {
        "streetAddress": "Ringelweg 35",
        "city": "Esslingen",
        "state": "GER",
        "postalCode": "73730"
    },
    "phoneNumber":
    [
```

```
        {
          "type": "home",
          "number": "212 555-1212"
        },
        {
          "type": "fax",
          "number": "646 555-4567"
        }
    ]
 }
```

Speed:

JSON is a lightweight format, and thus faster than other formats like XML. It is written in JavaScript object notation and thus require less time to parse.

Size:

JSON does not add much overhead to the data when parsed, which is not only of advantage for the readability, but also size. Data size is especially critical when it comes to transferring it over the internet and in big amounts and thus JSON is a good choice for this purpose.

2.3 Strengths and Risks

As mentioned in chapter 2.1 "Software Development Process", one of the advantages of an iterative approach is the fact that progress can be frequently presented to make sure it is exactly what is required. Another strength of this approach, is the fact that system components can be prioritised. To provide a proof of concept one would usually need a simulation. Defining and building a simulator can have a variable complexity – from static values to a highly complex model of the real world system. Depending on the remaining time and priority, this and any other task can be adjusted correspondingly. Since only proven tools and frameworks are used to create this system, there

is no speculation in the question, if the tools and frameworks are sufficient to fulfil this task.

There are several risks in the development process of this system. One of the biggest risks is the time to finish the whole project. The project requires a lot of research work on the subject of waste-water treatment process, as well as on the topic of data harmonization. Furthermore the creation of a corresponding relational database system, a fitting data-schema and the implementation of the harmonization process as well as the data provider is a time consuming task. Also as mentioned, some additional work has to be done to prove the concept entirely, since it will not have the access to actual wastewater treatment plants.

Another big risk is the fact that there is not much divergent example data from wastewater treatment plants. This not only makes testing of the system harder, but also leads to the possibility, that a better solution to the problem could be found if there was more data samples.

Additionally to the problems in the development phase of the project, there are also general problems. At the time of the implementation of this system, there is no possibility in accessing the waste-water treatment plants data any near real-time.

Risk	Effect	Classification	Strategy
Time develop the system is low	The System will not be finished until the time runs out	Catastrophic	The time-plan has to be as precise as possible, also including buffer time and has to be strictly followed.
No way for trivial comparison of different data-formats	The harmonization requires a different strategy for every format-structure included into the system	Critical / Delays the project	Try to merge the different formats into one common format as early in the harmonization phase as possible.
Low amount of divergent sample data	The system cannot be sufficiently tested, some solution strategies cannot be developed without sufficient sample data.	Critical	Only solutions which lie within the possibilities, based on the boundaries of this project will be considered.
No access to real-time data	The system cannot be tested with actual water-treatment plants	Marginal / Delays the project	The aim of this project is only to create a proof of concept and not a working system – thus a water-plant simulator is enough to fulfil this task.
No access to water-plant data in general	The system, even if finished, would have no use in the current time.	Insignificant	It is not the aim of this project to create a working system, but only to provide a working proof of concept showing, that under different circumstances, this solution would work.

Table 2-1: *Risks, Effects, Classifications and Strategies*

2.4 Project Management

Based on the spiral development strategy described in a previous chapter, this project will be developed one functioning component after another. Previous to that, the problem to solve has to be described in detail and separated into defined smaller problems. The following sub-chapter describes the tasks which need to be fulfilled in order to successfully end the project.

2.4.1 Project Tasks

Category	Task	Description
Analysis	Analysis of the Problem	Analyse why the problem exists, what possibilities and advantages it would bring to solve this problem and why it might be a difficulty to solve it.
	Analysis of water-plants	Analyse how water-plants work, what the water represent, how the water-plants differ from each other and what their similarities are.
	Analysis of data harmonization methods	Analyse what data harmonization is, how it is defined, how it is generally used and common problems.
	Analysis of similar projects	Analyse projects having similar aims as this project. Which knowledge, techniques and methodologies of data harmonization can be used for this project?
Interim Report	Creation of the interim report	-
Planning	Defining the development model	The development process will be defined by one, or a combination of proven models.

	Deciding on Tools and Frameworks	The Tools and Frameworks include the database, the cloud, the programming language as well as the IDE used in the project, as well as the data-format of the schema
	Detemining Risks and Strengths	Determining risks and strengths of the project includes finding tasks which for a reason might lead to problems within the project, and find strategies to deal with them, as well as finding strengths in the planning decisions.
Design	Design of the System	Designing the overall system. Defining how the components should interact with each other and the outside world.
	Design of the data-schema	Deciding on a common schema for the outsides world communication with the system.
	Design of the database	This step includes the design of the tables and their relations.
	Design of the harmonization service	Designing the harmonization service. This includes breaking down the service in components and defining each components output as well as defining the strategies for each
	Design of the data provider	Designing a data provider in terms of access and data format of the output, as well as its access to the database.
	Design of the simulator	Designing a simulator which can be used as real-time data provider as well as an endpoint to pull water-plant data from.

Implementation / Realisation	Setting up the data base	Setting up the database includes the creation of the tables, the views and the stored procedures, as well as making it accessible to the system.
	Implementing the harmonization service	Implementation of the harmonization service is the most crucial task for the proof of concept. It includes the implementation of the harmonization service and integrating it with the database.
	Implementing the data provider	Implementation of the data provider includes the integration with the database, as well as making the provider accessible by the outside world
	Implementing the simulator	Implementation of a tool which acts like water-plant, in terms of data providing.
Conclusion	Research on the system behaviour	Investigating if the proof of concept was successful, defining further work which needs to be done to make the system work in the real-world.

Table 2-2: Project tasks in categories and their descriptions

2.4.2 Gantt Chart

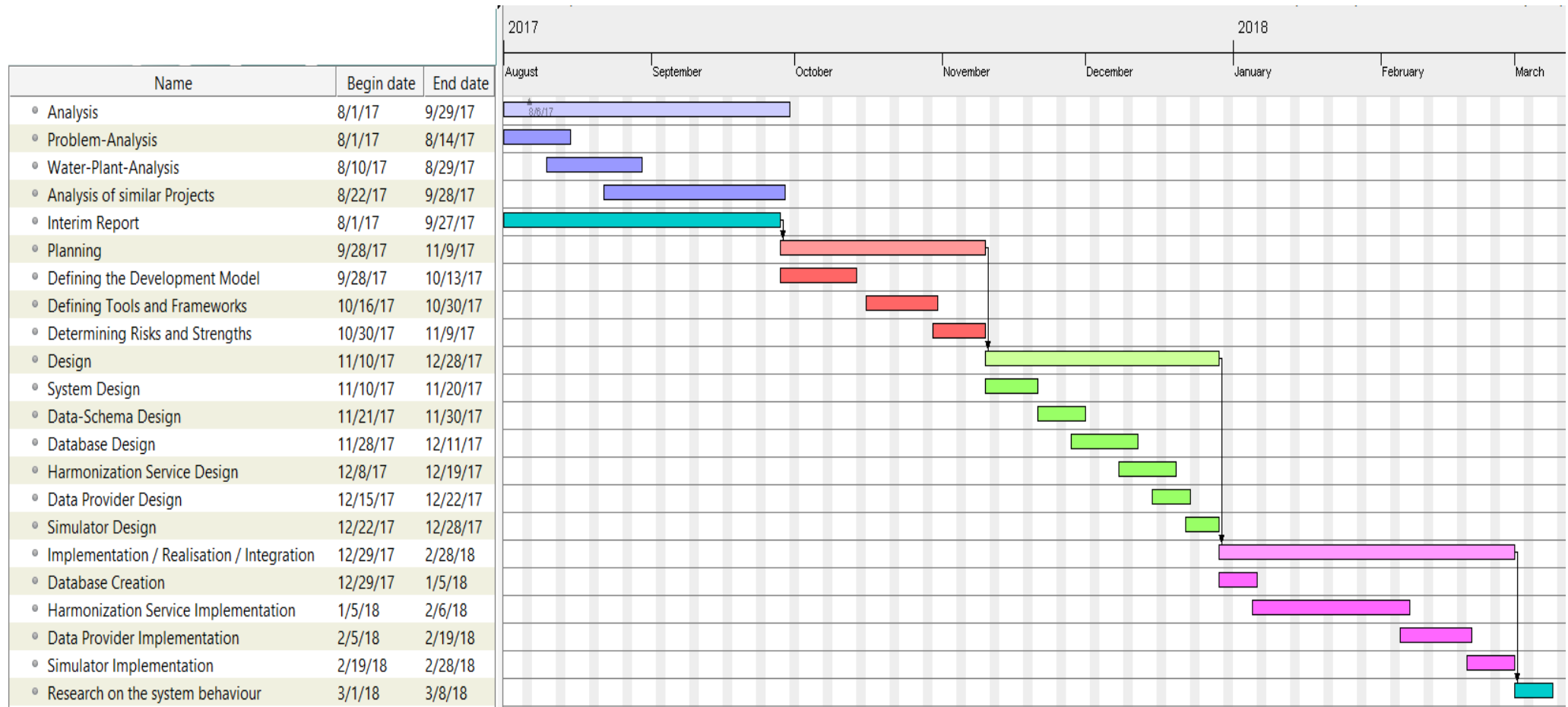| Name	Begin date	End date
Analysis	8/1/17	9/29/17
Problem-Analysis	8/1/17	8/14/17
Water-Plant-Analysis	8/10/17	8/29/17
Analysis of similar Projects	8/22/17	9/28/17
Interim Report	8/1/17	9/27/17
Planning	9/28/17	11/9/17
Defining the Development Model	9/28/17	10/13/17
Defining Tools and Frameworks	10/16/17	10/30/17
Determining Risks and Strengths	10/30/17	11/9/17
Design	11/10/17	12/28/17
System Design	11/10/17	11/20/17
Data-Schema Design	11/21/17	11/30/17
Database Design	11/28/17	12/11/17
Harmonization Service Design	12/8/17	12/19/17
Data Provider Design	12/15/17	12/22/17
Simulator Design	12/22/17	12/28/17
Implementation / Realisation / Integration	12/29/17	2/28/18
Database Creation	12/29/17	1/5/18
Harmonization Service Implementation	1/5/18	2/6/18
Data Provider Implementation	2/5/18	2/19/18
Simulator Implementation	2/19/18	2/28/18
Research on the system behaviour	3/1/18	3/8/18

Figure 2-6: Timeplan / Gantt chart

3. Literature Review

Designing the software and creating a fitting data-schema, as well as making decisions about the project infrastructure requires a deeper understanding of the subject. For this purpose it should be examined in detail with all its components.

The first survey will examine water-plants related topics. It will explain how a water-plant work and what the most important steps of the water cleaning process are. Additionally it will be investigated on which and how water quality indicators are gathered within water-plants.

The second survey will investigate on the topic data harmonization. It will explore the common methods of harmonization and the theory behind them.

The third survey will investigate on similar projects and industries, which also harmonized data coming from different sources for a common use. The advantages and disadvantages of each approach will be taken into consideration to see, which approaches – or parts of approaches, are fitting for this kind of a project.

3.1 Water-Plants

The United States Environmental Protection Agency [10] provides good insight on the topic of water-plants. Water-plants (wastewater-plants or sewage-plants) are used to clean water – primary household, -/ but also industries and businesses sewage – for further use. This is accomplished by speeding up the natural process by which water is purified. Today it is mostly done in three steps which will be explained later.

Water was privatised in the UK in 1989, since then the quality of water and its availability increased, but the administration of each water-plant differed more than before.

3.1.1 Functionality

The picture below shows a water-treatment cycle:
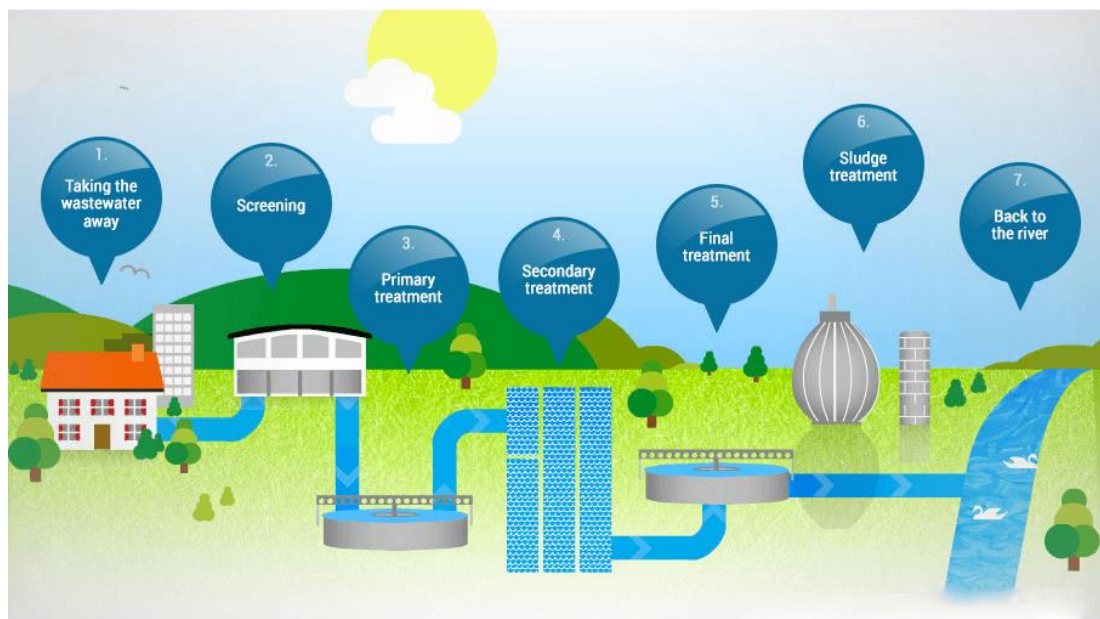
Figure 3-7: *Waste-water treatment plant stages [1]*

The DEFRA [2, pp. 1,2,6,10] gives a good general description of the treatment process. As mentioned before, there are several steps within which the water (already filtered of grit and large solids) gets treated during the cleaning process. Every step is controlled and monitored. Those steps[5] are:

0. Screening:

During the screening, large objects such as rags or boards are dismissed, to prevent them from damaging the instruments of the further treatment.

---

[5] Not the same numbers as in the picture above

1. Primary Treating

Physical and chemical settlement of suspended solid waste which didn't get removed before, as well as reduction of its biochemical oxygen demand.

This step should reduce:

- o Biochemical oxygen demand by 20%-30%
- o Suspended solids by 60%

2. Secondary Treating

This step involves biological treatment to break down and reduce residual organic matter. This treatment step must comply with the standards of the Directive.

3. Tertiary Treating (Final Treatment)

This treatment step depends upon the location. It can involve disinfection by violet light, nutrient removal or the removal of specific toxic substances.

3.1.2 Monitoring equipment and process

In the following chapter, the current state of the art in terms of wastewater treatment monitoring will be described, based on a paper by P.A Vanrolleghem and D.S Lee [11].

The four major subjects when it comes to wastewater treatment monitoring are:

- • Insight into the process
- • Sensors (data providers)

- Fitting strategy

- Actors which implement the controller output

Even though the four mentioned blocks evolved over the past years, the claim on water quality has evolved too and with that, a more advanced treatment for the water is required. Especially the standards and regulations for the level of organic carbon, nitrogen and phosphorus kept getting stricter. For a lot of outdated waste-water treatment plants this fact meant that they were forced to implement some new monitoring technologies, as the only alternative would be increasing the reactor volume – which is not only less flexible, but also more costly.

3.1.2.1 Sensor classification

Functional application: Most often, sensors are used only for monitoring purposes. In this case, the reason behind the monitoring is significant. It might be to provide information to the operators of the water-plant controlling system, or to consultants, whose aim it is to optimize the process.

Complexity: Depending on their purpose, sensory can either be simple, reliable and easy to maintain or advanced and maintenance intensive.

3.1.2.2 Commonly monitored data

The following sub-chapter will provide a list of water-properties, which are commonly monitored, as well as some description on why the data is needed, who it is interesting for and/or how it is measured, for the respective property. This list will be used to design the common schema in a later chapter. The properties are separated into categories which indicate the step of the treatment process they are measured in:

General:

This section describes properties, which are not specific to a certain treatment step, but instead are measured on different stations, as they are of high importance overall.

Temperature: Measured with a thermistor. Important for anaerobic digesters. Not of importance for the public.

Pressure: Measured for alarm functions in aeration and anaerobic digesters. Not of importance for the general public.

Liquid level: Measured with internal electric switch, conductivity switches, pressure transducers, capacitance measurements and ultrasonic level detection. This data is not only interesting for the waste-water treatment plant system operators, for high-level alarms and emergency shut-offs, as well as low-level alarms and leak detections, but might also be interesting for the public to view the storage inventory.

Flow of liquid/gas: Based on the change in water level as a result of an obstacle in the flow path of the water. The efficiency of a wastewater treatment plant is a function which includes the flow rate of the water entering the water-plant.

pH: pH is a value important in all biological processes, but is especially valuable in anaerobic digestion and nitrification. It is measured by pH electrodes. Despite its importance in the control of biological processes, it is not recom-

mended for process supervision and control of waste-water plants as it might be rather insensitive to indicate process changes.

Conductivity: Conductivity sensors are used to monitor influent composition changes. They also help with the control of chemical phosphorus removal.

Biomass/suspended solids: This might be the most important value in the waste water treatment process. It measures the suspended solids concentration (SS). The most common measuring techniques are optical, ultrasound and dielectric spectrometry.

Anaerobic digestion

This process is mineralizing organic material into gaseous products such as H_2, CH_4, CO_2 and H_2S. The measurement of intermediates and the final gaseous product properties is of high value, as it helps with the control of the entire process.

Gaseous products: This measurement gives insight about the gas composition.

