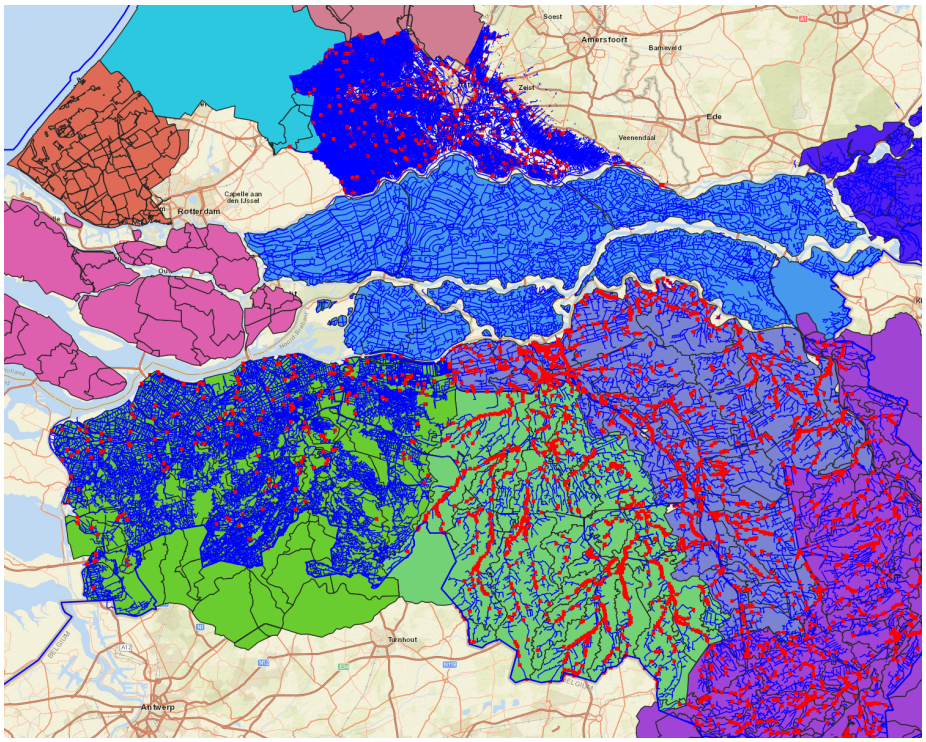
Data quality improvement of the NHI database

A civil engineering bachelor assignment



Rob Rikken

August 22, 2019

Preface

This report is the product of ten weeks of work at Deltares, and is the final step of the bachelor part of civil engineering at the University of Twente. The reason for this assignment was the possibility to improve the database of the NHI. Rules to check a database had been developed by Gerrit Hendriksen in the past, but the code was not compatible anymore with current technologies.

First I would like to thank Gerrit Hendriksen for the supervision of my work. With a few short tips and explanations I could quickly figure out what theory needed to be read, or which people should be interviewed. Especially Maarten Pronk, who within 45 minutes, gave me a good idea to solve the method of querying the AHN webservice. For the writing of the code in this report, I have made use of open source code, and would like to thank the communities that develop PostGIS, and Geoserver. Without these free technologies, this research would not be possible.

For the people reading this report trying to get a better understanding of my code, the result section and the appendix with the code explanation will be the most interesting

Summary

For the hydrological data and models for all of the Netherlands, there is an instrument called the Nationaal Hydrologisch Instumentarium (NHI). In the NHI there are models and data, the surface water data of the NHI is saved in the HyDAMO database. This HyDAMO database is filled with the geographic information system (GIS) data of the waterboards. The data is already checked for semantics, but there are consistency errors in the data. This research looks at the consistency in context of data quality.

Now all the systems need to be added together, inconsistencies are detected. These inconsistencies preclude the advantages of the NHI and HyDAMO to be fully realised. There needs to be pre-processing and error correcting before models are made, or data is used for other purposes. To help the waterboards with making the data more consistent and to communicate the data quality, Deltares has started this research.

Over the years waterboards developed their own way of working, different from each other. Although efforts where made to make the entries more consistent, DAMO is the latest iteration, full agreement how to add data to the GIS system has not been developed [13].

For every data type, rules to check the data quality are defined. This is done with the help of literature and interviews with experts. The rules are then mathematically defined, and implemented into computer code. The results of this computer code is then saved in the HyDAMO database.

The results show that the data quality of HyDAMO is still lacking in certain areas. Especially the objects called 'hydroobjects' and 'dwarsprofielen', have a low data quality. These results are presented to the waterboard via a web service that they can log into, and then check the objects that are erroneous.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the last decades, computing power has increased exponentially together with the sensing of our environment. These advancements have led to large amounts of data being gathered, and the need to analyse this data. Now the global and local data gathering efforts are able to be combined into large databases. These databases now hold all kind off different types of data. In the Netherlands a combination of global, national and locally available data in geographical form, is currently being processed for the waterboards. This effort is being made to improve the support the waterboards receive with their data needs for hydrological modelling.

1.1 Netherlands Hydrological Instrument (NHI)

Around the year 2000, regional and national models, data and technologies for hydrological modelling existed. There was an urgency to bring together all the knowledge across the different parties to make national available database. Deltares started working on moving all the different models and data to one location, the Nationaal Hydrologisch Instrument (NHI). In 2013, thanks to all the parties working together, a consensus on the NHI was reached on how to integrate the waterboard data and models with each other [4]. This meant that all parties could start using the data and models from the NHI.

The NHI combines different concepts into one model, hydrology and runoff models are all coupled together. To combine these models, with different backgrounds and data needs, the data is scaled and transformed based on the need of the models. The regional and national databases, that are owned by varying partners, are also coupled with each other. All the data and models used by the NHI are meant to be open source and freely accessible to all parties, with the organisations that benefit from the NHI all contributing to it. The contribution consist of monetary support, but also from expertise and code shared by parties.

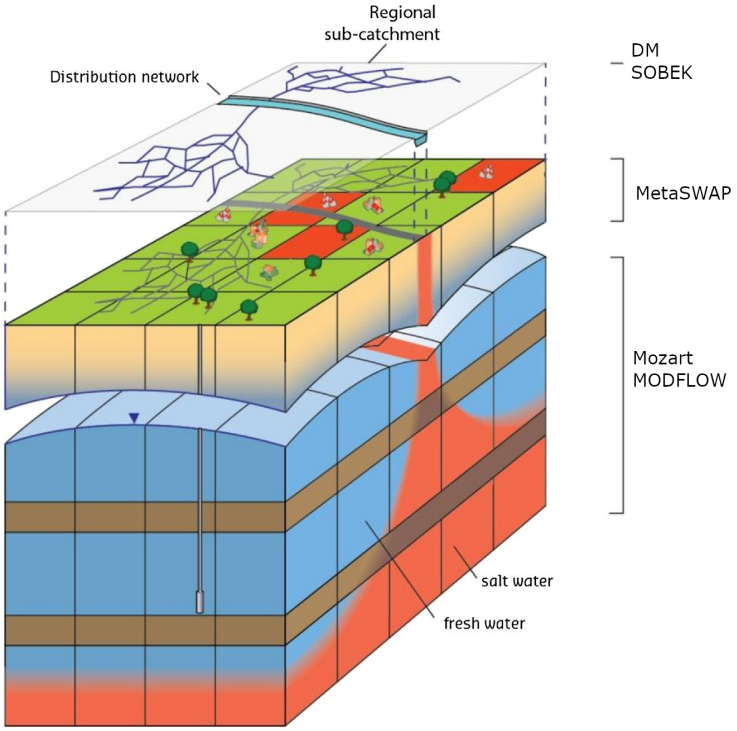


Figure 1.1: The domains of the five models in the NHI [4].