Calorimetry: This measurement monitors the heat production and provides direct insight into the biological process as all biological activities are characterized by the production of heat.

Volatile fatty acids (VFA): The VFA is the most important intermediate in the anaerobic digestion process. Not only does it help with the process con-

trol, as values out of range may lead to a process failure, the VFA concentrations also act as performance indicators. Due to the importance of this value and the lack in different implementations, the sensors in this area were developed a lot in the close past.

Activated Sludge

Dissolved oxygen (DO): The sensors measuring the oxygen are most likely the most common sensors in the wastewater treatment plants, as it plays a key role during the activated sludge processes. The aeration, which takes place during this process can take as much as 40% of the overall wastewater plant running costs, and its optimization has therefor a very direct influence on the turnover.

Respirometry: Indicates the respiration rate of activated sludge, defined as the amount of oxygen per unit of volume and time that is consumed by the microorganisms within the activated sludge. It is of importance for the characterization of wastewater and the activated sludge kinetics.

Biological oxygen demand (BOD): This is a measure that indicates the amount of dissolved oxygen required for the biochemical oxidation of the organic solutes within 5 days from the seeding of the test sample in a microbial system. This is not a typical real-time value, as it indicates the quality of wastewater and sludge from 5 days ago.

There are further measurement techniques, but they all require time and the less time they require the less representative their outcomes are.

Chemical oxygen demand (COD): Another highly monitored variable within a water-plant. It determines a plants efficiency in terms of carbon removal.

Total organic carbon (TOC): This measurement converts organic carbon into CO_2 and measures this product in the evolving gas phase.

3.1.3 Difficulties

The problematic in the context of the system under design, with the focus on its usefulness within the real world, lies mainly in the data acquisition. First of all, the data in the UK is not only not available to the public, but also not available to any authorities as online provided real time data. Even though this problem is not in the scope of this dissertation, it will become a problem when the system will be integrated in the real world. Additionally this fact makes defining divergences between water-plants harder.

Sensors which analyze the water and provide information are not entirely reliable. They may provide wrong data, data-holes or provide data in an inconsistent frequency. Those sensors may differ from water-plant to water-plant and be placed on different places within a plant, which makes comparison between different water-plants difficult. The sensors might not provide all the data which is needed to do a representative comparison between different water-plants. Also the data formats may not only differ in form but also in type. While the one might use an XML schema, the other one might use JSON or even a table based format like CSV or XLS/XLSX. While XML and JSON differ in their syntax, JSON and CSV or XLS/XLSX differ in their structure. A comparison between a tree-based data-structure like JSON and table-based data-structure like any format of Excel tables is not trivial and needs to

be defined based on the task. This may lead to a development of two or more different harmonization strategies depending on the source data-format.

Treatment steps may differ from water-plant to water-plant, meaning that even if a sensor is placed on the same position of a primary treatment in two water-plants, the data may still differ a lot.

There is not much real data to develop the system. Real, water-plant-created data, is very useful when designing a harmonizing system, because it gives an idea about the different formats and possible deviations within the schemas. Additionally, having multiple sources gives statistical insight on which data is 'usually' available/tracked, and which is rather rare.

3.2 Data Harmonization

Big parts of the following chapter are based on the "Introduction to Data Harmonization and Modelling" [12] and an interview [13] by A. Guess with A. Kaul.

Data harmonization's aim is to create a single source of information based on multiple sources. The general problem during a harmonization process is, that the different base-sources differ in the form in which their information is provided. To present a clean, harmonized information source, the data needs to be cleared of inaccurate and misleading entries. This means, that harmonization alone might not be enough for every process. To make sure the created information set is useful, it needs to undergo some additional processing.

3.2.1 Why Data Harmonization?

As already mentioned, harmonization is needed when dealing with different sources of information, but trying to get a common knowledge base. Sometimes the same type of information is stored in different types of data-formats. An example for this is the DateTime Format.

The DateTime might be stored in formats like:

- DD/MM/YY
- MM/DD/YY
- DD-MM-YYYY
- YYMMDD
- DD MONTH YYYY
- …

All the formats store the exact same information, but to extract this information further processing is needed. In some cases it might be enough to look at the data to find out which of the formats it is using, but in other cases (DD/MM/YY and MM/DD/YY) it is mandatory to know the format before extracting the needed information.

3.2.2 Single Window Harmonization

The **s**ingle **w**indow **h**armonization (SWH) is defined by the United Nations Economic Commission for Europe (UNECE) [14] as:

*"a system that allows all participants in trade and transport file requested information **in only one place**, in a **standard format**, in order to carry out import, export and transit operations."* [14, p. 3]

It was designed to simplify foreign trade operations, where control agencies require around 40 documents all together, with often repeated data. It includes:

- Simplifying or cutting procedures or documents

- Aligning them with national standards

- Automating documents and processes

The main idea of the SWH is to store information at a Single Authority (Single Window) with traders and control agencies located at different places, but connected information flow.

Even though the basic problem to be solved by the SWH is a different one, since it is coming from a different field, the solution of having a single authority gathering and providing the data to interested stakeholders is also of interest for this project, since it is basically the task. The below picture shows the basic idea:
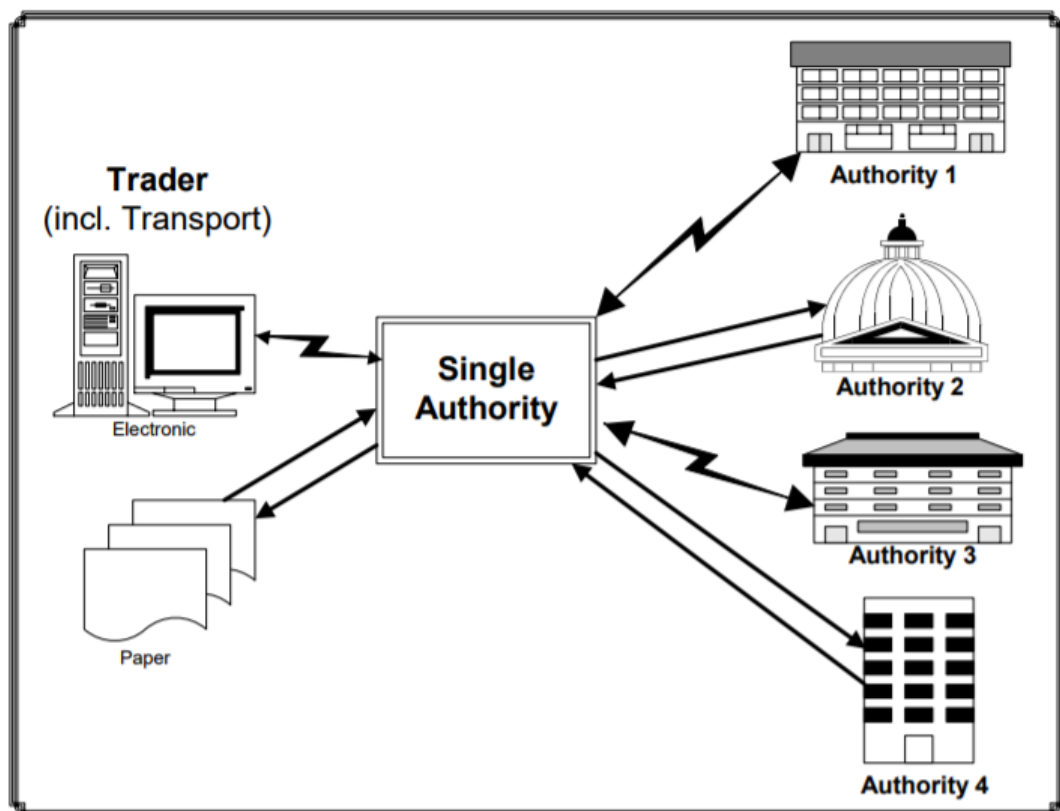

Figure 3-8: Single window system example – big picture [14, p. 4]

Knowledge gathered from UNECEs attempt to create a single window harmonization solution includes following points:

High-level support by strong lead organisations is needed. Those organisations are too important for the system as a whole to not be included and thus have a deciding role in its success. It is important to have a clear vision of the single window system form the beginning, not only to plan the system, but also, since the system is about connecting different sources, to be able to describe the advantages and methodologies in detail and thus get them interested in joining. 90% are negotiations and 10% are technical work. Companies function very differently in terms of their processes and techniques and most of the time every company will want to have its own process as the standard. Agile development of the system is of advantage, as the involved stakeholders want to see progress and avoid misunderstandings. The techniques of this UNECEs approach will be discussed in the chapter 3.3.

3.2.3 Data Simplification

This step eliminates unnecessary and redundant data. It is usually happening before the harmonization takes place. This has the advantage, that in high amounts of data, not all the data needs to undergo further processing. The disadvantage to have it as first step is, that some data might be removed that is actually needed, but not recognized as such and would be only after the harmonization step. A good approach is to have a simplification take place once before and once after the harmonization took place. The first simplification in this approach aims at clearly unneeded data and leave out data classified as 'unsure' to handle it in the second iteration after the harmonization is done.

Data sources are likely to have additional data, which is only relevant for the specific source itself and not for inter-source-analyses. This data needs to be identified and 'cleaned' from the source dataset.

Example with two different sources A and B:

Target Schema:

DataA
DataB
DataC

Source A:

Source B:

Figure 3-9: Data Simplification example

The Simplification requires to know the target schema to function. It doesn't add values which are missing when comparing source to target. It only removes the unnecessary and blacklisted entries.

3.2.4 Data Standardization

Data standardization is about processing of the dataset into a standard form through standard bodies. It is not said that a harmonized dataset matches the standard forms outside of the system. It might be harmonized but still useless to the outside world. This is why a standardization step might be needed to do further processing of the data, change its form or format, before publishing it. An example for standardization is converting the velocity from one of its many units like meters/s, km/h, knots… or temperature from Celsius, Fahrenheit or Kelvin into the one which is standard in the country/field of interest, since they all contain the same information in a different form.

3.3 Comparable Industries and Projects

To find the most fitting approach for the system, it is needed to take a look on projects and industries which already successfully developed such a system, or projects where such a system failed, to see what caused its success or failure, as well as understanding why specific approaches were more successful than others.

3.3.1 UNECE SWH Project

As already mentioned in a previous chapter, in the year 2008 the UNECE attempted to build a harmonized single window system for international trading. [15] This project lead the UNECE, in cooperation with UNNEXT (United Nations Network of Experts for Paperless Trade in Asia and the Pacific) and the United Nations ESCAP (Economic and Social Commission for Asia and the Pacific) to create a report on how the data harmonization should be opti-

mally approached. Since the UNECE project had a different context, only the parts of the report significant for this project will be investigated upon, as well as only the technique of harmonization and not the project planning.

One difference to this project is, that this project should also consider real time data, while the UNECE project defined a schema which is only complete with all elements filled out. This should not be a problem in the context of this dissertation, as there is the option to simply ignore missing data if it was not provided during the time period in which the water-plant data was created. The timestamp of each data entry in most cases will have no significance in the UNECE project, while a data entry without a timestamp is basically useless in this system.

The UML provides sufficient options to describe a Data Model as a Class Diagram with Property Terms[6] and Object Classes[7].

The following describes the five main steps taken to develop a data harmonization process, transformed into steps in the scope of this project.

Step 1: Everything starts with the capture of data requirements. This step is about collecting information about which data is interesting, as well as how this data is produced. This includes getting background and understanding of the information source work-flow. Understanding the work-flow helps in understanding the stakeholders needs, as well as the significance of specific data members.

Step 2: Providing a detailed definition of the data elements within single information sources. This step is about setting the data definition, type, format and constraints of each information type for each information source. The

---

[6] Describes a piece of information used to describe an object, such as "Name"
[7] Describes a set of Property Terms, such as "Person"

outcome of this step should be a data dictionary corresponding to a specific source of information.

Step 3 & 4: Analysing data elements across various information sources. This includes the organisation of data elements in a comparable manner so that it can be used for analyses. The desired outcome of this step is a data dictionary compilation as well as mapping to the desired data model.

Step 5: This step is about the creation of reports and not in scope of this project.

3.3.2 FCTC Project

In the year 2008, the Conference of the Parties to the WHO (**W**orld **H**ealth **O**rganization) requested **F**ramework **C**onvention on **T**obacco **C**ontrol (FCTC) to compile a report on data collection measures [16]. Within two expert meetings (2009 and 2010) the draft outlines were defined, for the further development of the report. It was supposed to be based on the most relevant international literature and other tobacco-related information sources. A review of existing data sources and data collection systems as well as an investigation on a possible data harmonization process was also a part of this meeting. The report created during this meeting gives an insight on possible problems which may come up during the creation of the system, as well as criteria to consider when deciding for a harmonization technique. It is especially interesting, as the tobacco control is probably one of the most researched areas in public health.

The international data collection systems of tobacco-related information were distinguished in two types: **population-based surveys** (primary data collec-

tion systems) and **policy monitoring surveys/systems** (secondary data collection systems).

Additionally, many countries have they own health survey systems, which have no direct relation to the international system despite their information being similar.

The reporting system of the Convention was established by the conference of the Parties in 2006 and enforces every participating party to report on its implementation of the convention for the first two years after being included in the convention, as well as after a specific time in the future. The aim of the system was to collect data already available at the time of reporting, instead of having to implement the new population based survey system first. Only the parties without any population-based system were required to implement it. The convention secretariat provided **feedback on the assigned reports**. This feedback included missing mandatory information, not suitable formats within the report and inconsistencies within the answers on the report and the supporting documentations. This data is then provided through a web based database maintained by the convention secretariat for further reports.

Another source of information on tobacco is the impressive number of databases which provide tobacco-related information.

Standardization and harmonization: Even though the majority of surveys follow standard methodologies and use standardized questions and patterns, the form still differs from survey to survey and therefore requires at the very least some form of harmonization. Additionally, as already mentioned, the majority of surveys are non-tobacco specific ones and thus need to be "cleaned" of unnecessary (not tobacco related) data. Additionally, the surveys