1.2 Nationwide Hydrological Model (LHM)

There are five models in the NHI that together calculate the flow of the surface and groundwater. Each model has its own domain and these domains are connected to each other via water fluxes. In figure 1.1 the different domains of the models are presented.

The DM (distribution model) is used for the optimisation and distribution of water. The model uses a simple representation of the main rivers and the Ijsselmeer. With this representation the model allocates the water to the users and allows for alternative routes of the surface water. The alternative routes are used to simulate water distribution to combat salinization and dike instability in periods of shortage. The second model used for surface water is the SOBEK model adapted for national scale. National SOBEK can calculate 1D and 2D flow, water quality, salt intrusion and morphology. To calculate the model at a national level, the regional data is used, but is up scaled and the setup from regional and national water authorities is used. This national model is called the 'Landelijk Hydrologisch Model' or LHM. The two surface water models work together to build a complete picture of the surface water.

The other three models work together to create a picture of the subsurface conditions of the Netherlands. Mozart is the model that moves the water from the surface to the subsurface using sub-catchments. These sub-catchments are in contact with each other and the groundwater model. Fluxes from the DM model are then distributed to the sub-catchments, and drainage is calculated to the groundwater. The groundwater component of the NHI is the MODFLOW model. For the NHI the way the data is input and output from the model is adapted for use over the whole of the Netherlands. Fully saturated groundwater is calculated with MODFLOW, MODFLOW used an aggregated version of the REGIS database. The REGIS database is a description of the subsurface of the Netherlands in 153 layers, which is simplified to 7 layers. The groundwater component can also, with an add-on, calculate the salt loads in the subsurface layer. The MODFLOW model is currently undergoing a rework, to enable parallel computation. MetaSWAP is the model used to calculate the column between the saturated groundwater and the atmosphere (the unsaturated zone). Vegetation and the transpiration is pre-calculated in a database, this way it can be used within the NHI.

1.3 HyDAMO database

The HyDAMO database is the database within the NHI where all the surface water features are stored. This database is a subset of the databases and data that are used by the waterboards, focused on the needs of the hydrologists (the 'Hy' in HyDAMO). These waterboards use the DAMO data model to map the features [13]. This DAMO data model takes into account the regulatory commitments the waterboards have, like the INSPIRE hydrography specification [8, p. 65] or the BGT (basisregistratie grootschalige topografie [12]). For this study, the focus is mainly on the data quality of the HyDAMO database, because it is easily query-able via SQL (structured query language).

Waterboards can add their data to the HyDAMO database, after which it is provided to the general public via a data view portal. The data from the waterboards is already checked for the semantics and structure. Semantics means that all the columns of every record has a value of the correct type. These schemata make sure the syntax of the data is correct, and are implemented in Geographic Markup Language (GML) [17]. The underlying technology for checking if the data adheres to the GML is using XSD, a way of making sure XML files are using the same standard [9].

Waterboards, Rijkswaterstaat and private parties are all interested in using the data and models in the NHI. Working together has been the primary focus and reason for success of the system, but working together on a single solution, means coming to a consensus on how the data must be structured. Before this is input into the models, the data must undergo some transformation to be compatible with the NHI database system. These can be scaling, consistency and type transformations. Parties believe the data transformation is useful, but the data is not always delivered

without errors. In this landscape, Deltares, Rijkswaterstaat and the waterboards of the Netherlands are working together on unifying the data that they gather into the system.

The problem becomes then to integrate the databases and check them for consistency. Errors in data need to be corrected before the models are run, because the model input errors will propagate through the whole of the model. There are many ways errors can creep into data, but the NHI has a distinct advantage to other modelling efforts; the data is sourced from multiple parties. These data sources can be compared against each other, and when one or more sources do not have the same value, they can be marked as erroneous. If there is only one source of data, it is harder to prove that the data is correct or incorrect. To make use of these different data sources to detect the inconsistencies that are not caught by the GML schema, is the goal of this research.

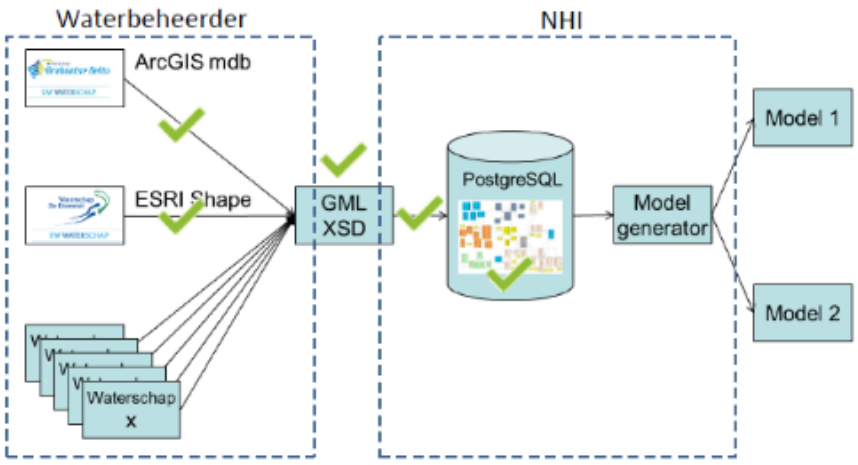


Figure 1.2: Waterboard data to NHI database [15].

1.4 Data

In figure 1.2 the different data sources can be seen. The waterboards have data sets for the HyDAMO database that are in the ArcGIS and ESRI Shapefile file structures [6]. When the data from the waterboards is delivered to the NHI, the GML and XSD schema get to work, filtering the data. When the data is able to pass through the schema it is added to the HyDAMO database. This database is implemented as a PostSQL database with the spatial extension PostGIS [20]. Via the NHI data portal files can be downloaded that contain the HyDAMO data, or the HyDAMO data can be accessed through a web service for maps. The data can also be directly interfaced with via GIS programs like ArcGIS [7] or QGIS [18].

The data used for this research can be found in the data portal of the NHI and the PDOK viewer [19]. This data can be accessed using the website of the portal, or when directly interfacing with the data, via a Web Feature Service (WFS) [24] on the Geoserver [11] implementation used by Deltares. The data from the NHI that is not in HyDAMO is available in three scales as a raster, 25 meters, 100 meters and 250 meters. The objects in HyDAMO are defined as GIS objects, (are vectors), so they can be scaled to the scale that is needed. The data of the AHN (height map of the Netherlands) is available in 5 meters and 5 decimetres. When looking at the cross sections the smallest scale is used.