may differ based on the age, since the questions might be directed more in a specific direction for a specific target group. It has to be made sure, that data which has the same identifier across multiple sources has also **the same definition** across those sources. Comparing seemingly the same datatype, even though it means something completely different within the scope of different systems will cause inconsistencies and false conclusions. Another important knowledge won in this project is, that there is a big amount of **redundant, repeated and overlapping data** across the different surveys and approaches.

In the case of population-based surveys it is important to have a common information schema, methodology and data definition, to create comparable data sources. In the case of monitoring systems, harmonization of the existing data is the main challenge. This knowledge transferred to this project means that especially in the case of historical data and data provided by the existing systems, the main challenge is to find a way to harmonize it, while it is desirable to have every participant use a common, efficient and suiting data-schema. Accomplishing this desire is a hard challenge.

Standardization of data not only refers to processing of the acquired data, but also to the **data acquisition / creation** (such as sampling techniques), in order to create more meaningful data. This may require additional training (i.e. workshops) for the data collectors. Step-by-step instructions in how to implement or use the new system might be needed in order to promote it.

Even if some parties may not be able, or may not want to fully adapt the common approach, they can still undertake steps to help the system to gather their data in a better way, like adding additional data or adjusting some data definitions. Additionally, experts have warned from creating a global

standardized approach in terms of survey content, as this would take away the variation between regions, where specific information might be of high value in some regions and help the parties to fulfil their reporting obligations, while it is of no use in others.

Differences: There is a lot of knowledge which can be won from the FCTC project, but some of it can't be fully or not even partially applied to this project because the two projects differ too much in their aims and objectives. This project needs a way to deal with real-time data. It also needs a way of adapting data into the own schema, as historical data placed on some storage will most likely not be pre-processed by the provider.

4. Design and Implementation

The key parts of the system to design are the data-schema, the database / storage and the processing / harmonization service. All the mentioned parts can be – to a degree – designed and implemented independently, if the interfaces are clearly defined. Additionally the harmonization service will process the data in a pipeline, which will get described later.

4.1 System overview

The below picture shows a white box diagram of the system, for a better understanding of its scope and relations:
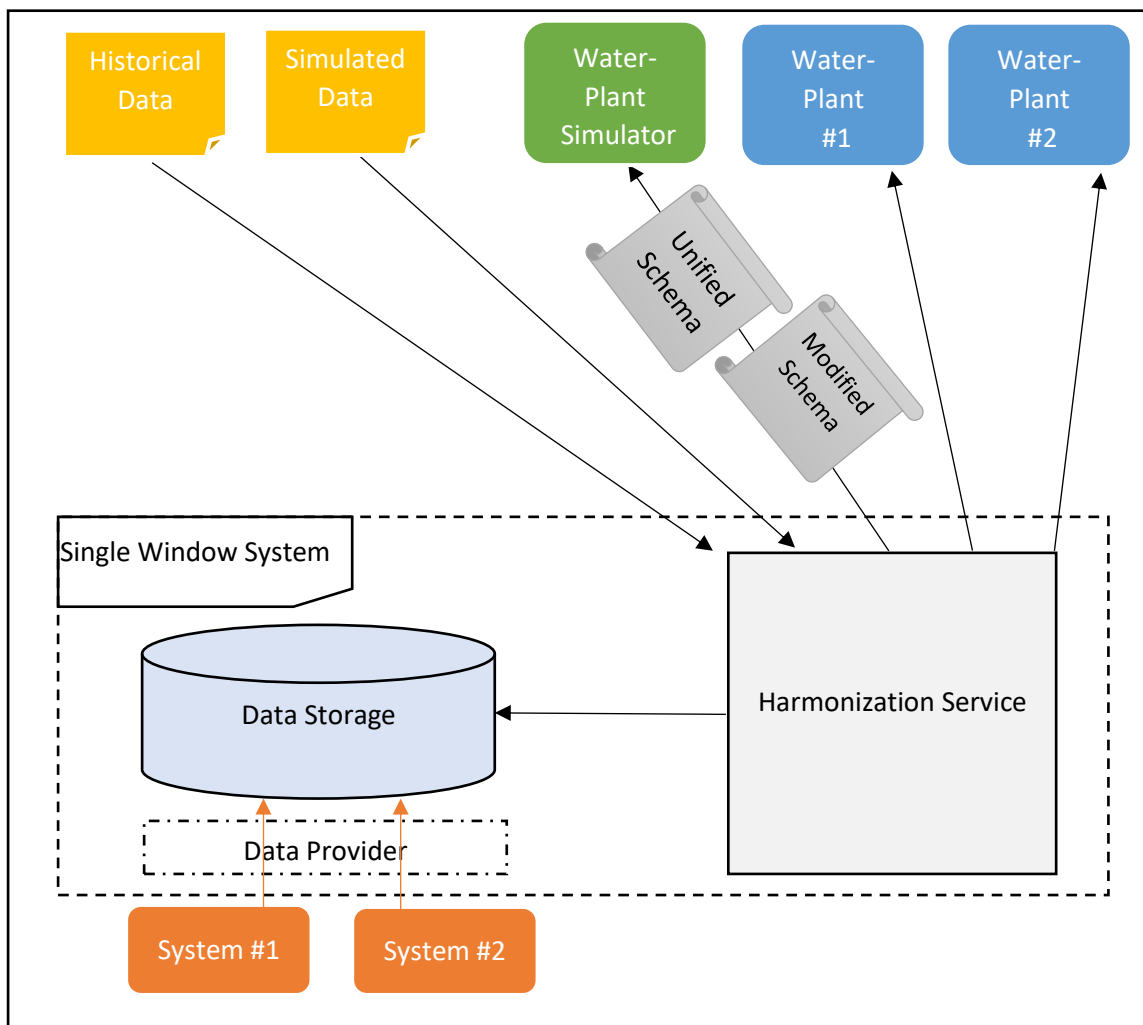

Figure 4-10: *Whitebox diagram – harmonization system*

The picture shows five data sources which can be split into two categories. Those two types will be referred to as **real-time data** (green and blue) and **report data** (yellow).

It is assumed, that real-time data is being pulled by the system frequently from the same (registered) source. If there is no new data, the system waits a defined time and then tries to pull the data again.

On the other hand, the report data is not pulled by the system, but rather provided through a dedicated API.

The unified schema is known to the harmonization system, and in best case the data from real-time sources and reports is coming in that format. It can happen, that the system receives data which is not in the unified schema, but a modified one. A modified schema will be defined as a schema that differs in any way from the unified schema. The harmonization system needs a way to deal with this kind of schemas. The report data will usually include a bigger amount of data in one session.

Real-time data should be of higher priority to process by the system, as the real-time character will fade with time passing.

Data processed by the harmonization system will be stored on a data-storage. This storage will not be public, but will be accessible for reading by 3rd party systems through a data provider service. The advantage of this forwarding service is, that the system can provide a consistent API, even with internal changes on the database and that the access rights and load balancing can be easier handled with an API gateway provided by the cloud.
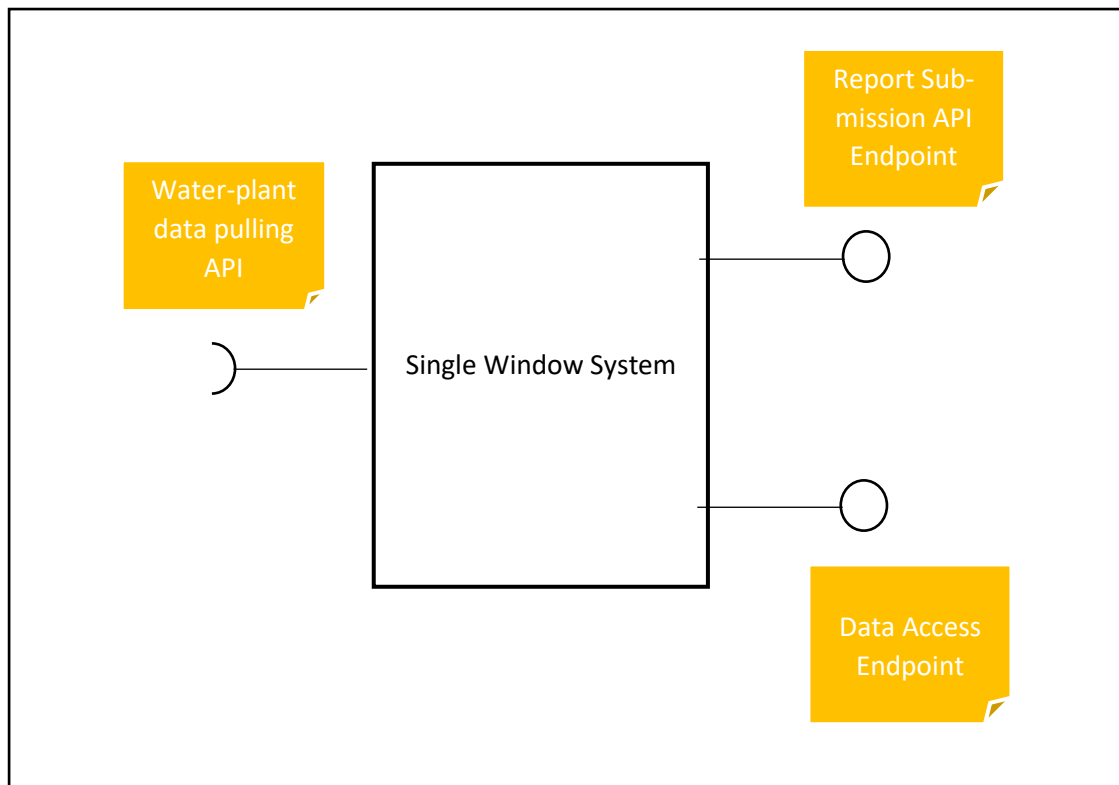
Figure 4-11*: Blackbox diagram – harmonization system and its interfaces*

Putting all the facts together, the picture shows the system as a black-box diagram with all the mandatory interfaces to the outside world.

The endpoints provided by the system will be described in detail in a later chapter. There is also a need for the water-plants to provide an endpoint for the system to access the data. This endpoint will also be described in a later chapter.

4.2 Data-Schema

The final schema, which is the product of this chapter can be seen in the **Appendix (A).**

The development of a common schema is one of the most important parts of this dissertation. It has to take into consideration, that the data-sources may vary a lot in their format, type of data and frequency, but also be specific enough to provide meaningful data for further analyses. Some values needed

to determine a key-factor may not be available on a data source, or only as an interpolation. Some values may not be as current as other values, or update equally frequent. An important factor for the data-schema is the fact, that it has to be extendable, as well as still remain consistent and valid if some of the elements are deleted.

4.2.1 Data Categories

At first categories of data need to be established to ensure a better understanding and overview over their source, meaning and purpose. In this project those categories will be:

- **Wastewater treatment-plant metadata**

- **Treatment step metadata**

- **Water quality data**

4.2.1.1 Description

In the following those categories will be described in more detail:

Wastewater treatment-plant metadata (WWTPM)

This category consists of data specific for a corresponding water-plant. It includes data which helps to not only identify the source of data, but also boundaries which describe the location and further static[8] metadata.

Treatment step-type metadata (TSM)

Measurements are done at various places of the waste-water treatment process. The value of a water quality indicator will vary depending on the place it

---

[8] Static in this case means – it is not changing very often. A change in this data might mean the historical data of this source might not be comparable to the current data.

is measured and thus will be of no value without a specification of the sensor position.

Water quality indicator data (WQID)

This category will consist of data described in a previous chapter (chapter 3.1.2.2). An entry not only requires a value, but also the timestamp and unit to be of full use for further analyses.

4.2.1.2 Relations

The following diagram shows the relational hierarchy of the three categories:

Figure 4-12: *Data-schema relations and hierarchy*

One water-plant may contain any number of TSMs, although at the moment the maximum number will be set to 7 as this is the number of defined treatment-step types. Each step may contain any number of WQIDs, although the maximum number in this project will be set to 16 as this is the number of de-

fined indicators (described in the chapter 3.1.2.2 – Commonly monitored data). Additionally to this hierarchy, there might also be some loose[9] variable data, like amount of water within water containers. This data will be assigned directly to the water-plant without any direct relation to a specific treatment step.

4.2.2 Data-Schema content

The following data will be included in each of the categories. Note that the property names will be actually written as camel case to follow the defined JSON standards and conventions:

4.2.2.1 WWTPM

Description of how a WWTPM entity looks:

Property	Description	Required?
Name	This property represents the name of the waste-water treatment plant. The name has to be mandatory as it acts as the primary key, which has to track back a specific plant.	✓
Location	This property indicates the geographical location of the water plant. The location is of interest in terms boundary information like weather. Location also includes the actual country, as UK consists of 4 (England, Wales, Northern Ireland, Scotland), which might have some water regulations for their own.	
Supplier	This property refers to the water-supplier owning or controlling the corresponding water-plant. Even though this property is not required, it is of high value.	

Table 4-3: Waste-water treatment plant in the context of this system

---
[9] Not belonging to a specific treatment step

4.2.2.2 TSM

Description of how a TSM entity looks:

Property	Description	Required?

Name	This property represents the name of the treatment step. Treatment step might be one of the following values:
	   o Entry
	   o Screening
	   o Primary Treatment
	   o Secondary Treatment
	   o Tertiary Treatment
	   o Sludge Treatment
	   o Exit
		✔

Table 4-4: Treatment step metadata in the context of this system

4.2.2.3 WQID

Properties

Description of how a WQID entity looks:

Property	Description	Required?
Name	Name of the water quality indicator (possible indicators will be described later)	✔
Timestamp	Timestamp of when the indicator was measured. It should include the Date and the time.	✔
Unit	The unit of the indicator.	(✔)[10]
Value	Value of the indicator	✔

Table 4-5: Water quality indicator in the context of this system

---

[10] The unit is in the most cases the same for the same type of indicator. It is good to have it passed, since this prevents errors, but storing the values with a "default" unit will be preferred to dismissing the values in total.

Possible Entries

The following table shows an overview of possible Water Quality Indicators, their units and known aliases. Known aliases are helpful when it comes to harmonization, since names can be mapped without having to make speculations or unnecessary calculations.

Name	Unit	Known Aliases
Temperature	C° (Celsius)	Temp
Pressure	Pa (Pascal)	
Liquid level		
Flow Rate (liquid / gas)		Flow; Liquid Flow; Gas Flow
pH	[no unit]	
Conductivity	S/m	
Biomass	mg/L	Suspended solids; Total Suspended Solids; TSS
Nitrate	mg/L	NO_3
Nitrogen Dioxide	mg/L	NO_2
Ammonium	mg/L	NH_4
Reduction Potential	mV	ORP
Calorimetry	C°	
Volatile fatty acids	mg/L	VFA
Dissolved oxygen	mg/L	DO
Biological oxygen demand	mg/L	BOD
Total organic carbon	mg/L	TOC

Table 4-6: Basic quality indicator types

Even though defined at this point, this list isn't final or will ever be hard coded into the software. It has to have an extension possibility as measurements are developing as well. To keep this as generic as possible, it will require the data provider to also return a list of available indicators for a specific plant. This topic will be further described in a later chapter.

4.3 Data Base

The database will act as the platform for many comparisons – to recognize patterns, map names and find mistakes and therefore needs to be designed to be performant. Additionally, requests from the outside world will be performing selects on the same database, which makes its scalability even more important. Note that even though the names of tables will partially equal the names of the data-schema properties, they might not represent the exactly same thing and will thus be defined in the following chapter.

4.3.1 Data Base Schema

The following picture presents the database schema including the names of the tables, properties, unique primary keys and foreign keys:
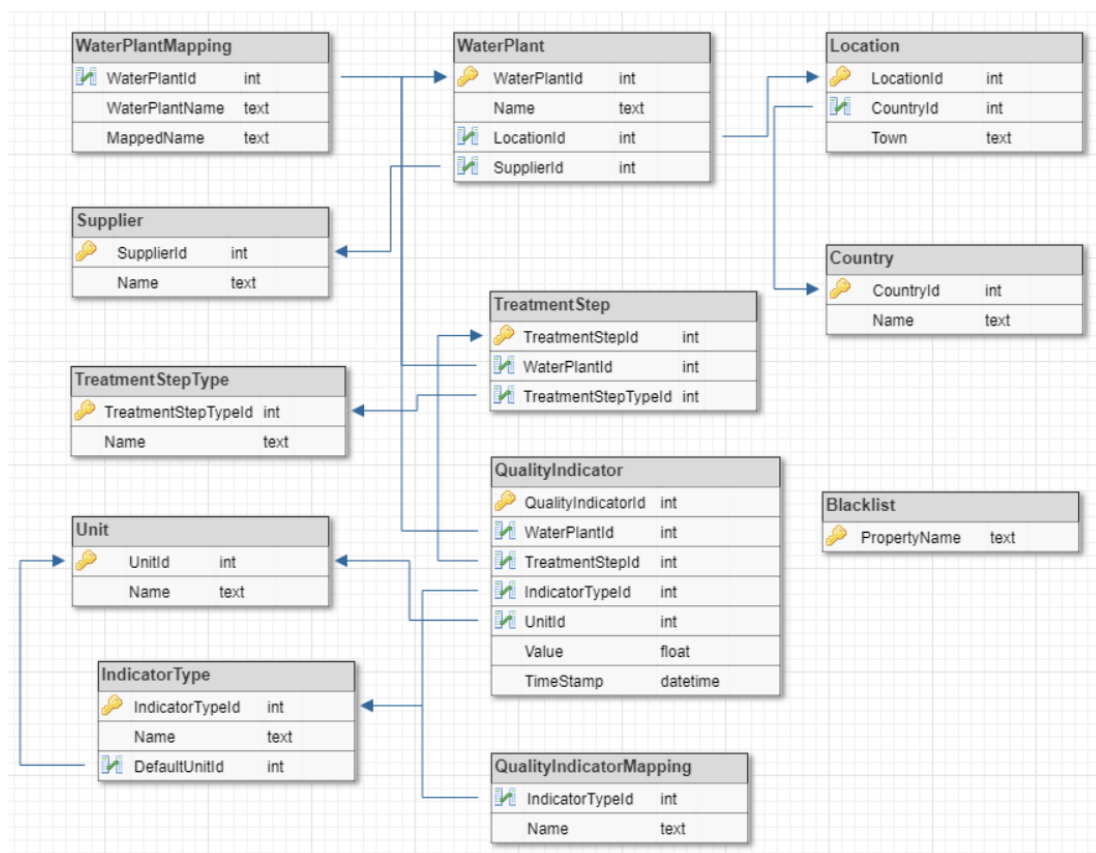

Figure 4-13: *Database schema*

There are no m:n relations to assure better data integrity and subtypes are extracted to make selects as performant and extensible as possible.

Following is a short description of the tables in the database, not mentioning the types and their properties:

Table	Description	Dependencies
WaterPlant	The WaterPlant table represents the physical water-plants. It includes the unique names of the various water-plants.	• Location • Supplier
WaterPlantMapping	This table contains names which are confirmed to be mapped from the specific WaterPlant world to this system, but only apply to a specific plant.	• WaterPlantId
Location	The location describes a graphical location.	• Country
Country	This table represents a country.	
Supplier	Supplier represents a water supplier.	
TreatmentStep	TreatmentStep represents the water-treatment step within the treatment process.	• TreatmentStepType • WaterPlant
TreatmentStepType	TreatmentStepType contains all the unique names of possible treatment steps.	
QualityIndicator	QualityIndicator refers to one of the values which describe the quality of water. A quality indicator can reference **either** a WaterPlant or a TreatmentStep.	• Unit • IndicatorType • WaterPlant • TreatmentStep
Unit	Unit refers to the unit directly corresponding one or more qualityIndicator types	
IndicatorType	The IndicatorType Table contains all possible qualityIndicator types which can be stored in the system. It also contains the corresponding default unit.	• QualityIndicatorMapping • Unit
QualityIndicatorMapping	The QualityIndicatorMapping table consists of known aliases for QualityIndicator names.	
BlackList	Names of properties found on this map will be ignored / cleaned from the data during the harmonization process	

Table 4-7: *Database schema description: Tables, description and dependencies*

4.4 Harmonization Service

The core service of this system is the harmonization service. Its purpose is not only to process the data into a common schema, but also receive incoming data and pull data from defined sources, as well as storing the data in the corresponding tables in the database. The data is processed step by step. The earlier in the process different formats[11] get combined together into a common format, the less work has to be done when extending the system by new formats. Investigations have shown, that the formats are too different to combine them before the actual harmonization step, and need separate treatment in the previous steps. The harmonization strategies of each format will be described in a later chapter.

4.4.1 Use Case Diagram

The use case diagram displays the functionality provided by the system, as well as the actors interacting with it:
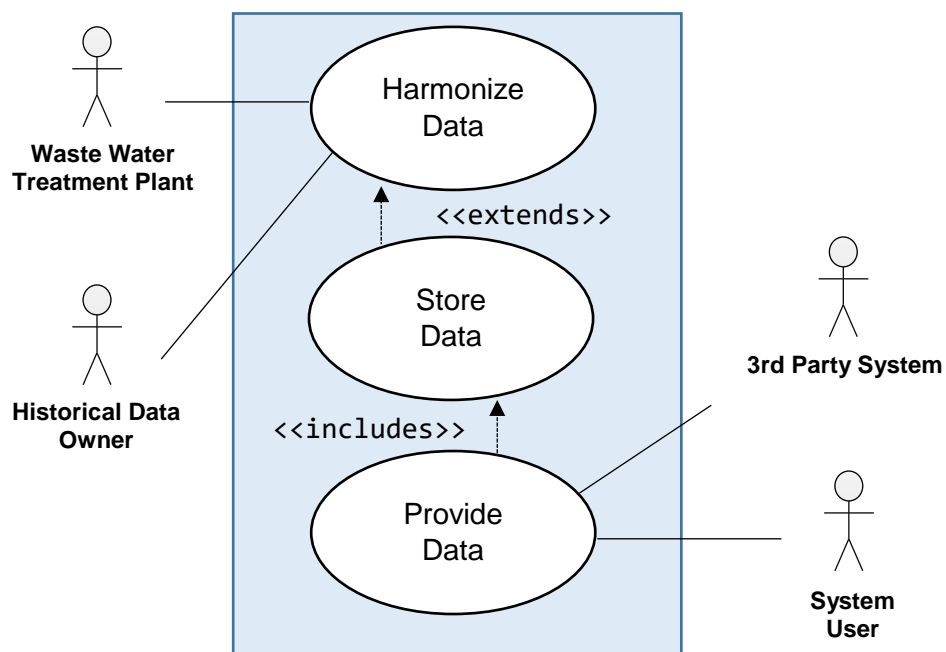
Figure 4-14: Use case diagram

---

[11] Not trivially comparable formats like tree-based format and table-based format

Waste water treatment plants added to the system can provide data for harmonization. Anyone owning historical data in a certain format can enter it into the system. The stored data can be requested and read by other systems and system users.

4.4.2 Component Architecture

The following diagram displays an overview over the components of the harmonization service, and their relations.

Figure 4-15: Component diagram of the harmonization service

The service is separated into 7 components, which are described in the following subchapters:

4.4.2.1 WaterPlantFetcher