Datasources:

- LHM 3.3 (Landelijk Hydrologisch Model)
- GEOTOP (Subsurface layers)
- REGIS (Deep subsurface layers)
- HyDAMO (Surface features)
- AHN 2 & AHN 3 (elevation map)
- Waterboard data (watercourses and objects)

From these data sources the AHN is the most precise as far as height data is concerned. The accuracy is defined as a 5 centimetre systematic error and a 5 centimetre stochastic error [25, p. 7]. Together, this means that 99,7 percent of the measurements fall within an error range of 20 centimetres, and that 95 percent falls in a range of 10 centimetres.

Next to the database itself, HyDAMO also consists of a data model [22]. This data model defines the GML mentioned earlier. In the current state of the database and data model, more objects have been defined then are in use in the database. For the research only the tables that have records are used, because without records the code that is implemented can not be tested.

1.5 Problem definition

Over the years, a lot of effort has been expended in putting together a hydrological system of the Netherlands. The surface water data is now collected in the HyDAMO database, but this database lacks consistency. Next to these inconsistencies, different models based on this data, have different requirements. The semantics of the data is already checked, but the consistency and ability to provide models with data need to improve.

1.6 Goal of the research

The goal of this research is to find ways of improving the consistency of the data quality. Through improvement of the consistency, less effort will have to be put into processing the data before it is useful. The other goal of this research is to define rules to which the data should adhere, and communicate these rules in a concise way. Communicating the rules and the clear definition of errors will help the waterboards with improving their data.

1.7 Research questions

As mentioned earlier, an advantage that the NHI has, is that all the data and models are brought together. This way, the data can be checked for internal consistency and against other data sources. This research will try to answer two questions, one about the data quality itself and one about the communication of that quality. With the questions 1.1 and 1.2 that support the first question.

> "How can the data quality of the HyDAMO database be improved?" (1)

The data quality in the HyDAMO database is found lacking by the waterboards and the parties using the data. The goal of this research is to define the errors in the database, and

> "What is defined as an error?" (1.1)

In measurements there are always uncertainties, but how big would a measurement difference have to be, to consist an error. Using the measurements of quality, we can say something about the definition of an error in context of the NHI. For every comparison between data sets, and for general rules, an error boundary needs to be defined.

> "Can rules be defined to improve the data quality of the NHI database?" (1.2)

When the definition of an error is clear, they need to be found in the data sets. To find these errors, rules need to be defined, so the errors can be detected via computer software. To build the rules, the parties that define a database 'fit for use' need to be questioned, and a list of rules made.

> "How can the data be presented to the users so they can correct the errors?" (2)

The parties in the NHI are diverse, and do not all have a clear understanding of the inner workings of the NHI. They do have a good understanding on how there own processes work and how they input the data. So the errors should be presented in a way that is clear to the end user, so they can correct the data. There should exist no ambiguity about why an error is an error.

Chapter 2

Theoretical framework

To answer the question if data can be combined to improve the data quality, first data quality has to be defined. Information and data quality can be split into different topics, these topics are called dimensions in the context of data quality. The quality dimensions and the schemata dimensions, together, are important for data quality. Where the schemata are important to combat redundancy and anomalies, the data dimensions are more relevant to daily use of data [3]. The data dimensions, that are used in this research are defined in this section. The theory behind data quality for this research comes mostly from the works of Battini [3][2] with additional sections from Morrison et al [16], Huh et al [14] and Shi et al [23].

Data quality can refer to the intention of the data, their schema, or to the extension, the values of the data. These usually are presented in a qualitative way, with no quantitative measures provided. To capture these in metrics, a few dimensions have been defined [2]. Often used is the measure of fitness for use, as in, can the available data be used for the task at hand.

2.1 Accuracy

Accuracy is the closeness of the recorded value to the real-life value. Two kinds of structural accuracy can be identified, syntactic accuracy and semantic accuracy. Next to these, because the world changes as the time goes on, there is another type of accuracy: temporal accuracy. This is the measure at which the data is updated when the real-life value has changed. To define how accurate the value are a ratio can be defined between the accurate values and the total number of values [3, p. 100].

Syntactic accuracy is the closeness of the value to its domain. This is not a comparison to the real-life value, but rather if the value is in the accepted range of values. For example, a placement of a GIS object might be in the correct projection, with

a longitude, latitude and elevation, but the values of those measurements might place it in another province. This would make the value syntactically accurate, but not semantically accurate. A metric for this type of accuracy, might be the ease of converting the projection, to the projection used in the database.

Semantic accuracy is the closeness of the value recorded into the database and the real-life value. Now the longitude, latitude and elevation do matter. The further away the recorded value is to the real-life value, the lower the accuracy. This type of accuracy should have bounds defined where a value is accurate, this would be the size of the accepted measurement error. No real-life value can be recorded perfectly accurate, so measurement errors will always need to be defined.

In case of temporal accuracy the size of the error is the time for the real-life value to change. For this research, temporal accuracy is the least important, as real-life values change relatively slowly in context of the data recorded. Especially height data like the AHN, can take years to update [1]. So the temporal errors in the data, fall outside the scope of this research.

To detect the errors in the data, deductive or inductive inference can be used. Inductive inferencing means to build a set of user defined error conditions, to build a set of conditions that may be compared to situations to detect an error. In the case of GIS data, the users could look at a map, and compare two data-sets, for example a photographic map and a data-set of pumping stations. If the pumping station does not show in both, the pumping station could be flagged as an error. Using this, means gathering the error conditions from the users and putting them into a database. Because of the need to finish this research in time, inductive inference will not be looked at.

Deductive means using general rules that are always valid, to check conditions against. A general rule might be "the river cross section cannot be above ground level". Using a set of these rules, errors in the data can be detected.

The NHI is in the unique position that it has sources for the same GIS data input by different parties. A good example is the ground level of the Netherlands, REGIS, GeoTOP and the AHN all have a ground level measurement for the whole of the Netherlands, but this will never be the same value for all them. When a value is out of family, (this could also be a deductive rule), this signals an error.

2.2 Completeness

Completeness can be defined as the extent to which data are of sufficient breadth, depth and scope for the task at hand. Important here in is the task at hand, because the data can never be a complete picture of reality. For completeness, there are three types that are defined: schema completeness, column completeness and population completeness. Schema completeness is the degree to which the concepts and properties needed are all present in the database. Column completeness

the measure to which values are missing from a record. Population completeness measures if all the records are there from a reference population, are, for example, all the weirs from an area represented in the database.

The task at hand is the calculation of the models in the NHI and the values in the database need to be complete in the sense that those models can be run. To evaluate the completeness of the database of the NHI, the completeness dimensions need to be evaluated against this reference.

2.3 Consistency

Consistency captures the dimension of the violation of the semantic rules defined for a set of database items. The correction of consistency errors is called imputation, and the rules that are formed to detect such errors are expressed as 'edits'. This research will mostly concern itself with the edit-imputation problem, which is the localisation and correction of errors.

2.4 Edit rules

The rules for detecting errors in the context of data quality are called 'Edits'. These edits came into being when checking questionnaires for errors. An example of a edit rule is that the underside of a bridge can not be higher than the topside of a bridge. A formal definition of this rule would be: $undersideheight > topsideheight$. When these rules evaluate to true, the value in the record is deemed inaccurate and must be changed (imputed) to reflect the real world value of the object. The combination of the edit rules and the following imputation is called the 'edit imputation problem'. Using this formal language for every rule that is used in the checking of data quality, a concise and clear representation can be given. These formal definitions can then be translated into computer code, to check the data for erroneous records.

Chapter 3

Methodology

The methods in this section describe how the research is conducted. The definitions in the theoretical framework will be used to define the errors found in the data. Then rules will be defined to find errors, these rules will b converted into code, and last, the errors and rules will be presented via GIS layers available via a web server.

The data is stored in a relational database, and that has GIS functionality added to it. The relational database that is used, is Postgresql, this database system is open source. This is an important quality in the context of the open government [21]. With this GIS functionality, values can be stored in normal table records along side a geometry column that defines the spatial object of that record. Every record has one geometric representation, so the data model has one geometry column for every table. To add GIS functionality to the relational database, the PostGIS database extension is used. This extension adds geometric functions and types to the Postgresql database. These functions can be used just like normal SQL queries in a relational database. Queries can not only be run on the attributes of the data, but also on the spatial properties. When defining functions to detect errors, the spatial location of a object is often important. For example, when checking the correct location of a river or ditch. Spatial properties can also be used to join tables and to query these properties. These properties of the Geospatial database will be used to convert the rules for checking the database quality into code.

First the data quality problems that are in the database need to be clearly defined. This will be done through interviews with the model builders that use the data and the waterboards that are providing the data. The errors will be classified according to the theory found in the theory section. An example of a classification is 'completeness', through this classification the errors can be defined better than plain text. When not all ditches in an area are input into the database, the records are clearly incomplete, and the error be classified as a 'completeness' error. When the errors are clearly defined, the next step can be taken.

Second, the rules to find the errors need to be defined. For surface water, this would

consist of rules to check if the watercourses in the database are valid for input into the model. An example would be if the culverts that transport the water, are defined at the same location as the watercourses (culverts are part of a watercourse). These rules will be first defined in human readable format, and will be listed as such. A rule would be, 'a culvert needs to lie within 10 metres of a hydroobbject'. Multiple rules can be defined to find the same error, and these should all be recorded as some rules might be easier to implement then others. After a exhaustive list of rules is produced, a selection of these rules will be implemented. The selection will depend on the time needed to implement them, and the importance placed on them by Deltares. When the rules have been gathered, to aid in the translation to code and a clear definition, edit rules (as described in the theory) are defined for every rule.

After the rules are defined, they will be converted into code. This code can both be written in language that works with the database or into separate code, depending on the nature of the rules and the data used. If the data is not in the HyDAMO database, the database language can not be used, so these rules would need to be written using standard computer code. For database rules, SQL with PostGIS functions will be used, when rules across databases and other sources are needed, Python will be used. Examples of queries using SQL would be to check if the geometries intersect, if the geometries are valid or what the distance is between objects. An example SQL query can be found in code section 4.1, here SQL is used to find if a watercourse has a cross section. The AHN database can not be accessed in this way, the access to this database is provided by PDOK as a web service. This web service needs to be accessed via the internet using a web feature service, which operates analogously to a web API. Using Python the connection to the AHN database can be made, using the python code rules using height information of the AHN can be implemented.

When the rules have been implemented, the errors can be saved to the HyDAMO database. As the data is now available as a database table with a link to the erroneous object, the choice has to be made how the data is presented to the user. To be sure what the most effective way of displaying is, the waterboards are questioned on their use of the error data. A choice is made to either, display the error data freely, or place it behind a login to let the waterboards only access their own data. These options are available within the Geoserver where the results are presented on. Each waterboard can have its own account and login, or all data can be made available to the public. The acceptance of the waterboards of the data quality assessments is the most important in this decision.

Chapter 4

Results

The results consist of rules and information gather from interviews, implementation of the rules and the presentation of the rules. These topics are each presented shortly. A more detailed result can be found in the appendices, the code can be found on GitHUB [5] and the resulting web services are available to the waterboards.

4.1 Improving the quality of the HyDAMO database

In this section, the research questions are answered. To find out how the data quality of the HyDAMO database can be improved, the questions 1.1 and 1.2 are answered. At the end of the results, there is also an example given of how a rule can be used to assess the data quality and possibly improve it. The improvement of the data itself will be done by the waterboards, so the results here, are tools for the waterboards.

4.1.1 Error definitions

The results in this subsection belong to research question 1.1. The definition of the errors in the database has been defined by interviewing experts on the usage of the NHI in practise. Next to these interviews, some rules for hydrography data are also found in the INSPIRE documentation, an explanation on INSPIRE and the link to the results can be found in appendix F. In the meetings more data quality rules where defined than presented here in the results section. The rules that where selected had the constraint that records must be present to test the rules on (only part of the tables in the database have records).

The rules described in table 4.1 all have a classification. This classification is the classification that is deemed most likely, because the errors can only truly be classified if these are compared with the real value. For example: the rule that a cross section must have a hydroobject, can mean that there is no cross section defined for that hydroobject (a population error). Or it can mean that the location of the cross section(s) that belong to the hydroobject have the wrong coordinates (semantic accuracy). This uncertainty in qualification of some of the rules, also makes it important to involve the producers of the data at the waterboards.

4.1.2 Rule definitions

Table 4.1: Error definition table

Id	Rule	Classification
1001	Catchment areas should not overlap.	Semantic accuracy
1101	A ground fall must lie on top of a hydroobject.	Syntactic accuracy
1201	A bridge must lie near a hydroobject.	Population completeness
1202	The top of a bridge should be higher than the ground level.	Semantic accuracy
1203	The bottom of the bridge must be lower than the top of the bridge.	Syntactic accuracy
1301	Every cross sections must lie on a hydroobject.	Population completeness
1401	The width and height of a culvert or syphon must be larger than zero.	Semantic accuracy
1501	The cross section should be within the measurement accuracy of the AHN value.	Semantic accuracy
1502	The low roughness value should be below the high roughness value.	Syntactic accuracy
1601	A pumping station must lie near a hydroobject.	Semantic accuracy
1701	Hydroobjects must be noded properly.	Semantic accuracy
1702	Every hydroobject must have a cross section.	Population completeness
1703	The low roughness must be below the high roughness.	Semantic accuracy
1801	A lateral knot should lie within the associate catchment area.	Semantic accuracy
2001	A pump needs to lie in range of a hydroobject.	Semantic accuracy
2101	A weir must lie near a hydroobject.	Semantic accuracy
2102	The lowest flow height needs to be lower than the highest flow height.	Syntactic accuracy

The results in this section belong to research question 1.2. The rules in table 4.1 now need to be defined as edit rules, to make the conversion to SQL easier. Clearly

defined rules will also make the communication about these errors unambiguous. In table 4.2 the error code is presented, together with the edit rule. If the edit rule is evaluated as true, the value it compares is marked as erroneous and a suggestion for improvement is saved into the suggestion table. For some edit rules, parameter values can be added if needed, like a minimum distance, or a measurement error range. These added parameters are not shown here, because they are implemented as changeable in the code, and can be different for every run of the code. Examples are the minimum distance to an object, or the AHN error margin.