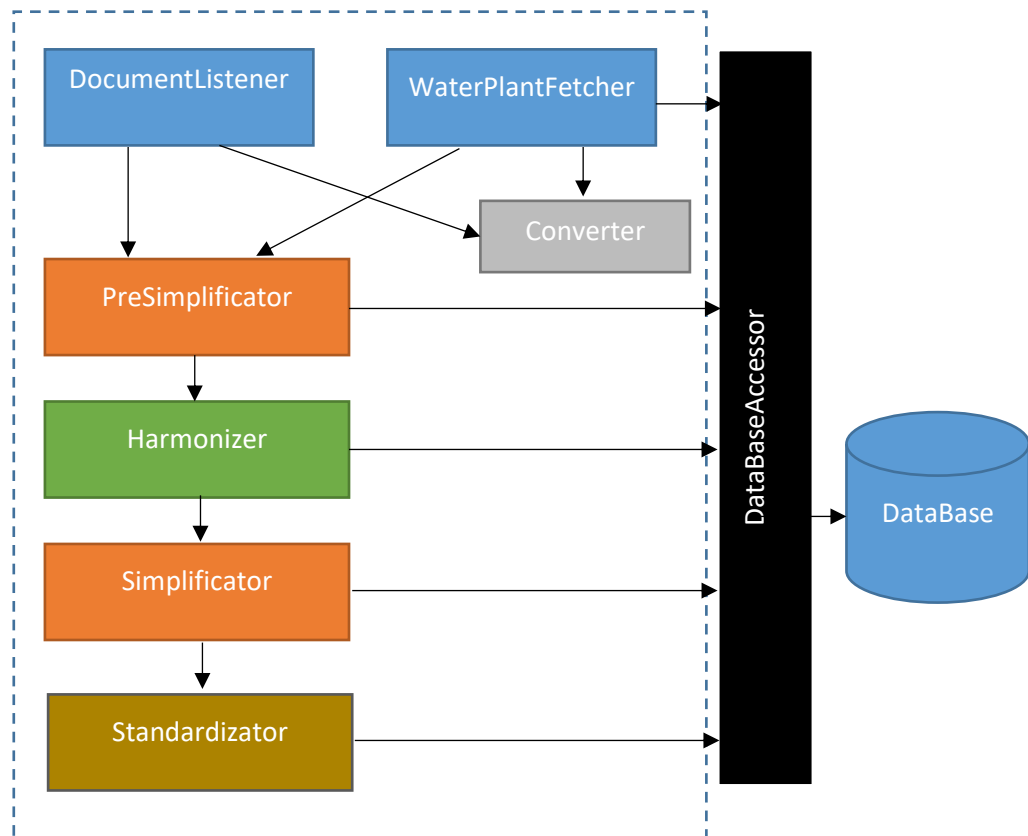The *WaterPlantFetcher* component is responsible for pulling data from de-
fined sources (Water-Plants). The data has to be under the configured path
and preferably in the defined schema[12]. The sources are stored in a separate
table with the following format:

Column Name	Type	Nullable
PullSourceId	Int	
SourcePath	String(100)	
WaterPlantId	Int	Yes
TreatmentStepTypeId	Int	Yes
QualityIndicatorTypeId	Int	Yes
DataType	String(10)	

Table 4-8: PullSources table definition

SourcePath defines the path from which the data is pulled. It expects a direc-
tory with any number of files meant to be harmonized. DataType gives infor-
mation about which of the allowed datatypes is pulled. If the data is stored as
JSON WaterPlantId, TreatmentStepTypeId and QualityIndicatorTypeId can
be set to null (provided, that they are defined in the JSON file). Files which
got processed do not get deleted, but renamed to "_filename". This requires
the client to have writing permissions on the directory, otherwise the same
file might get processed over and over again. The fetched data is not further
processed in this step, but passed over to the harmonization process.

4.4.2.2 DocumentListener

The *DocumentListener*, on the contrary to the *WaterPlantFetcher*, does not
actively pull data from specific sources, but waits until data is provided by the
outside world. For this matter, it provides an Endpoint which accepts docu-

---

[12] The common schema defined in a previous chapter

ments. It does not further process the data itself. Data provided through this endpoint requires more information, as the sender is "unknown"[13] to the system. The Rest API is described in the Appendix C.

4.4.2.3 Converter

The *Converter* component is used by both, the *WaterPlantFetcher* and the *DocumentListener*. Its purpose is to accept data and try to parse it into an object, which can then be further processed. If the *Converter* fails to parse the data, its processing ends at this step and the data is dismissed because this means it is not harmonizable by the system by any means.

4.4.2.4 PreSimplificator

The purpose of the *PreSimplificator* component is to minimize the data which will be further processed by removing the parts of it, which can already at this point be recognized and mapped to parameters in the blacklist. This step does not dismiss any further data or decide if the data will be further processed. Once its task is done, the data gets passed to the *Harmonizer* component. The *PreSimplificator* subscribes to the two system entry points and gets notified when new data is ready to start a harmonization process.

4.4.2.5 Harmonizer

The *Harmonizer* component is responsible for the actual harmonization of the data. It searches the object for familiar data-formats and structures, properties, dismisses incomplete data (with missing mandatory properties). Before doing so an attempt is made to "fix" the data. This attempt varies depending on the type of data. After finishing the harmonization process, the *Harmonizer* checks if the final data-object is complete (no missing mandatory proper-

---

[13] The sender cannot be tracked back through the configuration

ties), and if so, the object is passed on to the *Simplificator*. As already mentioned, accepted data-formats are too diverse to handle them the same way (table-based formats / tree-based formats…). This step is the first step in the chain which has the same output format for every incoming format.

4.4.2.6 Simplificator

The harmonized data is further reduced in this step by comparing the harmonized object to the blacklist.

4.4.2.7 Standardizator

The *Standardizator* represents the last step of the harmonization process and is responsible for adjusting the data to the conventions set in the system. The formats and structures are finally adjusted in this step before the *Standardizator* pushes the object into the database.

4.4.2.8 DataBaseAccessor

A general component used by many services. Its purpose is to act as an abstraction layer between the components and the database. (Not an exclusive part of the harmonization service). This component uses the entity framework to access views and stored procedures. It does not access database tables directly.

4.4.3 Accepted Data Formats

The two described entry points to the system – the *WaterPlantFetcher* and the *DocumentListener* – both expect a valid file in one of the following commonly used formats:

- JSON
- CSV

- XLS/XLSX

It is important to mention that JSON is a tree-based format, while CSV and XLS/XLSX both represent table-based formats. This fact requires a different processing strategy for both of the base formats up until the *Harmonizer* component.

4.4.4 Harmonization Workflow

The detailed functionality and workflow of each component depends upon the format of provided data. While data coming in in the predefined JSON format does not need any mapping or further processing, XLS/XLSX and CSV data, as well as different kinds of JSON files require conversion.

The following chapters describe methods which help to process different kinds of formats.

4.4.4.1 CSV / XLS/XLSX Parsing

CSV (Comma Separated Values) and XLS/XLSX (Excel Spreadsheet) are both commonly used file-formats for table-formatted data storage. To create a parsing method which is as generic as possible, yet still usable it is needed to investigate on example data coming from water-plants, as well as define what criteria needs to be fulfilled to make a file usable.

Common Table Structure: A table needs to at least have timestamps, indicator values and indicator names to be of use. If no unit is detected or found, the default unit is used (data is then validated by comparing to the average values and standard deviations of the same datatype). If there are more value columns detected than indicator names, the data is dismissed, as this provides risk of data misinterpretation by assigning the wrong indicator name

to the wrong row. Boundaries like step or water-plant metadata will be required as URI or configuration parameters in the first step, for this base-format.

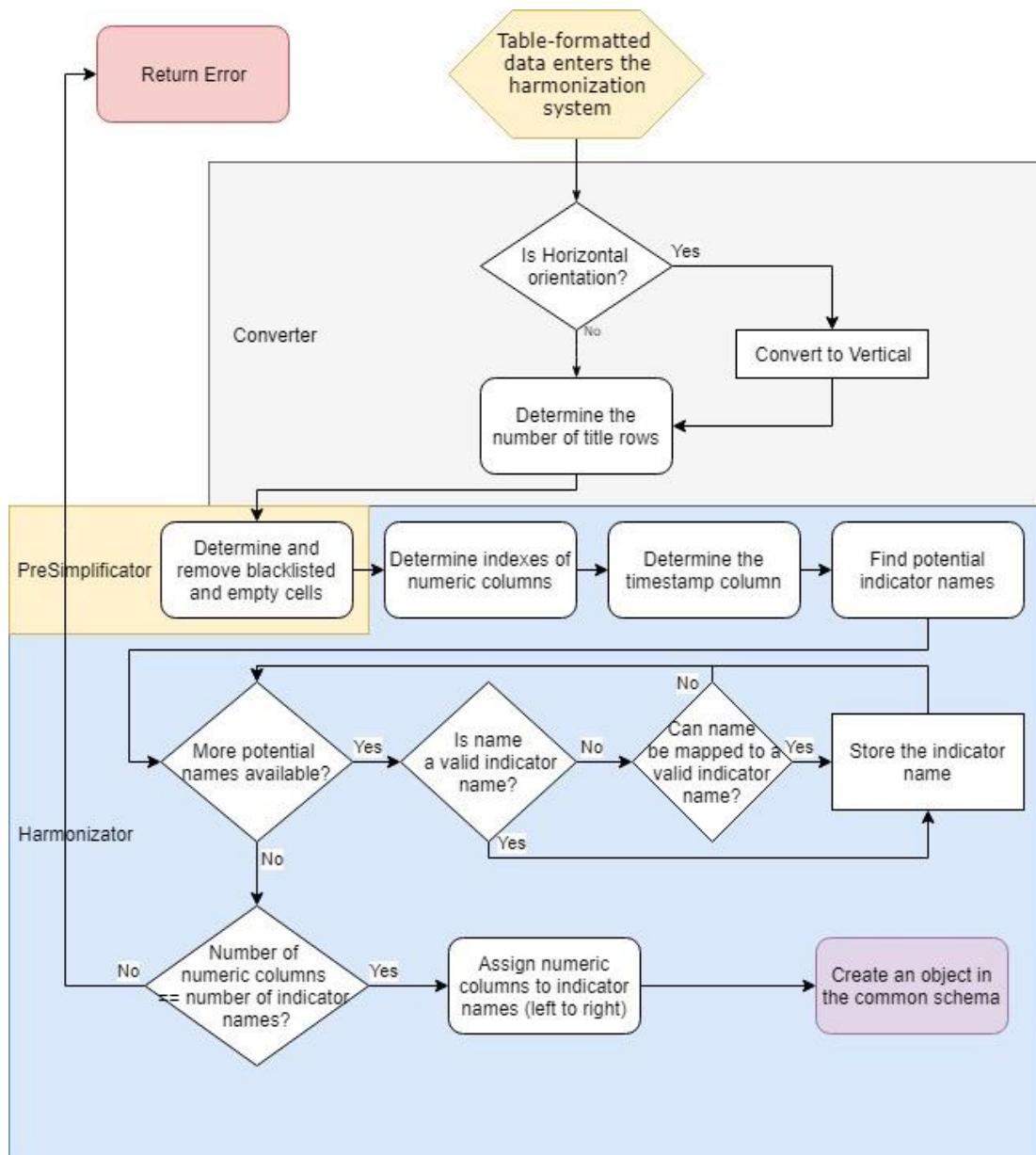**Harmonization Strategy workflow:**

Figure 4-16: Workflow diagram – table formatted data

The last step of the workflow is also the last step of the *Harmonizer*. It creates a valid object in the common water-plant schema. Further processing is

equal tree and table based data and consists of cleaning and storing the data.

Problematics:

- Finding the name of the indicator after identifying a value column (title columns might not always have the same cell size per indicator and have differently placed names)

- Finding redundant columns (indicator can be provided in multiple units)

- Identifying the correct timestamp format (might not always be possible – best effort principle)

Example of a Table:

	N20	1,928344	CH4		0,702761	CO2		1,928212	H2O		N
		200	385,6688		50	35,13807		3	57,84637		20000	
	ppm	mg/m3		ppm	mg/m3		%	g/m3		ppm	
4.10.16 13.25	Z	0,046199	0,089	Z	1,00237	0,704	Z	0,195815	3,776	Z	1460,09	Z
4.10.16 13.26	Z	0	0,000	Z	0	0,000	Z	6,47E-05	0,001	Z	40,8642	Z
4.10.16 13.27	Z	0	0,000	Z	0	0,000	Z	0	0,000	Z	24,2549	Z
4.10.16 13.28	Z	0	0,000	Z	0	0,000	Z	0	0,000	Z	20,1773	Z
4.10.16 13.29	M	0,022649	0,044	M	3,21124	2,257	M	0,273937	5,282	M	2861,04	M
4.10.16 13.30	M	0,002997	0,006	M	6,10013	4,287	M	0,188628	3,637	M	3727,03	M
4.10.16 13.31	M	0	0,000	M	2,986	2,098	M	0,188532	3,635	M	3734,9	M
4.10.16 13.32	M	0	0,000	M	2,34712	1,649	M	0,20779	4,007	M	3724,94	M
4.10.16 13.33	M	0	0,000	M	1,74299	1,225	M	0,191868	3,700	M	3708,45	M
4.10.16 13.34	M	0	0,000	M	1,73801	1,221	M	0,213847	4,123	M	3698,46	M
4.10.16 13.35	M	0	0,000	M	2,55001	1,792	M	0,282027	5,438	M	3699,75	M
4.10.16 13.36	M	0	0,000	M	4,0402	2,839	M	0,315758	6,088	M	3704,8	M
4.10.16 13.37	M	0	0,000	M	4,65569	3,272	M	0,395623	7,628	M	3720,35	M
4.10.16 13.38	M	0,000721	0,001	M	10,246	7,200	M	0,368239	7,100	M	3738,32	M
4.10.16 13.39	M	0	0,000	M	9,19548	6,462	M	0,262311	6,084	M	3767,03	M

Figure 4-17: Example table

Process of entire harmonization on example table:

1. Table is converted into an object containing *title rows* (containing everything of colour) and *payload columns* (everything else). (*Converter*)

2. Object is cleaned of payload columns which cannot be interpreted by the system (i.e. columns only containing Z and M) (*PreSimplificator*)

3. The data is matched into the predefined JSON schema object by try-ing to map payload values with their indicator names and units in the title. (*Harmonizer*)

4. The now harmonized data is cleaned of properties which are not tracked by the system (i.e. H2O) (*Simplificator*)

5. Adjusting formats / units and storying in the database (*Standardizer*)

4.4.4.2 JSON Parsing

Even though in best case the incoming data is already in the predefined schema, a usual case is that the data is indeed in a JSON format, but the structure differs from the predefined one. For this matter, the previously men-tioned *dynamic* data-type will be of high importance. One of the biggest strengths of a tree-based format like JSON is its expandability. There is no limit to the children nodes and nesting levels. This requires for a recursive solution. Additionally, the datatypes of the properties can reach from simple types like strings or numeri's to complex types like further JSON objects. The core of the harmonization process will be the cast of a JSON object to a dic-tionary of string and dynamic pairs, where the string represents the property name and dynamic the actual object. Mapping of properties is of high im-portance, to interpret the object correctly.

Harmonization Strategy Workflow:

The following flow-chart shows the harmonization strategy in the case of a JSON formatted object entering the system. The current data type and con-versions will be mentioned in the chart, as the dynamic type requires the de-veloper to keep track of the type at all times.
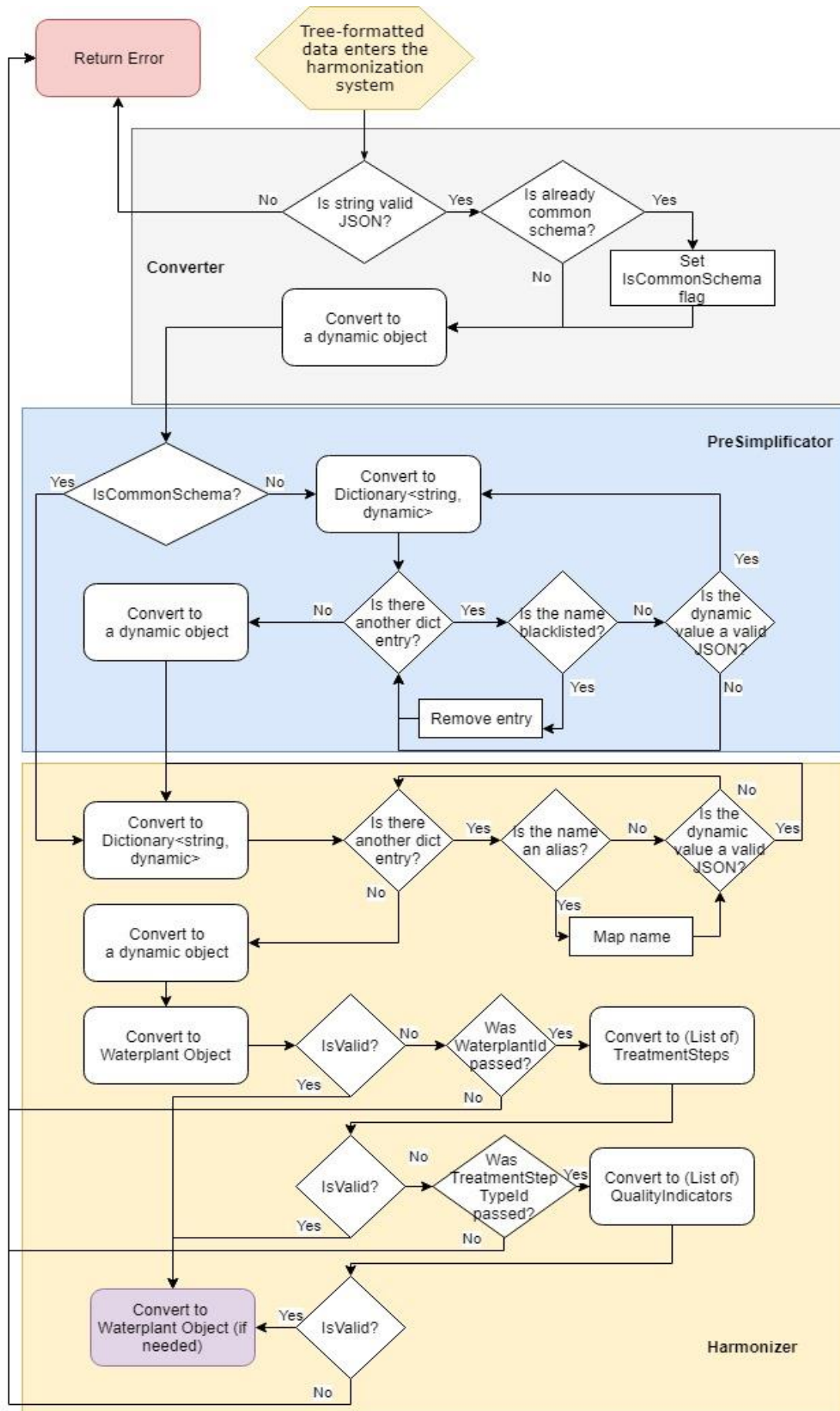
***Figure 4-18**: Workflow diagram – tree formatted data*

The workflow of tree-based harmonization is more complex than the one of the table-based strategy. It requires two recursive sub-workflows (which are hard to display on a workflow chart). Additionally the number of passed URI parameters is of high importance for the casting options. If the user i.e. has passed the water plant id, it is an option to convert the payload to only the treatment steps, and retrieve the water plant metadata from the database based on the id.

4.5 Data Provider

The purpose of the data provider is to act as an abstraction layer between the data storage of the system and the outside world. The data provider accepts data requests through a HTTP listener and provides the requested data.
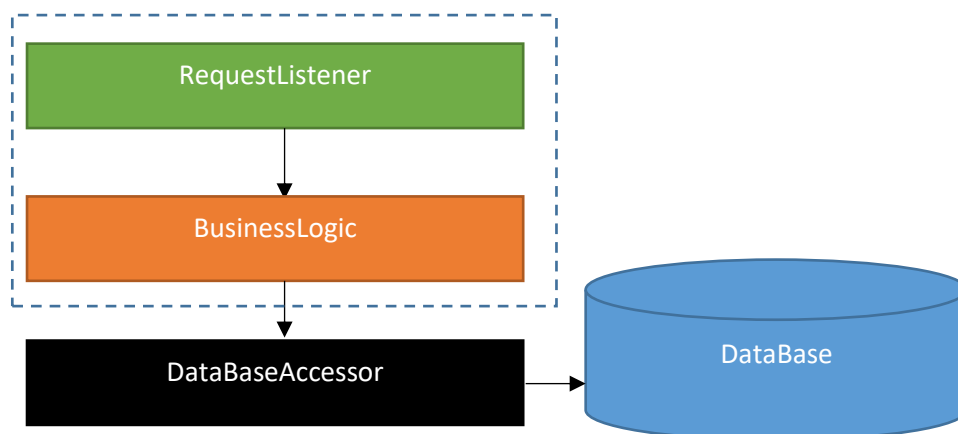
4.5.1 Component Architecture

Figure 4-19: Component diagram of the data provider

4.5.1.1 RequestListener

The *RequestListener* provides an API (described in a later chapter) to access the data stored in the system. The requests converted and forwarded to the *BusinessLogic* component

4.5.1.2 BusinessLogic

The *BusinessLogic* is responsible for the processing of the requests. It is also responsible for "combined database calls"[14].

4.5.2 API

The REST API is described in Appendix B. It follows the common REST rules and provides 4 endpoints:

- GetWaterPlants
- GetTreatmentStepTypes
- GetQualityIndicatorTypes
- GetData

The structure leads to flexibility and extendibility, as new entries can be added without having to adjust the clients or the system. The usual workflow in the API is:

1. Request WaterPlants > determine desired **waterPlantId**

2. Request TreatmentStepTypes for the desired waterPlantId > determine desired **treatmentStepTypeId**

3. Request QualityIndicatorTypes for the desired treatmentStepTypeId > determine desired **qualityIndicatorTypeId**

4. Use all 3 IDs to request actual data through a dedicated endpoint

The data can be filtered according to the API description. General indicators can be requested by leaving out step 2 and directly requesting QualityIndicatorTypes on the waterplant.

---

[14] Where one database call is not enough to generate the requested datastructure due to dependencies or formats.

5. Experimental Results and Analysis

The following chapter presents the results and conclusions of the research, the design and the implementation of the system, as well as a proof for a working prototype.

5.1 Harmonization Strategies

Generally spoken, the more different formats the system accepts and harmonizes, the better. This is only true if the harmonization is complete and correct. Trying out different harmonization strategies for each of the formats has shown, that the more format-tolerant the system is, the more misinterpretations take place and therefore the more incorrect data gets store in the database.

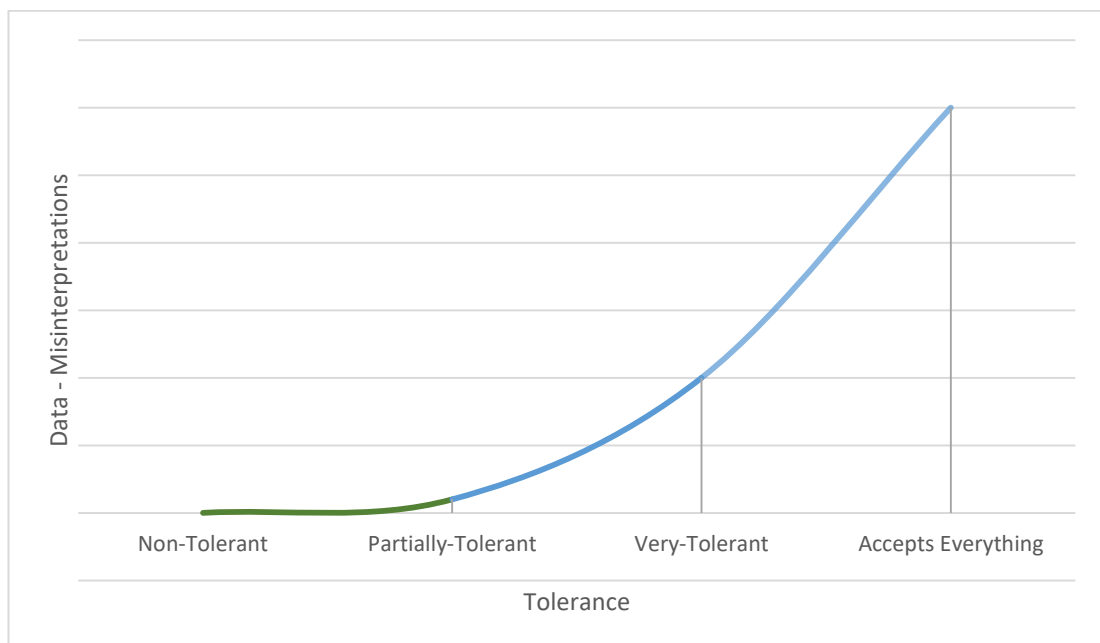The following diagram shows experiences won from the design and implementation phases of this system:

Figure 5-20: Service tolerance to data-misinterpretation diagram

The diagram shows 4 levels of a systems format-tolerance. The names are chosen by the author of this dissertation.

A **Non-Tolerant** system is a system, which only accepts one predefined **data-format**. This system (if implemented correctly) will not have any misinterpretations (wrongly assigned data, not recognized fields, misinterpreted values). This results in a robust, but very inflexible system.

A **Partially-Tolerant** system accepts multiple formats, but their **data-structure** has to be strongly defined and complied by the user. If the format entering the system does not match the structure definition, the data is dismissed. This leads to a more flexible system than a system using the first strategy, but is harder to implement and a little more prone to errors.

A **Very-Tolerant** system accepts data in a format which is similar enough to the defined format and its structure, to be interpreted by the system. This system more flexible than a Partially-Tolerant system but also leads to a lot more misinterpretations and errors and thus to inconsistencies in the stored data, which limits the usefulness of the system.

A system which accepts **everything**[15], and tries to interpret it the best possible way will require the most design and implementation time, but will also have the most inconsistent data.

In conclusion this leads to the trade-off between flexibility and data-consistency, but there is a save grade of flexibility which can be implemented into the system without the danger of making the data inconsistent, so the Partially-Tolerant system with its defined structure seems to be the best choice.

---

[15] Format and structure - wise

5.2 Proving the concept

The following chapter focuses on the implemented software, its description and end-to-end tests. The main objective of the dissertation was to create a software prototype of a system to gather historical and real-time data from waste water treatment plants, harmonize it and provide it for other systems and prove that this system, under certain circumstances, would work in the real world. The user feedback in the case of harmonization requests is given as reply with the error message, or with a 200 OK http code in case of a successful process.

5.2.1 Expandability

The system is extendable during runtime. This characteristics is granted due to the fact that the water-plants, treatment step types and water quality indicators are read from the database for every request. Extending the database dataset also means extending acceptable and harmonizable data.

To prove the system is expandable in the entire chain, the following JSON object will be harmonized:

Request:

```json
{
    "name":"TestWaterPlant",
    "treatmentSteps":[
        {
            "qualityIndicators":[
                {
                    "name":"TestQualityIndicator",
                    "timestamp":"2018-03-15T16:40:55",
                    "unit":"mg/L",
                    "value":123
                }
            ],
            "name":"TestTreatmentStepType"
        }
    ]
}
```

The output of the request-result, when the three names are not present in the database:

```
{
    "Message": "An error has occurred.",
    "ExceptionMessage": "Data could not be harmonized. Rea-
son: Could not retreive the wastewater treatment plant id for
the name [TestWaterPlant]",
    "ExceptionType": "System.Web.HttpException"
}
```

After adding "*TestWaterPlant*" to the WaterPlant table:

```
{…
    "ExceptionMessage": "Data could not be harmonized. Rea-
son: Could not retreive the treatment type id for the name
[TestTreatmentStepType]"…
}
```

After adding "*TestTreatmentStepType*" to TreatmentTypes table:

```
{…
    "ExceptionMessage": "Data could not be harmonized. Rea-
son: Could not retreive the indicatorTypeId for the indica-
torTypeName [TestQualityIndicator]"…
}
```

After adding "*TestQualityIndicator*" to QualityIndicatorTypes table:

```
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2018-03-15T16:40:55",
                    "unit": "mg/L",
                    "value": 123
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

This response represents the object added to the service storage. Note, that this is an entirely new treatment step and an entirely new quality indicator type and not mapping.

5.2.2 Mapping

Mapping is a big topic in the harmonization process. The system has to deal with aliases[16] on plant-basis as well as on global basis, meaning that aliases for all incoming data can be added, as well as aliases for data coming from specific plants.

As a follow up to the previous example, the following JSON object will be harmonized:

```json
{
    "name":"TestWaterPlant",
    "treatmentSteps":[
        {
            "qualityIndicators":[
                {
                    "name":"TestQualityIndicatorALIAS",
                    "timestamp":"2018-03-15T16:40:55",
                    "unit":"mg/L",
                    "value":123
                }
            ],
            "name":"TestTreatmentStepType"
        }
    ]
}
```

Request result:

```
{…
    "ExceptionMessage": "Data could not be harmonized. Rea-
son: Could not retreive the indicatorTypeId for the indica-
torTypeName [TestQualityIndicatorALIAS]"…
}
```

---

[16] A name defining the exactly same thing as a different name

After adding the entry "*TestQualityIndicatorALIAS*" to the QualityIndica-torMapping table, and referencing the *TestQualityIndicator* type:

```
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2018-03-15T16:40:55",
                    "unit": "mg/L",
                    "value": 123
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

Note that this time the name of the quality indicator changed – this means "*TestQualityIndicator*" and "*TestQualityIndicatorALIAS*" will both be stored referencing the same quality indicator type.

5.2.3 Format Tolerance

The system defines 3 data formats as valid formats. Those formats are CSV, XLS/XLSX and JSON. CSV and XLS/XLSX are handled in a different way than JSON, due the differences in their structure. The following example shows how different formats get harmonized to create a comparable output.

5.2.3.1 XLS/XLSX / CSV

First an XLS file will be harmonized. The harmonization URI containing wa-terPlantId and treatmentStepTypeId is used for that matter.

The example table looks following:

Figure 5-21: *Example data table*

```
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "Nitrogen Dioxide",
                    "timestamp": "2016-11-01T00:00:00",
                    "unit": "mg/L",
                    "value": 1.19213
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

As the result implies, the name NO2 got mapped to the quality indicator type "*Nitrogen Dioxide*". Also the screenshot rounded the value, while the service stored the exact value in the database. Empty and dummy columns got cleaned or ignored.

The next step is to harmonize a table with multiple different indicators:

Figure 5-22: *Example data table – multiple indicators*

The names of the columns are purposely called "*1st_NAME*" and "*NAME2* " to display, that mapping isn't done based on title alone, and the value columns are found anyway:

75

```
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "Nitrogen Dioxide",
                    "timestamp": "2016-11-01T00:00:00",
                    "unit": "mg/L",
                    "value": 1.19213
                },
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2016-11-01T00:00:00",
                    "unit": "mV",
                    "value": 1.337
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

The values are stored accordingly to the request. Next, a vertical table will be

harmonized:

DATE	1.11.16 3:45 PM	1.11.16 3:50 PM
1st_NAME	NO2	
VALUE	1,30	1,40
NAME2	TestQualityIndicator	
Value	1,22	1,20

Figure 5-23: Example data table – vertical orientation

In this case, the values of an identifier are placed horizontally in the table instead of one above the other. Following is the outcome:

```
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "Nitrogen Dioxide",
                    "timestamp": "2016-11-01T15:45:00",
                    "unit": "mg/L",
                    "value": 1.3
                },
                {
                    "name": "Nitrogen Dioxide",
                    "timestamp": "2016-11-01T15:50:00",
                    "unit": "mg/L",
                    "value": 1.4
                },
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2016-11-01T15:45:00",
                    "unit": "mV",
                    "value": 1.22
                },
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2016-11-01T15:50:00",
                    "unit": "mV",
                    "value": 1.2
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

All values are mapped correctly to their indicator types, despite the different orientation of the source table.

5.2.3.2 JSON

Following examples show different JSON formats being harmonized. At first

a request is sent using the harmonization URI containing a waterPlantId.

Request:

```json
{
    "qualityIndicators":[
        {
            "name":"TestQualityIndicatorALIAS",
            "timestamp":"2018-03-15T16:40:55",
            "unit":"mg/L",
            "value":123
        }
    ],
    "name":"TestTreatmentStepType"
}
```

The step does not contain any information about waterplants. Additionally,

there is an alias for one of the quality indicators. The outcome is following:

```json
Status: 200 OK
{
    "name": "TestWaterPlant",
    "treatmentSteps": [
        {
            "qualityIndicators": [
                {
                    "name": "TestQualityIndicator",
                    "timestamp": "2018-03-15T16:40:55",
                    "unit": "mg/L",
                    "value": 123
                }
            ],
            "name": "TestTreatmentStepType"
        }
    ]
}
```

The system recognized that only treatmentsteps were passed, and mapped

the quality indicator alias to the correct name.

The next request contains only a quality indicator without any information on the step type or waterplant within the body. The identifiers of both are present in the URI. Request:

```json
{
    "name": "TestQualityIndicator",
    "timestamp": "2018-03-15T16:40:30",
    "unit": "mg/L",
    "value": 123
}
```

Corresponding answer:

```json
Status: 200 OK
{
   "name": "TestWaterPlant",
   "treatmentSteps": [
      {
         "qualityIndicators": [
            {
               "name": "TestQualityIndicator",
               "timestamp": "2018-03-15T16:40:30",
               "unit": "mg/L",
               "value": 123
            }
         ],
         "name": "TestTreatmentStepType"
      }
   ]
}
```

The quality indicator got mapped into the right plant and right step.

5.2.4 Robustness & Feedback

Robustness and user feedback are described in the same chapter for a simple reason – if the harmonization was successful and the data got stored in the system, the only feedback the user requires is the http code 200 OK. Returning the created object, like in the previous chapters, is good for control, especially in the development phase, but not necessary – since no further user actions are required. On the other side, it is very important to give the

user feedback on why the harmonization of his data failed, so he can "fix" his data structure to comply the system. The system must be able at any time to deal with "wrong" data, without having impact on the stability.

The following examples show how the system reacts to data, which it cannot harmonize:

DATE	NO2	Value	TestQualityIndicator
1.11.16 3:40 PM		1,19	
1.11.16 3:45 PM		1,19	

Figure 5-24: Example of a table with two valid indicator names for one value column

The figure shows a table with two valid indicator names (NO2 and TestQualityIndicator) but only one value column. The chance of misinterpretation is high, as without knowledge on how to interpret the table both indicators are valid options. Following is the output of a harmonization attempt:

```
{…
    "ExceptionMessage": "Data could not be harmonized. Rea-
son: The number of numeric columns doesn't match the number
of indicator names"…
}
```

The exception message is modified into a meaningful user message – implying, that there is either a column with values missing, or too many specified quality indicator names.

Some more examples of errors:

```
{…
   "ExceptionMessage": "Data could not be harmonized. Reason: The
passed object didnt have a supported format"...
}
```

```
{…
   "ExceptionMessage": "Data could not be harmonized. Reason: Could not
retreive the wastewater treatment plant name for the id [10]",…
}
```

{…
    "**ExceptionMessage**": "Data could not be harmonized. Reason: This file-Format is not supported on this endpoint. Please use: [base-uri]/harmonize/fileFormat/{fileFormat}/waterPlant/{waterPlant}/treatmentStep Type/{treatmentStepType}",…
}

{…
    "**ExceptionMessage**": "Data could not be harmonized. Reason: no datetime was found",…
}

The messages are meaningful to the point, that the person looking at it, can imagine what went wrong without looking at the request itself.
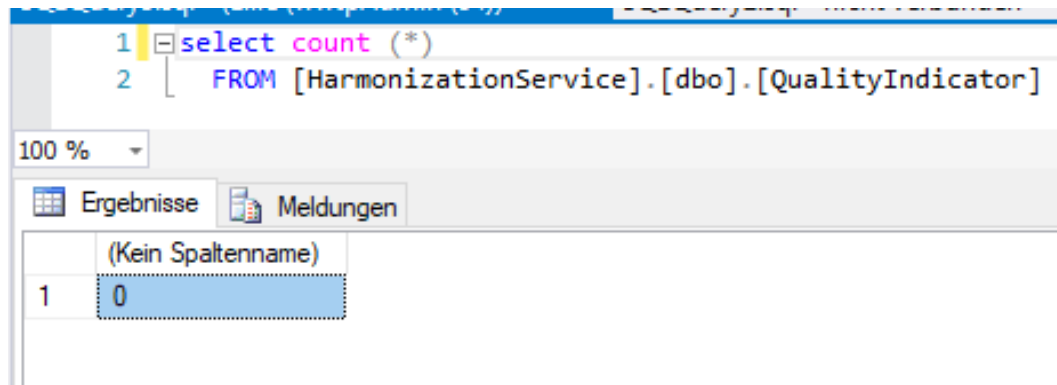
5.2.5 Big amount of data

Historical data will usually come in bigger chunks. For that reason, the system must be able to harmonize files with a higher amount of data. The following examples show, how the system handles such files. The database is cleaned of indicators previous to every test. Each of the tables will contain only the indicator name, its value and the timestamp, as well as a title row.

1. Table with 10.000 entries

9999	NO2	5,521	05.12.2016 17:05:00
10000	NO2	5,532	05.12.2016 17:10:00
10001	NO2	6,532	06.12.2016 17:15:00
10002			

Figure 5-25: End of a table with 10000 entries containing indicator name, value and timestamp
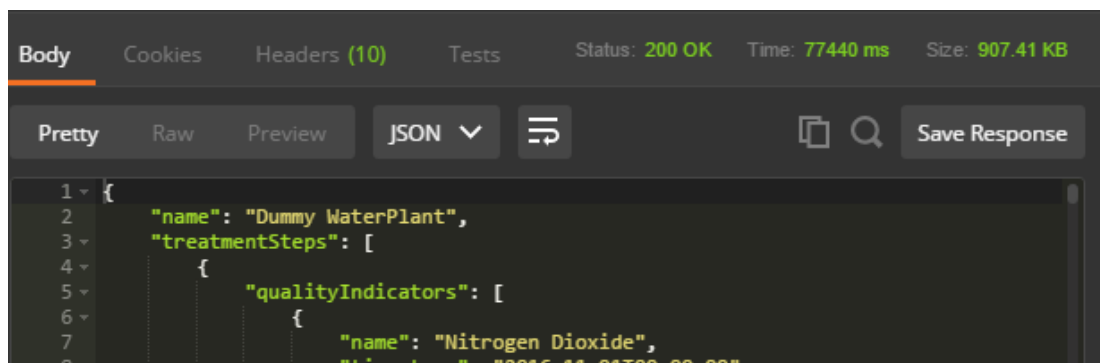
Figure 5-26: The amount of entries in the QualityIndicator table before the test

Result:

The below picture shows the time it took to harmonize the dataset and its size and that the request was successful.
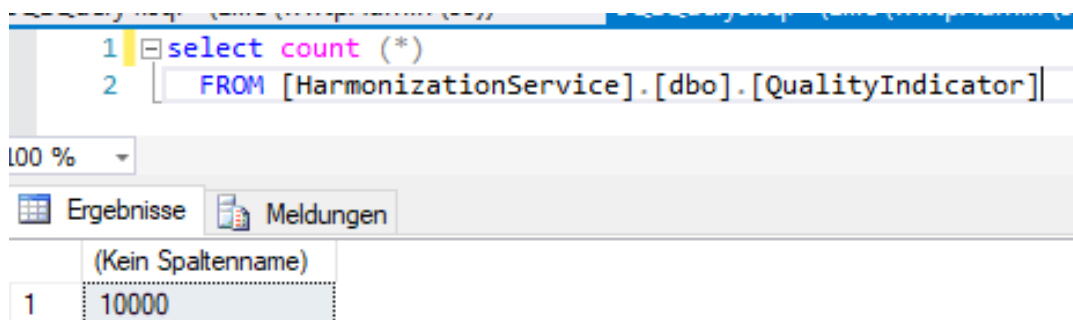
Figure 5-27: Request result (10.000 entries)

The below figure shows that all the entries are actually stored in the systems previously empty database.

Figure 5-28: Database after harmonization request
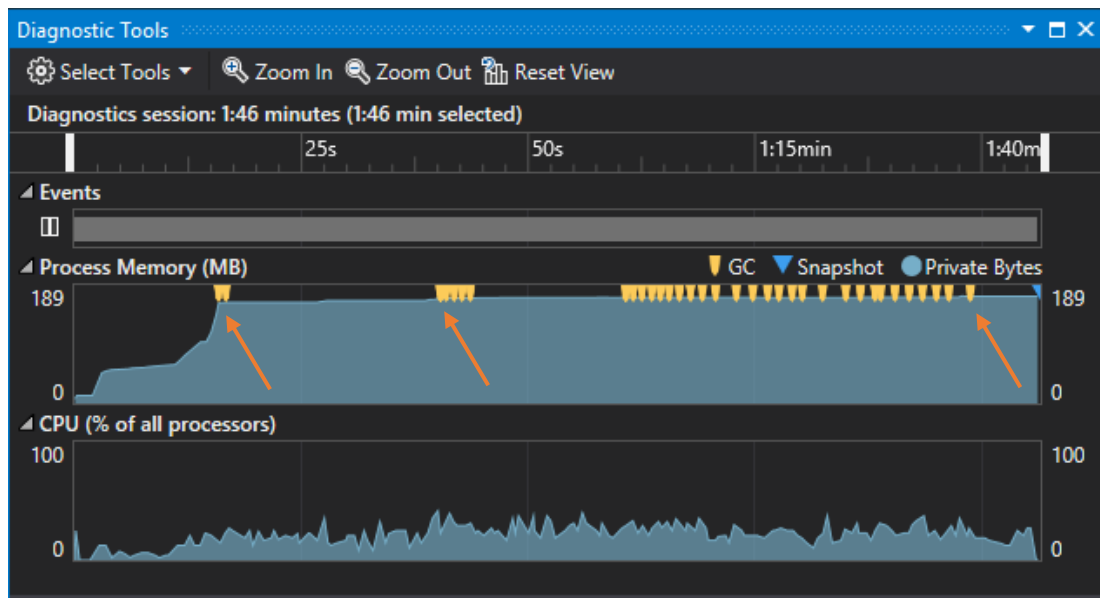
Figure 5-29: *Visual Studio 2017 performance profiler*

The CPU usage during the process does not go over 50% at any point. The memory has its first peak after the program is done loading dependencies, starting the background pull service and starting the web listeners. The second arrow shows the start of the harmonization process. It can be seen, that the memory usage barely goes up during that process until the end of harmonization at the 3rd arrow.