Table 4.2: Edit rules results

Id	**Edit rule**
1001	$catchment_A \cap catchment_B \neq \emptyset$
1101	$ground\ fall \cup hydroobject = \emptyset$
1201	$\sqrt{(x_{bridge} - x_{hydro})^2 + (y_{bridge} - y_{hydro})^2} > minimumdistance$
1202	$ground\ level > bridge\ top$
1203	$bridge\ bottom > bridge\ top$
1301	$cross\ section\ line \cap hydroobject = \emptyset$
1401	$culvert\ height <= 0 \vee culvert\ width <= 0$
1501	$cross\ sectionlevel \neq ground\ level$
1502	$low\ roughness > high\ roughness$
1601	$\sqrt{(x_{pumpstation} - x_{hydro})^2 + (y_{pumpstation} - y_{hydro})^2} > minimum\ distance$
1701	$nodeconnections > 1 \wedge hydroobjectconnections > 2$
1702	$hydroobject \cap cross\ section = \emptyset$
1703	$low\ roughness > high\ roughness$
1801	$lateral\ knot \cup catchment\ area = \emptyset$
2001	$\sqrt{(x_{pump} - x_{hydro})^2 + (y_{pump} - y_{hydro})^2} > minimum\ distance$
2101	$\sqrt{(x_{weir} - x_{hydro})^2 + (y_{weir} - y_{hydro})^2} > minimum\ distance$
2102	$low\ inflow\ height > high\ inflow\ height$

An example of an error and a rule is a watercourse that does not have a cross section. The model that uses this data, needs a cross section on a watercourse to be able to calculate how much water can flow through a watercourse. In figure 4.1 in pink watercourses can be seen, with no cross section. The cross sections are coloured yellow, and the hydroobjects with a cross section are coloured blue. This hydroobject would not be able to be input as a watercourse into the model. In PostGIS a query can then be written to check if a watercourse is intersected by a cross section. If this query evaluates as false, then the hydroobject is marked as erroneous. The qualification of this error would be a lack of completeness.

Another example would be the height of the cross sections themselves. These cross sections should not be above the ground level next to the watercourses. To check for the correct height of the cross sections relatives to the ground level, the general height map of the Netherlands could be used. The start and end point of the cross section could then be checked if the are on the same height as the values on the height map.

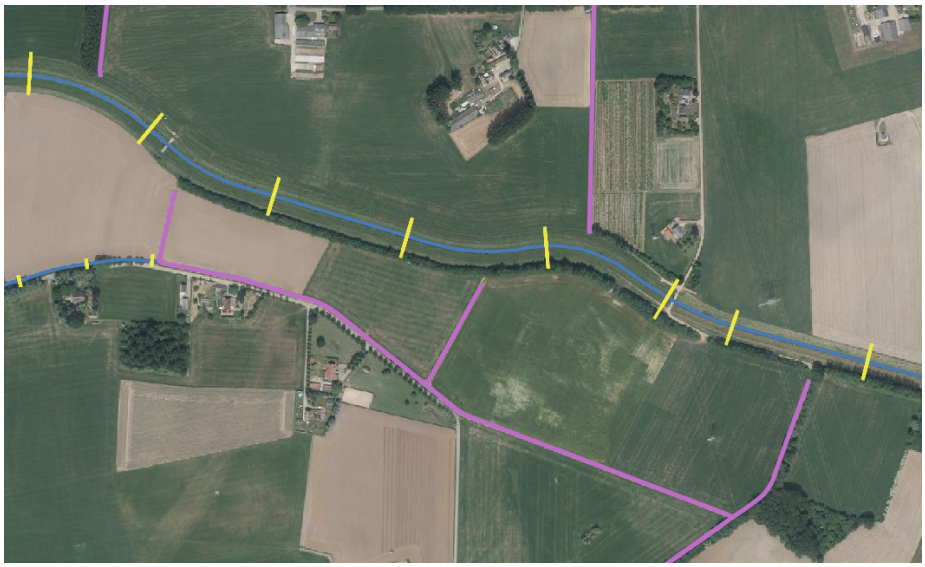


Figure 4.1: Hydroobjects without cross section (in pink), from the NHI database

With the previous results, the code for the error checking can be written, an example of a rule implemented in SQL and PostGIS can be found in appendix E. This code is its entirety has also been published on GitHUB on the NHI GitHUB page [5]. The results of these rules are the layers that can be viewed in GIS clients, but also the total errors per rule and for the height value, how big those errors are.

Source Code 4.1: Implementation of rule 1702

```
def check_if_object_has_cross_section(self):
    # Every hydroobject needs to have a cross section. So check
    # this with the intersection function if there are intersections
    # between the datasets
    with self.get_datasource().get_connection() as connection:
        results = connection.execute('''
            SELECT hydroobject.*, {quality_schema}.cross_section_lines.*
            FROM hydroobject
                LEFT JOIN {quality_schema}.cross_section_lines
                    ON st_intersects(
                        hydroobject.geometrielijn,
                        {quality_schema}.cross_section_lines.cross_section
                    )
                WHERE {quality_schema}.cross_section_lines.profielcode IS NULL
        '''.format(quality_schema=self.get_quality_schema()))

    self.insert_error_records(results, self.get_cross_section_suggestion())
```

To ensure a readable and well structured code, the code has been developed with object oriented methods. Every table in the database has its own detector that holds the logic that evaluates the rules. The detector for the table inherits some functions from a parent class called 'Detector'. This way every new model/table that is added to HyDAMO can inherit these functions. The functions to detect the errors, and the suggestion messages are the main body of the child classes. A function to detect one of the rules can be found in code example 4.1. For the rules that need external data sources, a helper is used, that gets the data from the external data source.

Every detector also has a function that can build the functions in the detectors as threads. Because the queries on the database take much longer than the Python code to run, the code needs to run in parallel. This parallel evaluation makes the code runs much quicker than if the code runs sequentially. When the database returns the results, the results are entered into the database and the thread is closed.

Table 4.3: Errors found in the HyDAMO data and their ratios.

Id	**Total**	**Erroneous**	**Ratio**
1001	10175	5412	0,468
1101	420	187	0,555
1201	3670	38	0,990
1203	3670	1	0,999
1301	146741	15328	0,896
1401	40883	1250	0,969
1501	2248334	1105954	0,508
1502	2248334	355768	0,842
1601	460	17	0,963
1701	167587	37153	0,778
1702	167587	91474	0,454
1703	167587	30278	0,819
1801	10523	531	0,950
2001	248	6	0,976
2101	7205	229	0,968
2102	7205	390	0,946

With the results saved, they can now be displayed. The first step is making a view that links the HyDAMO object together with the suggestion. This view is constructed during the setup of the tables in the data quality portion of the database. With these views, a GIS layer is published with Geoserver. This layer can then be accessed by the user. The detection code can run within 15 minutes, and should be run, every time new data is added to the HyDAMO database.

With the definition of these rules, we can now check the ratios [3, p. 162]. These ratios are meant to give in indication how good the data quality is. If a ratio is 1, no error has been detected, if the ratio is 0, all the records have an error attached

to it. So a larger number means a better data quality. For the rules that are also in the INSPIRE documentation, there may be different recommendations on how to present them (different from ratios).