2. Table with 100.000 entries

The following picture shows the result of the harmonization:
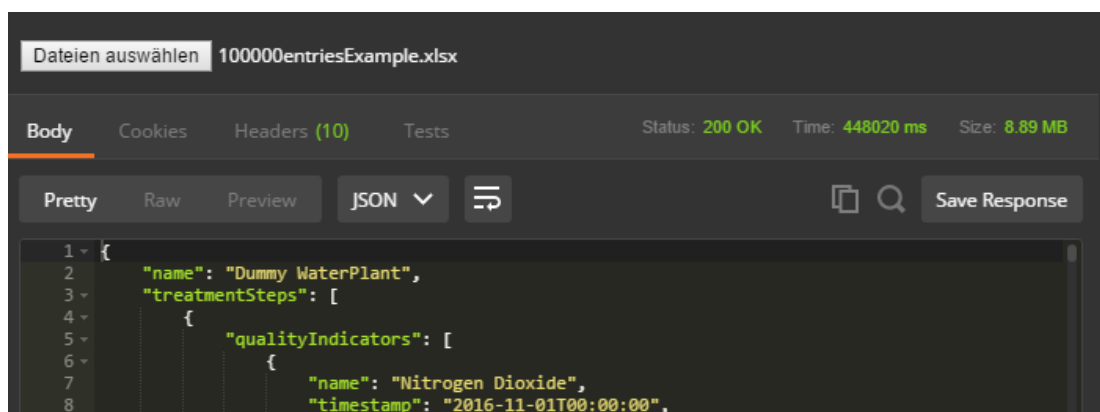
Figure 5-30: *Harmonization result of 100000 entries-file*

Multiplying the amount of data by 10 had the following impact on the performance:
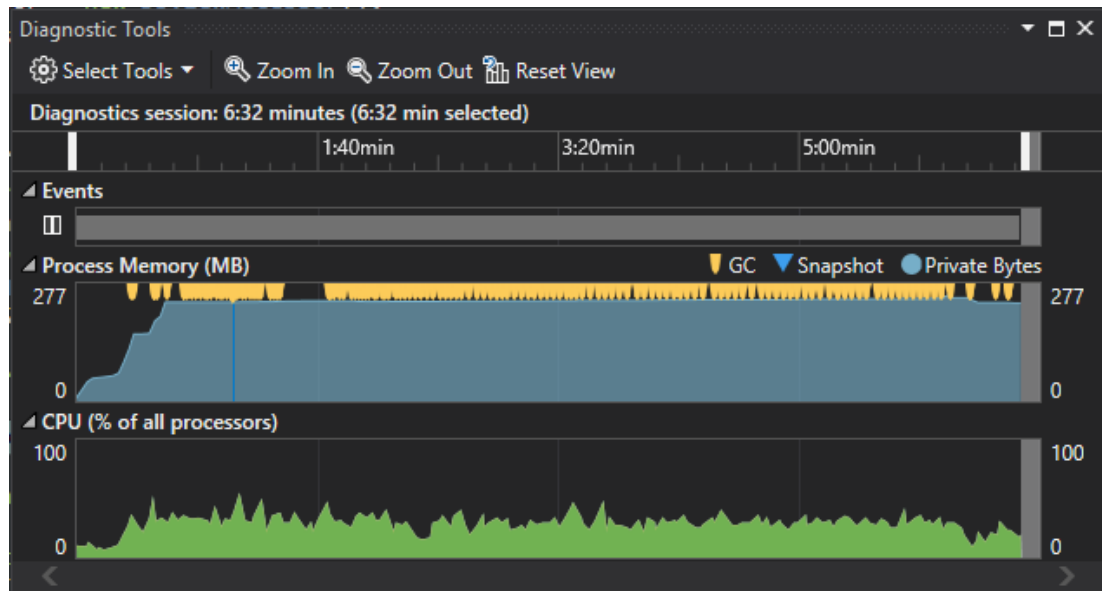
Figure 5-31: *Visual Studio 2017 performance profiler – 100000 entries*

The memory usage went up significantly to around 250mb. CPU usage does pick at around 50%

3. Table with 1.000.000 entries

Last includes a file containing 1.000.000 entries.

Result:

The request was timed out, so there is no postman result for this case. The harmonization process required around 140min to finish.
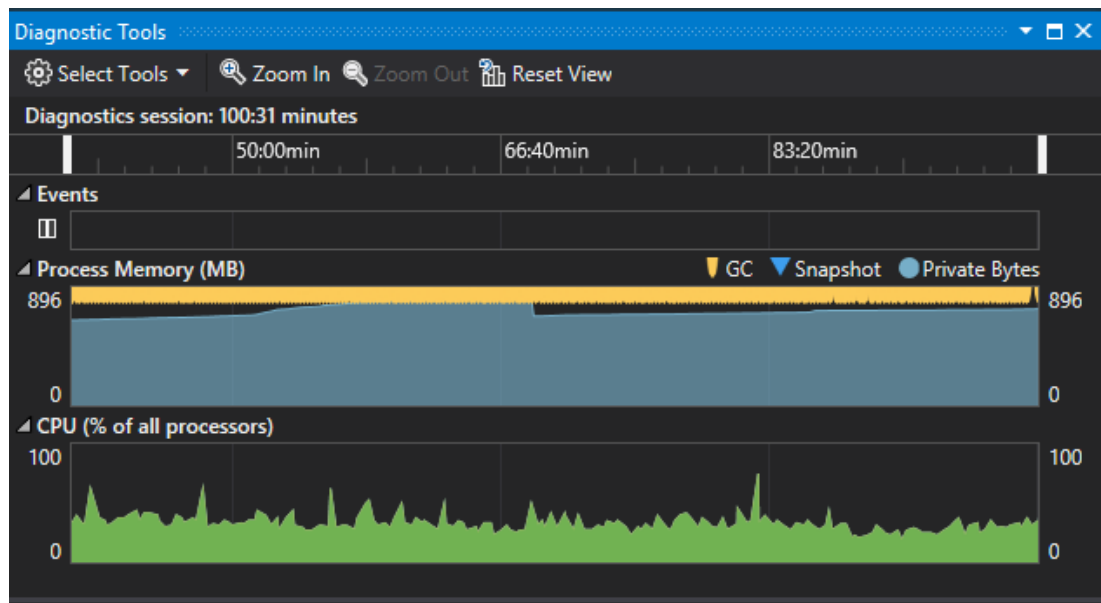
Figure 5-32: *Visual Studio 2017 performance profiler – 1000000 entries*

The ram requirement peaked around 900mb.

Conclusion:

The system was able to harmonize all test files. The times required for an entire harmonization in minutes:

Entries	Time
10.000	1,29min
100.000	7,46min
1.000.000	140min

Table 5-9: *Measured harmonization process times*

Based on the time it takes to harmonize 10.000 entries and the factor 10, the following diagram shows the expected time it should take to harmonize 100.000 and 1.000.000 entries:

Figure 5-33: *Diagram: Expected time vs actual result – time to harmonize*

The time it takes to harmonize a dataset with 100.000 entries is a lot lower than the calculated expectation. This is mainly due to the fact, that for 10.000 entries, the initialization of the mapping list, the blacklist and other events which take place only once during a harmonization plays a significant role. The more entries there are, the less significant this time becomes – as can be seen in the 100.000 entries set. The set with 1.000.000 entries takes longer than expected, even though the previous set actually was a lot faster. This is because of the fact that the system checks if the value is already stored in the system. The check requires to know if there is another entry for a specific treatment step type, on a specific water plant for a specific quality indicator for a specific datetime. If there is, the entry is dismissed. The more entries there are in the system, which match many of those conditions, the longer it takes to perform this check.

The required RAM is going up significantly for bigger files. The data is stored in the memory, because it is processed a lot. Storing it on the hard drive

would make the time it takes to harmonize a dataset (especially the bigger ones) significantly higher.

5.2.6 Providing harmonized data

Based on the data acquired during previous tests, the system is now capable of providing data to the users. The test will follow the defined generic call chain containing a request for available:

- Water plants
- Treatment step types
- Quality indicator types
- Data

The following picture shows an example call to the web-service running local-ly:
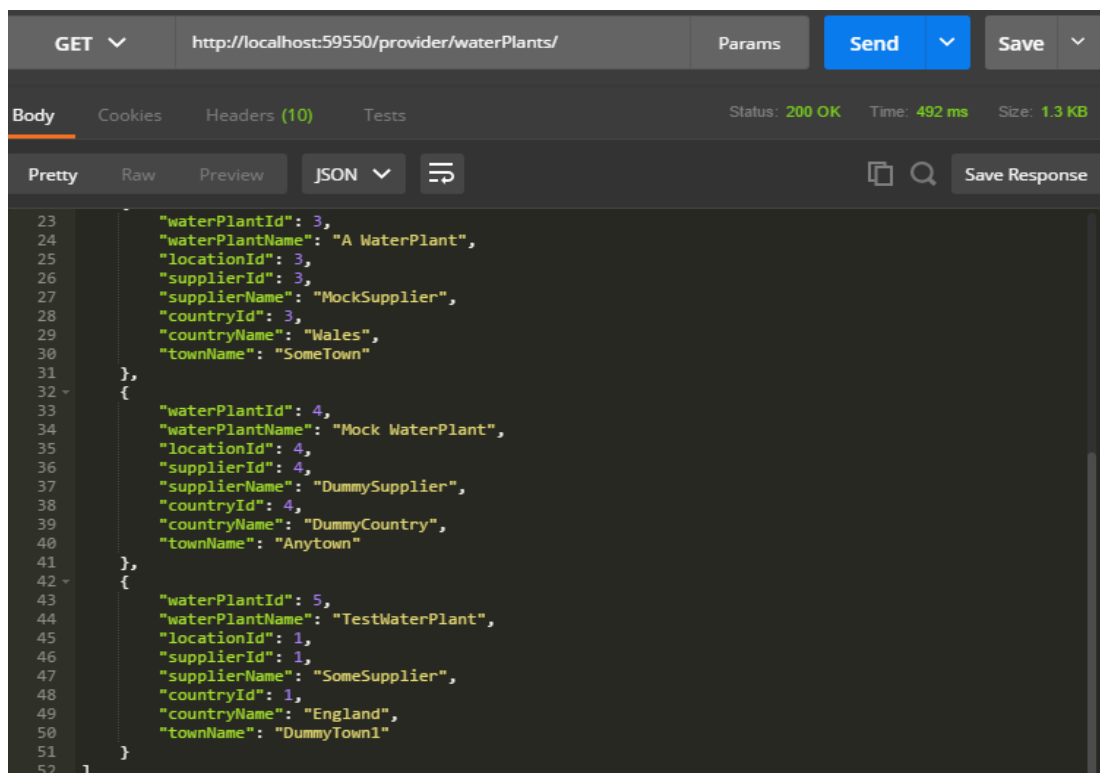
Figure 5-34*: Example postman call towards locally running web-service*

It can be seen, that the response contains a waterPlantId followed by a lot of metadata, which is not required to retrieve specific quality indicator data. Thanks to the relations in the database, this call requires only 492ms, as the bigger tables are only referencing the smaller ones instead of containing their values.

The next call goes towards a specific water plant with the id 5:
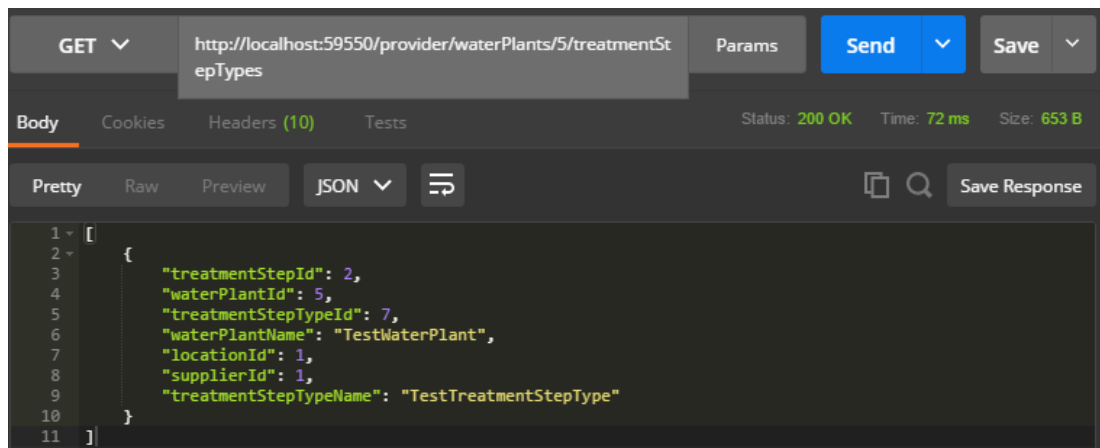
Figure 5-35: Example postman call for treatment step types on a water plant

The call takes only 72ms. It provides all treatment step types, which contain values on a specific water plant. Having the id of the water plant and the id of the treatment step type, the available quality indicator types can be requested:
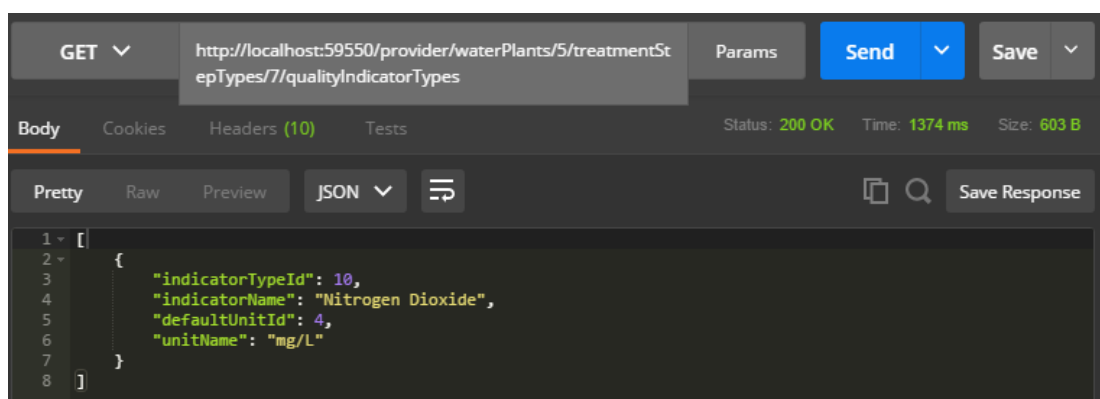
Figure 5-36: Example postman call for quality indicator types of a treatment step

The request of actual values is the longest and most expensive call, and thus can be filtered by date:
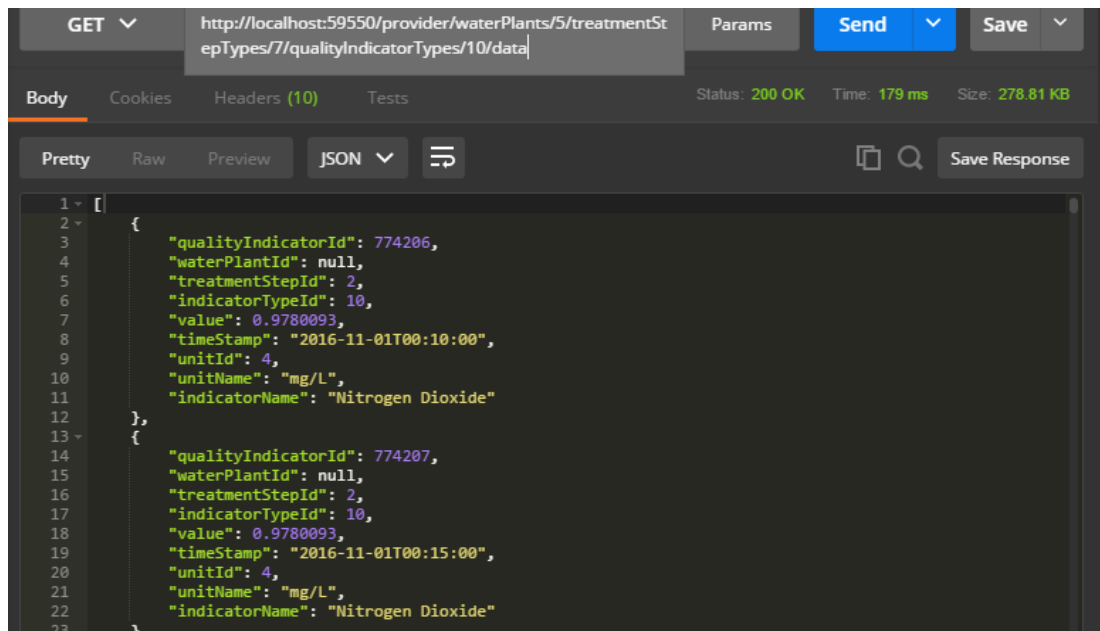
Figure 5-37*: Example postman call for quality indicators of a treatment step*

The user cannot tell if the data has its origin in a CSV, a XLS or a JSON file. It is coming out in a defined format which can be easily used for further processing and analyses due to the knowledge of its format. For the matter of format and available endpoints, the service describes itself with the help of a swagger file to the outside world. The swagger description can be seen by adding a /swagger/docs/[version number] to the end of the base URI:

Figure 5-38*: Swagger file of the harmonization system*

The swagger file contains not only the formats of accepted files and expected result types, but also the description of all available endpoints.

6. Summary and Further Work

The following chapter summarizes the work done in this dissertation, the won knowledge of the project development and the problematics which occurred during the design and implementation phase. Furthermore it describes what is still to be done until this system can work in the real world, what the restrictions and problematics are and what it takes to solve them.

Investigations on the functionality of waste-water treatment-plants have shown, how their specific components relate to each other and what the constants between various water-plants are. Since the waste-water industry is developing, the designed common schema was required to be extendable and generic, yet still specific enough to provide comparable data. Researches upon comparable project have shown the importance of a tolerant, yet consistent solution. The more companies can join a harmonization platform, the more use it provides to the stakeholders, due to the variety of data and data sources. The research on this fact has led to the insight, that the most historical data is stored in a table structured format and the need to implement two different harmonization strategies – one for the "desired" tree-based format and one for the commonly used table format. Knowledge won from the UNECE Project [14] was especially meaningful for the design of the system, since it provided insight on the problems which occurred during the process of the creation of their Single Window Harmonization System and where the focus should be put in order to create a convincing, long-terms solution. Even though subjects like security, availability and load balancing were only marginal in the scope of this project, the conception of the system as a cloud web-service enables the future development to add those features

without big overhead. The research upon harmonization methods has led to the pipeline architecture of the harmonization service. The advantages are not only the expandability by additional steps, but also the clean structure and separation of the main harmonization steps which are: simplification, harmonization and standardization. Analyses of the system have shown, that even with 1.000.000 entries in the memory, the system is still within limits of the modern systems capacity, with around 900mb of ram in use, in the worst case of an XLSX file with large data-overhead.

6.1 Time-Plan comparison

The following chart shows the actual time, which was required to fulfil the tasks. Grey bars have the same length as was initially planned, green bars represent tasks which were done quicker than expected and the red marked bars represent tasks, which took longer. It can be seen, that the designing phase took longer than expected. The main reason for this, was the design of a data schema, which had to be done with the consideration of the results from the *Water-Plant-Analysis*, the *Analysis of similar Projects* and the plan, to design a dedicated database schema. Due to this delay, the lower priori-tised *Simulation* was deleted from the project. During the implementation phase, the harmonization system required a lot more time than expected, which resulted in using up almost all of the buffer time at the end of the pro-ject. The cancelled simulation would definitely have required too much effort at this point.