4.2 Presentation

These results belong to research question 2. The presentation of the data quality is done by serving a layer of suggestions. These layers are served using web services from a Geoserver, behind a login. Login credentials are available that show a subset of the layers from the waterboard that logs in. In this way the layers can be accessed via the internet by all waterboards, while they cannot view the data of other waterboards. The users can then use the layers to check if there are errors in the data.

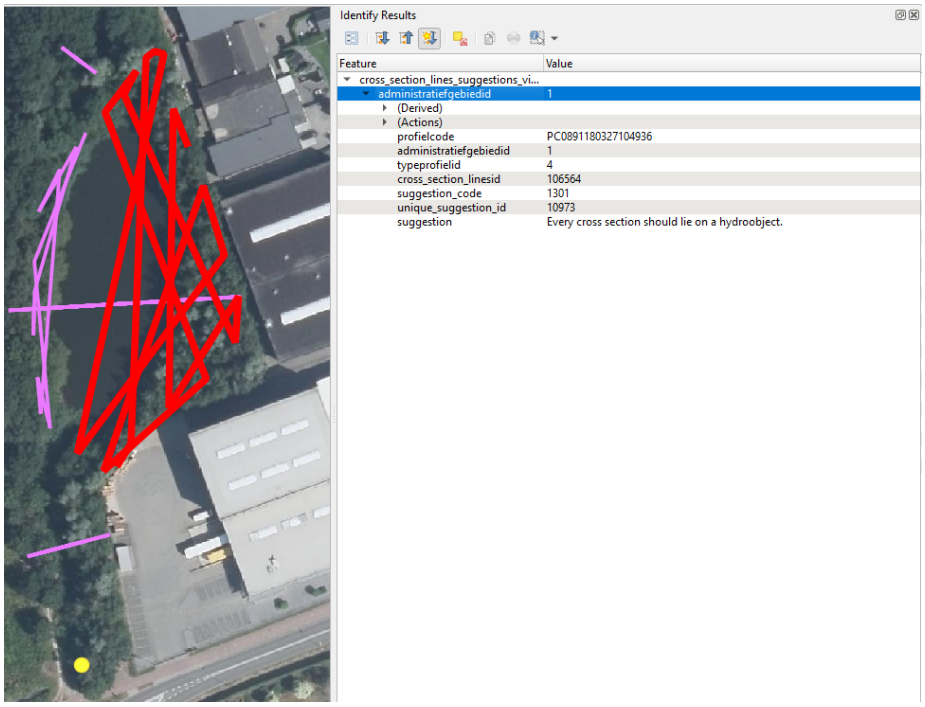


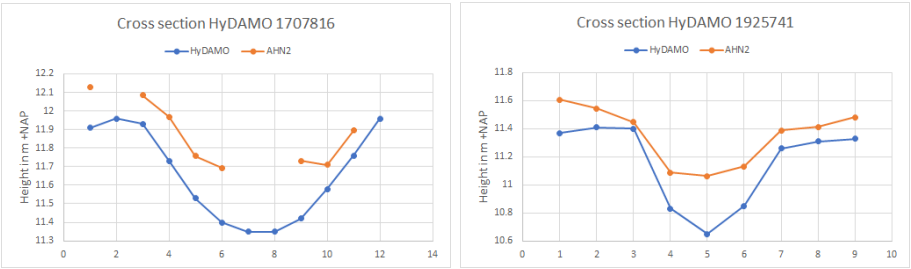
Figure 4.2: Example of the layer representation of the suggestions in QGIS.

In the meetings with experts that can be found in the appendix and the symposium on HyDAMO, it became very clear that waterboards would like to be able to check the rules before uploading it to the NHI server. Also the public display of the errors in the NHI portal was not something that was deemed feasible. The layer display for the suggestions was well received, so the presentation of the suggestions

was accepted. To be able to display the errors as a layer in the context of a GIS application, and to make the suggestions only available to the waterboard that uploaded the data. A Geoserver has been set up, with the layers locked behind a login. Using this solution, the data can be checked centrally by the NHI, and then the suggestions can be made available to to only the data owner. The data will be presented via layers, behind a login, on a Geoserver that resides at the NHI.

4.3 Example of rule 1501

Apart from the results that support the research questions, an in-dept result of a rule is presented here. The cross section comparison with the second version of the AHN (AHN 2) and the data from this. For all cross sections the difference between the HyDAMO and the AHN value is checked. First two cross sections are presented in figure 4.3 Then the histogram of all differences between the cross sections and the AHN values is presented, together with a table of the characteristic values of the differences.



(a) Cross section 1707816, missing values. (b) Cross section nr. 1925741.

Figure 4.3: Two cross sections and their AHN 2 values.

In the AHN there are multiple raster sizes and interpolation options. For this comparison, the highest resolution of 50 by 50 centimetre is chosen. The ground level data that closes small no data areas has been chosen for the interpolation (ahn2_int) This AHN product has values for every half meter where there are no buildings or water. With this data, the cross sections of HyDAMO can be checked if they have the correct height values. If the values of the HyDAMO cross sections are out of range of the AHN measurement error (20cm for 99,7 percentile), then the cross sections points will be marked with a suggestion. The figures 4.3a, 4.3b, give a two examples of the AHN values of the cross sections and the HyDAMO values of the cross sections. For one of the two graphs, there are missing values in the AHN data. These missing values are the places that the laser measurements (LIDAR) cannot measure, or not completely measure, the water level at those coordinates. For most of the cross sections, these values are missing.

The measurements of the AHN are made with a laser altimeter in conjunction with a

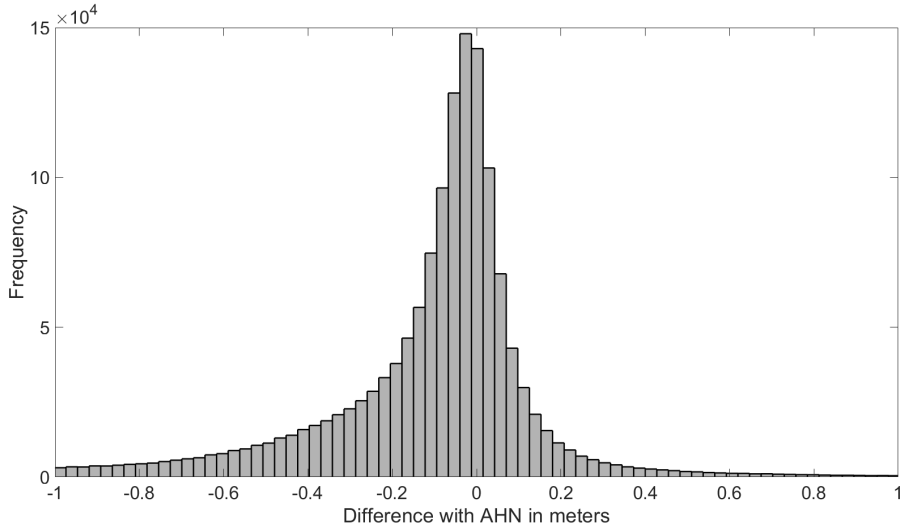


Figure 4.4: Histogram of the differences between the AHN 2 and the HyDAMO cross sections.

GPS system. Data from these measurements is a number of points with coordinates, a point cloud. For every 0,5 by 0,5 metres there is at least one point that determines the height. If there is no point for the square, the value in the raster data, will be no-data. Because of the way laser altimeter works, bouncing light of of objects, dense foliage's like grass, can not be filtered out. There is no way of determining where the grass ends and the ground begins. For large vegetation, this is possible, because there will be measurement around them.

As a general remark, the AHN values have the largest difference with the HyDAMO records, near the slope of a watercourse. This can have multiple reasons, but this seems to be a systematic error, and would be interesting to investigate in the future. A hypothesis is that on the banks of the river, there is usually a lot of plant growth. Measurements by the AHN are taken from the top of the dense foliage, and the measurements for the cross sections, are taken at the true ground level. This would explain the large difference between the measurements on the banks. If this hypothesis is true, it would mean that the rule should be adjusted to add a larger range than 20 cm, for when the HyDAMO value is smaller than the AHN value. Because the foliage adds height to the AHN value, cross sections values that are higher than the AHN values, are most likely to be wrong. This way the margin above and below the AHN value can be defined separately. The values chosen for this research are 10 centimetre above the AHN value, as the AHN defines as the 99,7% value where in the margin of error lies, and 20 centimetre below the AHN value. This allows for a 10 centimetre extra margin, which is presumed to be the average height of foliage's.

Table 4.4: Important descriptors of the cross section differences.

mean	0,4201 meters
median	-0,0600 meters
5th percentile	-1,0860 meters
95th percentile	0,1940 meters
Minimum value	-554.6697 meters
Maximum value	1000,9 meters

For all the cross section point values, that are known in the AHN. A difference value can be computed, in figure 4.4 part of the histogram of these values are shown. Because there are values that are much larger or smaller then the 5 and 95 percentile, the graph has been constrained to one meter difference with the AHN value. A histogram of the data is found in figure 4.4, because not all values could be represented well in the graph, statistics that represent the data are found in table 4.4:

Chapter 5

Discussion

To have a more complete picture of the data quality, more rules should be added. The current rules do not take into account the pre-processing that most of the models do. The model builders could add specific rules for a model that can be checking by the NHI. Users of the data have no direct way of communicating the lack of quality back to the data provider.

Uniformity should be an important goal to reduce the pre-processing.

To improve the knowledge what data quality problems are important to improve upon, the sensitivity of the models to their input should be investigated. If the data quality is poor of a data set, but the models that are made with the data are not sensitive to these quality problems, focus can be put on other data sets. If there are models are are very sensitive to small changes in values from specific pieces of data in the NHI, even a small error might be to much.

The search for rules to evaluate the data quality has succeeded and a number of rules have been implemented. Many more can be implemented, but the results show that the implementation works. The Python, PostGIS and GeoServer combination is easy to work with, and new rules can be added in a matter of minutes.

These errors are however not complete. To find every error condition in the database of the NHI, every use of every model would have to be known. Also, when time passes, some models can become defunct and new uses of the data may be found. The definition of a single error is easily defined when knowing the using party/model. This knowledge of the use cases and the parties using the data should be increased. The better the communication about how the actors use the data, the easier it will be to define errors.

As the data in the NHI database is moved to be open data, there will be users that are not part of a organisation supporting the NHI. Not only for the finding of errors, but also for the support (monetary or in kind), a dialog between users should be encouraged.

Chapter 6

Conclusions & Recommendations

6.1 Conclusions

> "How can the data quality of the HyDAMO database be improved?" (1)

When looking at the ratios in table 4.3, the data quality between the data objects differs quite a bit. On the one hand, weirs and pumps with a high ratio, they seem to be placed in the correct location, and low ratios for hydroobjects missing their cross sections. These differences in ratios not only differ by rule, they also differ by waterboard. Rule 1703 that requires the roughness's to be valid, has a seemingly reasonable ratio, but when looking at the waterboards, they or have it all correct or all wrong. So what happened here? The data that needed to be input where the roughness values to calculate flow. The used equations are formulated in a way that a low value means a high roughness. If the employees inputting the data don't have knowledge about this particular usage of hydraulic equation, the mistake to input a high value for a high roughness is easy to make. So for this rule, the data quality could be vastly improved by flipping the entries for the waterboards that have inputted them the wrong way around.

> "What is defined as an error?" (1.1)

For different use cases, different errors can be defined. In the results section, a list of errors and their definition can be found.

> "Can rules be defined to improve the data quality of the NHI database?" (1.2)

From the interviews in the appendix, the INSPIRE data recommendations and by

using logic (bridge underside must be lower than topside), rules have been distilled and are presented in the results. To use these errors to improve the data quality, the waterboards can use the suggestion layers. The rules that are defined in this report do not improve the data quality by themselves. When the objects are violating a rule, they should be looked at by the people imputing the data.

An example of a rule defined from an interview is the requirement of a cross section. In the interview with Joachim Hunink, a discrepancy in the data that was discussed is the need for a cross section to define the boundary conditions for the MODFLOW model. This was then more narrowly defined as "a hydroobject must have a cross section".

> "How can the data be presented to the users so they can correct the errors?" (2)

The data quality is presented to the user as a layer of the erroneous objects, with an error message attached to it. This layer is implemented as a view in the database of the NHI, and is published using GeoServer. Via Geoserver Waterboards can access the layers via QGIS or ARCGis. An example of the visualisation can be seen in figure 4.2. Here the selected cross section is visible in red, the other detected cross sections from the same layer in pink and the yellow dots are erroneous weirs. The selected erroneous cross section has three added attributes, a unique id, a suggestion and a suggestion code.

Meetings with Daniel Tollenaar, Timo Kroon and Gerry Roelofs (appendix C) revealed that waterboards want flexibility when quality checking. The reports on the data quality should not only be available in a central location (NHI server), but should also be available before uploading the data to the HyDAMO database. The consensus was also that the quality reports should not be available to the general public, but only to the waterboards that provide the data. At a later point in time, when the waterboards are more experienced with the data quality checks, this can reevaluated. Accessing the layers that contain the reports, is therefore implemented as a layer on a GeoServer that is behind a login. By putting the layers behind a login, only the user that has the username and login can access the data. Every waterboard has their own layers and login to view the suggestion layers on the NHI Geoserver.

The data quality is not good enough to reduce the pre-processing for models. While the HyDAMO data model has improved the data quality by standardising it, the lack of consistency in the data makes pre-processing necessary. The improvement of the data is left to the waterboards, but the rules reveal some clear errors, that can be fixed with the help of the rules. A example of this are the flipped roughness values for a few waterboards. With a few lines of code, this can be processed and improved.

The code written for this research implements the rule written in this report, and the results show that it is feasible to develop rules for data quality. The rules are clear, concise, and with the use of the edit rule definitions, easy to implement. They

have been evaluated over the data that was available in the HyDAMO database, and suggestions where added for every rule.

Through the addition of suggestions, the reason that an object is erroneous is clear and unambiguous. This will help the employees at waterboards make a quick judgement if the data needs to be improved.