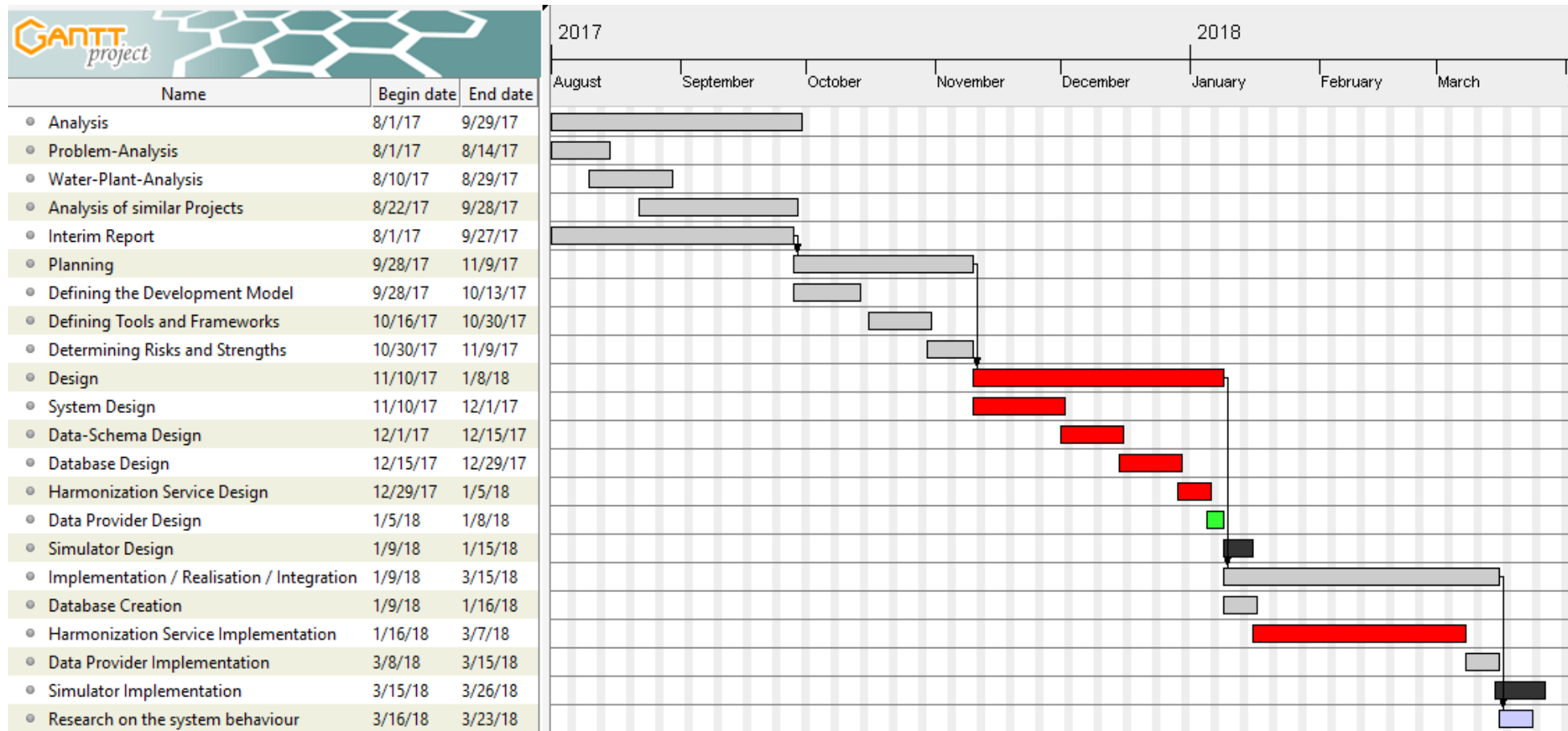| Name	Begin date	End date
Analysis	8/1/17	9/29/17
Problem-Analysis	8/1/17	8/14/17
Water-Plant-Analysis	8/10/17	8/29/17
Analysis of similar Projects	8/22/17	9/28/17
Interim Report	8/1/17	9/27/17
Planning	9/28/17	11/9/17
Defining the Development Model	9/28/17	10/13/17
Defining Tools and Frameworks	10/16/17	10/30/17
Determining Risks and Strengths	10/30/17	11/9/17
Design	11/10/17	1/8/18
System Design	11/10/17	12/1/17
Data-Schema Design	12/1/17	12/15/17
Database Design	12/15/17	12/29/17
Harmonization Service Design	12/29/17	1/5/18
Data Provider Design	1/5/18	1/8/18
Simulator Design	1/9/18	1/15/18
Implementation / Realisation / Integration	1/9/18	3/15/18
Database Creation	1/9/18	1/16/18
Harmonization Service Implementation	1/16/18	3/7/18
Data Provider Implementation	3/8/18	3/15/18
Simulator Implementation	3/15/18	3/26/18
Research on the system behaviour	3/16/18	3/23/18

Figure 6-40: *Final Gantt Chart*

6.2 Problems and trade-offs

During the planning and implementation phase of the project, many problems occurred and at some points decision had to be made, which not only provided advantages for the system. The following chapter summarizes the problems faced during the dissertation.

The objective of the dissertation was not to build a system which harmonizes data from waste water treatment plants in the United Kingdom, but only to build a system which would be capable of doing so. Due to this fact, it may seem like missing access to real water-treatment plants should not be a problem. In reality the problem was not the missing data, but the missing information on the formats and data structures the data gets stored in. Available data was provided by water-treatment plants from Italy and only included two different structures, which then led to the structure definition, which the harmonization service is using. Not only the internal structure of files, but also the data types were needed to create a useful system. For that matter, the system was created with the most common data storage formats, also including formats in which the example data was provided. The missing information on file formats has led to a system design, which accepts multiple different file formats. The formats were either tree or table based and could not be processed equally. This has led to a split in the harmonization process which required the user to pass the data format of his provided data. Up until the point of the final simplification, both data types required a different processing strategy. This also means, that every other base format added to the system requires a moderate amount of new code.

The next problem focuses on the usability. The created system was not implemented with the focus on ease of administration, but functionality. This

has led to several processes being only able to be performed on the data-base directly at the present time. This includes things like:

- Adding new water plants
- Adding new treatment step types
- Adding new quality indicator types
- Adding new locations
- Adding new units
- Adding new sources for pulling of data
- Cleaning and modifying data in the database

A general problem was the time to finish the dissertation. Even though the project was divided in tasks and every task was planned, the implementation and integration on the amazon cloud took longer than expected. The work with dynamic types has led to many implementation mistakes and runtime errors. This required some lower priority tasks to be dismissed or only partial-ly done. There was no simulator implemented. Instead of a simulator, the program "postman" was used to send requests to the system. Due to the fact, that the pulling is done from a specific path and the expected data is in file-form, there was no need for the implementation of a water-plant simulator for the testing of data-pulling.

6.3 Further work

There are two categories of work which should still be done to make this sys-tem work in the real world. The first category includes improvements and modifications on the system, which can be done at the present time and the second category includes work that needs to be done by the industry / water

plant administration and adjustments on the system after those changes took place.

The system is built around waste-water treatment-plants. It fully supports information coming from any of the stages of treatment-plants, but does not contain data regarding information like weather or governmental regulations, which are of importance for the context of the stored, water-related data. Additional endpoints for the system could be built to store and retrieve such data, so the "single window" system becomes even more self-contained.

The current system lacks any kind of access permission and gives out redundant data which is useless to the common user. This includes data like database identifiers irrelevant for user requests. Every user can add data to any available water-plant. This security issue requires some kind of user-management in the system, allowing specific users to insert data for specific water-plants. Additionally the system requires some sort of logging, so the system administrator can also monitor the processes on the system. Even though different requests are handled in parallel on the system, all data within a request is processed successively. The code requires some refactoring to optimize the CPU usage. Additionally, the already mentioned additional endpoints or user interface for management of the database would be highly recommended.

To integrate the system into the real world, first of all, it would require the waste-water treatment plant administrators to provide their water quality information. The format of the information would have to be one of the formats supported by the system. Additionally, to efficiently use the data, the system requires a 3rd party system, which uses the endpoints of this system to evaluate and analyse the stored data.

6.4 Conclusion

The prototype of the system was finished in time. It is based on research towards harmonization, combined with a data schema based on waste water plant functionality. The concept is proven with example data from Italian water-plants, as well as self-created data. It is clarified, what needs to be fulfilled to make the system usable in the real world, with real waste water treatment plants. The implementation of the system took longer than it was estimated during the planning phase. The reason behind the delay was mostly the work with dynamic types, as well as problems with amazon cloud integration.

The current solution can be used to harmonize data of commonly used formats. It can be deployed on amazon cloud or on a local computer. It provides user feedback, as well as maintainable code. It could become the first step towards an infrastructure capable of monitoring the water quality at a national level.

References

[1] Water UK, "Water UK," Water UK, 2018. [Online]. Available: http://www.water.org.uk/about-water-uk/wastewater. [Accessed 18 09 2017].

[2] Department for Environment, Food and Rural Affairs - DEFRA, "Sewage Treatment in the UK," Crown, March 2002.

[3] European Environment Agency, "Urban Waste water treatment," European Environment Agency, 15 December 2017. [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/urban-waste-water-treatment/urban-waste-water-treatment-assessment-4. [Accessed 10 02 2018].

[4] American Society of Civil Engineers, "Failure to Act - The economic impact of current investment trends in water and wastewater treatment infrastructure," American Society of Civil Engineers., Boston,, 2011.

[5] Water UK, "WATER UK," Water UK, 2018. [Online]. Available: https://www.water.org.uk/consumers/find-your-supplier. [Accessed 28 09 2017].

[6] B. McAlinden, "What are water transfers and interconnections?," United Kingdom, 12 November 2015.

[7] E. &. D. A. Mousavi, "WWTP-Global-300617," United Kingdom, 2017.

[8] B. N. &. M. WD, "Top-Down Design," August 2001.

[9] Amazon Web Services Inc., "Overview of Amazon Web Services," April 2017.

[10] Unitet States Environmental Protection Agency, "How Wastewater Treatment Works... the Basics," Office of Water (4204), Washington, DC 20460-0001, 1998.

[11] P. V. a. D. Lee, "On-line monitoring equipment for wastewater treatment process: state of the art," IWA Publishing, 2003.

[12] S. Lim, "Introduction to Data Harmonization and Modelling," ESCAP, UNNExT, 19-28 April 2017.

[13] A. Guess, "In-Depth Interview: Five Steps to Data Harmonization with Absolutdata CEO Anil Kaul," May 2017.

[14] M. Apostolov, "The Single Window and Data Harmonization in Line with the International Standards," United Nations Economic Comission for Europe - UNECE, 8-9 September 2008.

[15] UNNExT, ESCAP, UNECE & WCO, "Data Harmonization and Modelling Guide," United Nations publication, Thailand, 2012.

[16] Who Framework Convention on Tobacco Control, "Standardization and harmonization of data and data collection initiatives," FCTC, Punta del Este, Uruguay, 15 August 2010.

[18] Energy Star, "ENERGY STAR Score for Wastewater Treatment Plants in the United States," ENERGY STAR PortfolioManager, November 2014.

Appendix A

JSON Schema

The following presents the JSON schema defined in the chapter 4.2, including the

parameter properties:

```
"title": "WaterTreatmentPlant",
"type": "object",
"properties": {
  "name": {
    "required": true,
    "type": "string"
  },
  "supplier": {
    "type": "string"
  },

  "location": {
    "type": [
      "string",
      "null"
    ]
  },
  "treatmentSteps": {
    "type": [
      "array",
      "null"
    ],
    "items": {
      "type": [
        "object",
        "null"
      ],
      "properties": {
        "qualityIndicators": {
          "required": true,
          "type": "array",
          "items": {
            "type": [
              "object",
              "null"
            ],
            "properties": {
              "name": {
                "required": true,
                "type": "string"
              },
              "timestamp": {
                "required": true,
```

```
                          "type": "string"
                },
                "unit": {
                  "type": [
                    "string",
                    "null"
                  ]
                },
                "value": {
                  "required": true,
                  "type": "number"
                }
              }
            }
          },
          "name": {
            "required": true,
            "type": "string"
          }
        }
      }
    },
    "generalIndicators": {
      "type": [
        "array",
        "null"
      ],
      "items": {
        "type": [
          "object",
          "null"
        ],
        "properties": {
          "name": {
            "required": true,
            "type": "string"
          },
          "timestamp": {
            "required": true,
            "type": "string"
          },
          "unit": {
            "type": [
              "string",
              "null"
            ]
          },
          "value": {
            "required": true,
            "type": "number"
          }
        }
      }
    }
  }
}
```

Appendix B

API Definition - DataProvider

GetWaterPlants

Uri: {*baseUri*}/waterPlant

Method: GET

Body: empty

Returns: List<WaterPlant>

GetTreatmentStepTypes

Uri: {*baseUri*}/waterPlant/{*waterPlantId*}/treatmentStepType

Method: GET

Body: empty

Returns: List<TreatmentStepTypes>

GetQualityIndicatorTypes

Uri:

For step-based indicators:

{baseUri}/waterPlant/{*waterPlantId*}/treatmentStep/{*treatmentStepTypeId*}/indicator

For general indicators:

{baseUri}/waterPlant/{*waterPlantId*}/indicator

Method: GET

Body: empty

Returns: List<QualityIndicatorTypes>

GetData

Uri: {*GetQualityIndicatorsBaseUri*}/{indicatorId}

Method: GET

Optional Uri Parameters: startDateTime : DateTime, endDateTime : DateTime

Body: `empty`

Returns: List<QualityIndicators>

Appendix C

API Definition - DocumentListener

<u>SetData</u>

POST	/harmonizer/harmonize/fileFormat/{fileFormat}/waterPlant/{waterPlant}/treatmentStepType/{treatmentStepType}

Parameters

Parameter	Value	Description	Parameter Type	Data Type
fileFormat	XLS ▾		path	string
waterPlant	(required)		path	string
treatmentStepType	(required)		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		

Try it out!

POST	/harmonizer/harmonize/fileFormat/{fileFormat}/waterPlant/{waterPlant}

Parameters

Parameter	Value	Description	Parameter Type	Data Type
fileFormat	XLS ▾		path	string
waterPlant	(required)		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		

Try it out!

POST	/harmonizer/harmonize/fileFormat/{fileFormat}

Parameters

Parameter	Value	Description	Parameter Type	Data Type
fileFormat	XLS ▾		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		

Try it out!

Description: This endpoint expects a call with the file format within the URI. A file format might either be XLS/XLSX, CSV or JSON. It is required for easier handling in the further process. XLS/XLSX and CSV files may only be send through the first API

endpoint, requiring the user to define the waterPlantId and the treatmentStepTypeId. JSON formatted data can be sent through any of the three endpoints, but if one of the parameters is missing (waterPlantId or treatmentStepTypeId), the service expects it to be defined within the provided data. The payload is expected as binary data in the provided format.

Interim Report

EE5500

Name: -

Student number: 1644612

Electronic and Computer Engineering

School of Engineering and Design

Dr. Alireza Mousavi

Sonntag, März 25, 2018

Table of content

Introduction

Acquisition, analysis and modelling of historical and real-time water-plant data.

Overcoming some of the existing barriers of interoperability and harmonization of data and information.

Access to clean water is the most basic and fundamental type of the human infrastructure. The quality of life highly depends on the accessibility to clean water. We require water not only for drinking, but also for cooking, and washing. Additionally, various professions and commercial establishments, like farmers or restaurants, could not exist without certain quality and quantity of water. The quantity of clean water in most cases, depends on collecting water and sewage from rivers and lakes, cleaning it in dedicated water-plants and thus bringing it to a specific quality standard, and then distributing it back into the waters.

A software groundwork for acquisition, analysis and modelling of historical and real-time data of water-plants will be the main topic of this master thesis. The Project will be done in partner work, although the tasks will be strictly separated and the outcome of one part of the project won't affect the outcome of the other part. This dissertation is dealing with the problem of acquiring, harmonizing and providing water-related data and leaves the analysis and presentation to the partner project. [1]

The specific cleaning process in the United Kingdom consists of 7 steps, which will be described later in the dissertation document [2]:
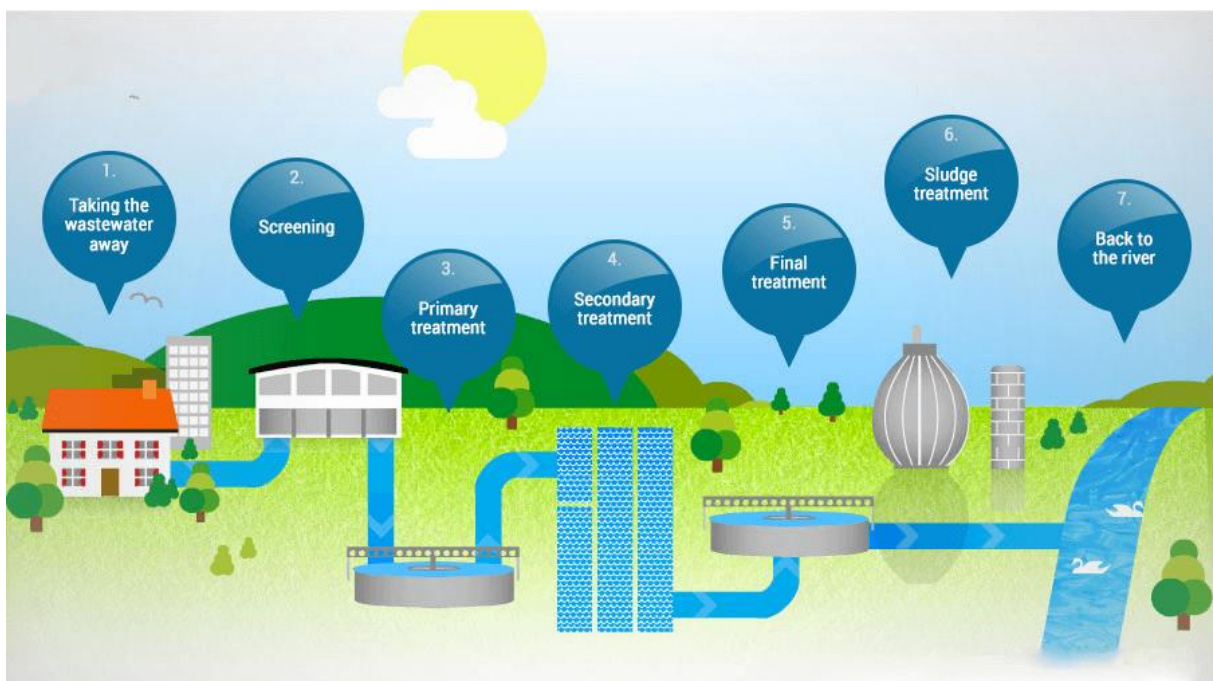
(Source: http://www.water.org.uk/about-water-uk/wastewater 18.09.2017 last accessed: 28.09.2017)

The most individuals will be interested in the outcome of step 7, which also indicates the quality of water available for public usage, nonetheless the incoming and outgoing water of the other steps

provides different kind of data which might be interesting for different kind of reports, especially due to the fact, that each step deals with a specific problem, meaning that all possible to gather data will also be gathered harmonized and stored by our system, for further investigation.

Background to the project

On average it costs a UK customer 1 pound a day to drink high quality water. This money goes into the water and wastewater treatment of around 16 billion liters of wastewater, gathered in around 345.000km of sewers, in around 9000 wastewater plants – every day. [3] [4]

The water supply regulations, set by the government, regulate the water treatment process of every water provider whose area is wholly or partially in the United Kingdom. The list of indicator parameters is long and contains minimum, maximum values and ranges within which values are allowed to lie. Only if all regulations apply the water may be called drinking water. With all the regulations and monitoring organizations the quality of UKs water might seem assured – yet the process of doing so is very troublesome and laborious. Twelve big companies, responsible for water and sewerage, cover most of UKs water supply. Additionally, there are some water-only companies providing water for some of the remaining regions. [4] [5]
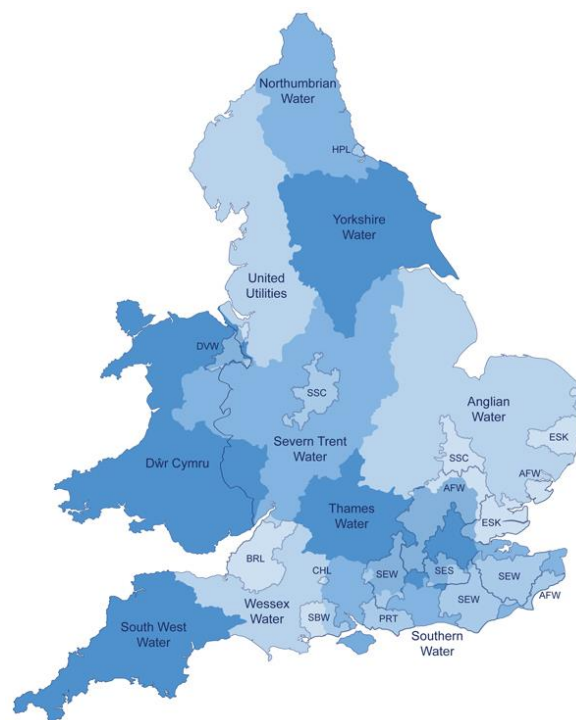
(Source: http://www.ofwat.gov.uk/households/your-water-company/map/ last accessed: 28.09.2017)

The water quality is regulated UK-wide, yet the way the different companies ensure their quality and monitor their water treatment process is not unified. This makes comparison between companies, as well as getting a global picture difficult. Reacting to lack of quality water in specific regions, or fore-

casting such a scenario, while still monitoring which of the remaining regions has enough "spare" quality water to help out the company in need would be a lot easier with a common information base. It would simplify the monitoring of local area changes caused by changes in the water and wastewater treatment regulations. To assure better forecasts or more meaningful reports, other information bases, like weather information might be taken into account – but those external systems are not a topic in this part of the (data-gathering) system.

The advantages of a big dataset from various sources are obvious – especially in a case where the geographical location of sources also mattering. Co-operating, comparing, planning, monitoring and analyzing is a lot easier when all the data is stored at seemingly one place in a unified format. Attempts to fulfil this task were started in more than one topic area and will be investigated before the start of an own attempt.

Initial survey

Quality of water in the United Kingdom is ensured by a number of organizations consisting of governmental, regulator and consumer organizations, all of them having their own task including the following [6]:

- Governments
  - Defra
    - Looking after the natural environment
    - Supporting the food and farming industry
    - Sustaining a thriving rural economy
  - Welsh Government
    - Improving the lives of people in wales
- Regulators
  - Drinking Water Inspectorate
    - Providing independent reassurance about UKs water quality
  - Environment Agency
    - Regulating industry waste
    - Regulating water quality and resources in England
    - Managing the risk of flooding from rivers, reservoirs, estuaries and the sea
  - Natural England
    - Helping to protect England's nature and landscapes
  - Natural Resources Wales
    - Ensuring sustainability of resources in England
  - Ofwat (for England & Wales) and WICS (for Schottland)
    - Regulating the water and sewerage sectors
    - Setting price limits for customers
    - Ensuring companies run efficiently
    - Encouraging resilience
- Customer Watchdog
  - CCWater
    - Promoting customers interests to governments, regulators and water companies
    - Providing advice and complaint handling service for customers

Those companies might all have interests in the water specific data. The source of information for possible questions to our system will be their webpages.

The already mentioned water suppliers will need to be investigated on their provided data and on their interests.

Government website [5] provides all regulations which need to be followed by water suppliers.

The Biobank Standardization and Harmonization for Research Excellence in the European Union described an attempt of harmonization of various different sources of data. [7]

Aims and Objectives

The purpose of this project – and this part of the project in specific – is to investigate and design knowledge and data engineering infrastructure for big amounts of water and wastewater treatment process specific data. This includes finding the best way to gather data from water providers in terms of cost, efficiency, effectivity and security, as well as harmonizing and providing the data in the most fitting manner. The below picture shows the idea of how the system should interact with the outside world. In this picture the Large Network Performance Collider is our Data Harmonization Layer and Data Provider on top of which our partner project creates analyses and reports.
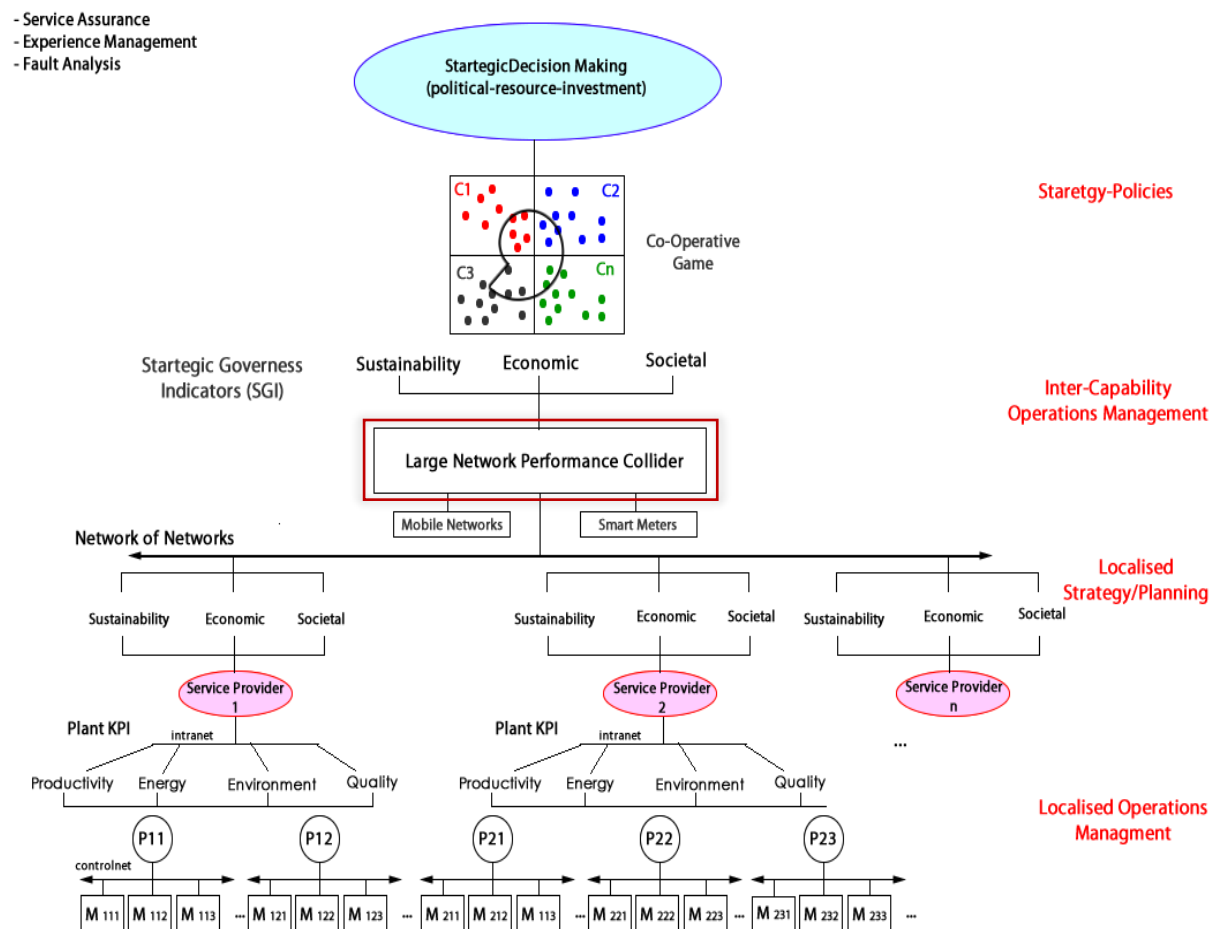
(Source: [8] Page 3)

Defining questions, which the system must/should answer. **Designing a fitting data schema** with appropriate target variables. In the desired solution, the system should **gather data** from different service providers, without their need to adapt their data schema, and **convert** the data to an own, predefined and harmonized data schema as well as **provide** this data to third parties. In the best case, all required data is provided to the public by every service provider in some way – then the only problem is the harmonization of the data. Aside of the real time service provider data, also historical data should be accessible through our system. This data is meant to be used for comparisons and forecasts and needs to be adapted to our data schema as well. Adding new sources of data to the system should not affect the systems performance by a noticeable amount and data requests should be processed in the most efficient way possible. Adding additional variables of interest should be possible.

Aside of the techniques, investigating on what the best technologies are to fulfill the task is also a part of this project. This also includes finding the best Data Host.

Experimental/investigative methods to be adopted

The most important investigative method in this project will be pure research which will show if there is already an existing harmonization system which is close enough to our use-cases to be adapted and applied in order to fulfil this projects purpose. Since there is no such working system in the field of water and wastewater treatment, the investigation will have to be expanded into familiar areas (like electricity supplement) and further into areas, where the field is completely different but data harmonization is also applied (like studies).

The following is a closer look at a project which attempted to harmonize data from very different sources. The article describes this attempt in a highly abstracted manner which could be used as a guidance for our project, since 4 steps out of 5 which the project needed to manage will be a problem in this system as well [7]. A closer look is required at this point in order to get an overview of what will be needed to be taken into account when making a time table and planning the system.

The Biobank Standardization and Harmonization for Research Excellence in the European Union aimed at pooling and harmonizing large amounts of population-based studies coming from six different European countries.

The attempt included a lot of communication between the studies resulting in a set of 96 variables describing questions of interest. The harmonization was assessed using:

- The studies questionnaires
- Standard operating procedures
- Data dictionaries

Possibly harmonizable data was processed in an open-source software and transformed from study-specific into the target format. The harmonized data from each center was then placed on a centralized database for further analysis. The result was a generation of common format variables for 73% of matches considered (96 targeted variables across 8 studies).

The conclusion of this pilot project was: "*New Internet-based networking technologies and database management systems are providing the means to support collaborative, multi-center research in an efficient and secure manner. The results from this pilot project show that, given a strong collaborative relationship between participating studies, it is possible to seamlessly co-analyse internationally har-*

monized research databases while allowing each study to retain full control over individual-level data. We encourage additional collaborative research networks in epidemiology, public health, and the social sciences to make use of the open source tools presented herein." [7]

Problems which came up during the project:

- Managing and harmonizing large amounts of data from different sources
- Ethical,
- Legal and
- Consent-related restrictions

Benefits resulting from the harmonization:

- Integrated data allows for lot bigger sample sizes
- Improved generalizability
- Easier ensuring of result validity
- More efficient secondary usage of existing data
- Provides opportunities for collaborative and multi-center research
- Satisfies Governments, funders and researchers

A closer look into the harmonization process:

1. **Recruiting studies to participate in the project**
   The requirements for a participation were collection of specific data needed for the analysis. They studies were also required to allow remote access to the collected data. Another requirement for the studies was to make study metadata (questionnaires, data codebooks, standard operating procedures) and ethical and legal documents/policies available to the BuiSHaRE coordinating group. A standardized online description form found on the Mica-powered[17] BioSHaRE website. The cataloguing process was helpful for the understanding of the heterogeneity level between the study designs, as well as for the potential sample sizes available for the analyses.

2. **Defining a set of target variables**.
   The purpose of those variables was to answer specific research questions. This set of variables was the **Data Schema**[18] of the project and defined the common format of the different studies. A common schema is needed to work with multiple independent studies. Developing such a schema needs to be done carefully, since it requires a balance between uniformity and acceptance of a level of heterogeneity across the studies, meaning the same questions and data collections are likely to be phrased differently. Those variables were selected within two workshops organised for the studies, each of which targeted one specific *question which needed to be answered by the project*, and defining which variables would be needed to answer it. Each variable was described with 3 properties:

---

[17] Mica [38] is a software application developed to create web portals for individual epidemiological studies or for study consortia. Including number of participants, their characteristics, methods of recruitment and so on...
[18] DataSchema: https://www.bioshare.eu/content/healthy-obese-project-dataschema

- Variable(-name)
- Definition
- Format

3. **Defining the potential for each of the studies to generate each of the defined target variables**.
This step included a deeper look into every study and determine their compatibility with the defined Data Schema. This not only included the value being present in a specific study, but also having a meaningful value – e.g. weight not being self-reported by a participant but instead being measured by a doctor. Having this restriction meant, that not all studies would be able to provide all 96 defined variables, but as much as 73% of the sought information. The difficulty in harmonization of data differed between the variables.

4. **Processing of the acquired data**
For processing of the variables the, a software called Opal[19] was used. This software required data, needed to calculate values covered by the target variables to be extracted to dedicated Opal servers, as well as a reference to the DataSchema structure. Afterwards, the algorithms, calculating the harmonized variables data were defined for each of the studies. The derivation of each variable was completely independent of the derivations of the same variables in different studies.

Example of a variable calculation:

---

[19] "Opal is an software application used to manage study data and includes a software infrastructure enabling data harmonization and data integration across studies. As such, Opal supports the development and implementation of processing algorithms required to transform study-specific data into a common harmonized format. Moreover, when connected to a Mica-web interface, Opal allows users to seamlessly and securely search distributed datasets across several Opal instances." [Source: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4175511/#B1]
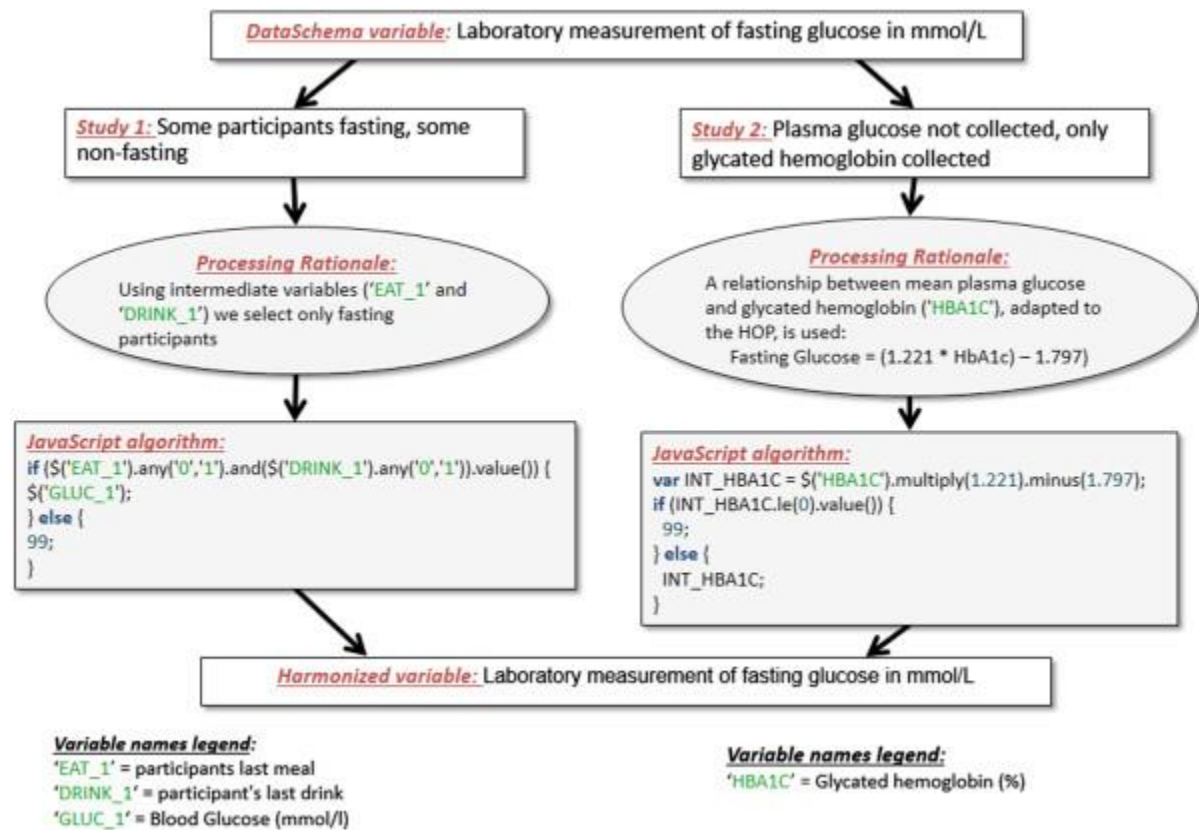
5. Creating analyses using the acquired data

Analyses are a part of the partner project and will not be covered within this document.

Time-plan

	2017					2018		
	August	September	October	November	December	January	February	March
Interim Report	01.08	28.09						
Documentation			01.10					09.03
Define questions to be answered by the system			01.10-07.10					
Investigate service-provider offered data			04.10-14.10					
Define DataSchema			14.10-31.10					
Research for DataHost / Data Integration technique / technology				1.11-14.11				
Research for Data Harmonization technique / technology				1.11-14.11				
Research for Data Providing technique / technology				1.11-14.11				
Specification				14.11-30.11				
Architecture								
Implementation					1.12		28.2	
Testing							14.2	01.03
Integration								01.03-09.03

Interim Report: This document

Documentation: Master thesis including all research questions and all following documents

Define Questions: This step not only includes defining the questions which the system needs to be able to answer, but also finding out all different ways to answer those questions. A question might have different ways to get answers because a variable might have different ways to get calculated.

Investigate service-provider offered data:

Investigate on which service provider offers what kind and which amount of data, what the quality of data is and what the conditions are to be able to access this data.

Define DataSchema: Within this step the data schema will be defined. This is how the harmonized data will look like. It is important to see the target variable coverage by all suppliers, because if a variable can only be calculated by 5% of suppliers it might not be useful to support it.

DataHost Research: This step includes finding the best way to store the data. This contains researching the techniques used system solving similar problems in other fields. Not only does this step include finding the most fitting technology (i.e. Cloud), but also the most fitting technology provider (i.e. Amazon)

DataHarmonization Research:

This includes finding the most convenient way to convert the data from provided DataSchema to our own DataSchema. Again, comparing solution chosen by similar systems is a part of this step.

DataProviding Research:

This step is about finding the best way to provide the collected data to the partner project / 3rd party systems.

Specification: Creating a specification for the system. This document will define the scope of the system as well as the specific technologies and techniques which will be used to implement all system compo-

nents. This document also includes the specification of external communication with the system.

Architecture: Creating the highest level of the architecture to define, which components of the system will interact with each other and with the outside world.

Implementation: Implementation of the system. (This step will be split in smaller tasks once the system, technologies and techniques are defined.)

Testing: Unit-/ and Black box tests, (Integration tests)

Integration: Integration of the system and testing with the partner project

Deliverables or specific outcomes

The outcome of this project includes a solid fundamental research on 3 areas: The most fitting way to acquire and store data from public service providers, as well as from any historical data sources in terms of efficiency, pricing and considering any legal issues. This research also takes extendibility into account which is mostly about the adding of new data sources. The second research area is the data harmonization. This includes the most efficient and effective way of determining target variables and the data schema as well as the most appropriate technique and technology for converting the acquired data into the self-defined schema. The last and also least weighted research is on the best way to provide acquired and harmonized data.

Aside of the researches, this project will include at least a prototype of all the 3 mentioned modules, which should be able to pull data from a specified source, transform that data and provide the data for defined 3rd party systems. This prototype will be created the way, so that it also acts as a proof of concept for the research results.

References

[1] A. S. o. C. Engineers, Failure To Act - the economic impact of current investment trends in water and wastewater treatment infrastructure, 2005.

[2] W. UK, "http://www.water.org.uk," 28 09 2017. [Online]. Available: https://www.water.org.uk/about-water-uk/wastewater.

[3] W. UK, "https://www.water.org.uk," 28 09 2017. [Online]. Available: https://www.water.org.uk/consumers/what-water-companies-do.

[4] F. a. R. A. Department for Environment, "Sewage Treatment in the UK," Crown, 2002.

[5] U. Government. [Online]. Available: http://www.legislation.gov.uk. [Accessed 28 09 2017].

[6] W. UK. [Online]. Available: http://www.water.org.uk/about-water-uk/regulation. [Accessed 28 09 2017].

[7] P. B. Y. M. A. G. B. H. R. W. M. P. R. P. S. L. F. C. M. M. W. R. H. K. K. H. L. H. A.-M. Dany Doiron. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4175511/. [Accessed 28 09 2017].

[8] E. K. &. A. Mousavi, "WWTP-Global-300617," 2017.