6.2 Recommendations

The research showed that it is feasible to define global data quality rules. The qualification of errors was however not conclusive. To qualify an error in the correct way, knowledge of the real object is required. This knowledge lies with the the data recorders at the waterboards. It is very helpful for further research, if the editors that correct the errors, also record what qualification the error falls under. This way the rules might be improved.

When waterboards start to improve their data sets using these tools, focus should first be put on low hanging fruit. Looking at the percentile values in figure 4.4, there are many values that can easily be improved by removing erroneous values. This way, models made with the HyDAMO data do not have to filter erroneous outliers. To get a sense of which data should be improved first, more research can be done to define what values models are sensitive for. The most sensitive input variables should then be improved first. The suggestions are now available behind a login for the waterboards. In the future, when the waterboards are more confident with the data quality tools, the suggestions for improvement should be published openly. The consumers of the data can then decide for themselves what to do with this information. They don't heave to built their own pre-processing to filter the errors, but can use the suggestions to filter unwanted data.

Data quality is a topic where multiple organisations are working on. To prevent duplication of work, investments from the waterboards in data quality tools should be open sourced. The NHI GitHub is a obvious way to combine the efforts from multiple organisations.

Bibliography

[1] AHN. *AHN | De details van het Actueel Hoogtebestand Nederland*. URL: https://ahn.arcgisonline.nl/ahnviewer/.

[2] C. Batini and M. Scannapieca. *Data quality : Concepts, methodologies and techniques (Data-centric systems and applications)*. Berlin: Springer Berlin Heidelberg, 2006. DOI: 10.1007/3-540-33173-5.

[3] C. Batini and M. Scannapieco. *Data and Information Quality*. 2016. ISBN: 978-3-319-24104-3. DOI: 10.1007/978-3-319-24106-7. URL: http://link.springer.com/10.1007/978-3-319-24106-7.

[4] W. J. De Lange et al. "An operational, multi-scale, multi-model system for consensus-based, integrated water management and policy analysis: The Netherlands Hydrological Instrument". In: *Environmental Modelling & Software* 59 (Sept. 2014), pp. 98–108. DOI: 10.1016/j.envsoft.2014.05.009. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364815214001406.

[5] R. Rikken E. de Rooij. *erikderooij/nhi: Nederlands Hydrologisch Instrumentarium*. 2019. URL: https://github.com/erikderooij/nhi.

[6] ESRI. *What is a shapefile?* URL: http://desktop.arcgis.com/en/arcmap/10.3/manage-data/shapefiles/what-is-a-shapefile.htm.

[7] ESRI Nederland. *ArcGIS Desktop*. URL: https://www.esri.nl/producten/arcgis/desktop.

[8] European Commission. "Data Specification on Hydrography – Technical Guidelines". In: March (2014). URL: https://inspire.ec.europa.eu/Themes/116/2892.

[9] Forum Standaardisatie. *XSD*. URL: https://www.forumstandaardisatie.nl/standaard/xsd.

[10] Inc. Free Software Foundation. *GNU GPL License*. 2007. URL: https://www.gnu.org/licenses/gpl-3.0.en.html.

[11] GeoServer. *About - GeoServer*. URL: http://geoserver.org/about/.

[12] Het Kadaster. *Basisregistratie grootschalige topografie*. URL: https://zakelijk.kadaster.nl/bgt.

[13] Het Waterschapshuis. *DAMO Objectenhandboek*. 2019. URL: http://damo.hetwaterschapshuis.nl/.

[14] Y.U. Huh et al. "Data quality". In: *Information and Software Technology*. Data-Centric Systems and Applications 32.8 (1990), pp. 559–565. DOI: 10.1016/0950-5849(90)90146-I. URL: http://link.springer.com/10.1007/3-540-33173-5.

[15] T. Kroon. *Overzicht*. 2016. URL: http://www.nhi.nu/nl/files/2014/9398/5484/NHI_symposium_30_juni_2016_-_Timo.pdf.

[16] J. Morrison and H. Veregin. "Spatial Data Quality". In: *Manual of Geospatial Science and Technology, Second Edition*. CRC Press, June 2010, pp. 593–610. DOI: 10.1201/9781420087345-c30.

[17] Open Geospatial Consortium (OGC). *Geography Markup Language (GML)*. 2012. URL: http://www.opengeospatial.org/standards/gml.

[18] Open Source Geospatial Foundation. *Discover QGIS*. URL: https://qgis.org/en/site/about/index.html.

[19] PDOK. *pdok.nl*. 2019. URL: https://www.pdok.nl/introductie/-/article/actueel-hoogtebestand-nederland-ahn2-.

[20] PostGIS Steering Committee. *Spatial and Geographic Objects for PostgreSQL*. URL: https://postgis.net/.

[21] Rijksoverheid. *Open overheid | Digitale overheid | Rijksoverheid.nl*. URL: https://www.rijksoverheid.nl/onderwerpen/digitale-overheid/open-overheid.

[22] E. de Rooij. *HyDAMO Datamodel*. 2019. URL: https://github.com/erikderooij/nhi/blob/cf313d2e3e9f3110371d73e8c1a57ecc75f51634/datamodel/datamodel_v12.xlsx.

[23] W. Shi, P. Fisher, and M. F. Goodchild. *Spatial Data Quality*. CRC Press, Sept. 2002. ISBN: 9780429219610. DOI: 10.1201/b12657. URL: https://www.taylorfrancis.com/books/9781134514403.

[24] The Open Geospatial Consortium. *Web Feature Service | OGC*. URL: https://www.opengeospatial.org/standards/wfs.

[25] N. van der Zon. *Kwaliteitsdocument AHN-2*. Tech. rep. Actueel Hoogtebestand Nederland, 2013, pp. 1–30. URL: https://assets.amsterdam.nl/publish/pages/704401/kwaliteitsdocumentahn.pdf.

Appendix A

Meeting with Govert Verhoeven & Gerrit Hendriksen

Meeting time: 30 april 2019 11.00

The cross section, hydroobject, and culvert data is used by Govert Verhoeven in the D-Hydro and HyDAMO models. These data sets will also be used in other models, but the rules that follow are developed with those models in mind.

Other interested parties in the improvement of the data in the NHI, are Mark Hegnauer and Guus Rongen (HKV).

Some general remarks about the data consistency:

- The direction of the vector decides whether the flow is modelled positively or negatively.
- There are large differences in how waterboards input the GIS data into the database
- To make sure the rules are valid, their needs to be a check with the waterboards, on what there practises are. They might have made choices for reasons that are unknown to Deltares. Before talking to the waterboards, we need a solid list of examples and consistency checks.
- Not all fields of the records that should have been filled, are actually filled with data.
- There are many hydroobject that do not have a cross section associated with them in the database, but these cross sections might be known to the

waterboards. There may be 'standard' cross sections that the use for certain line segments.

- As a general rule, there are three categories of waterways that waterboards record. Main, secondary and tertiary. These tertiary waterways often don't have cross sections recorded.
- To save the errors, tables will be added to the NHI database. This will help with the reporting of the errors, backing them up (versioning), and accessing speed.

A.1 Rules to check

"Does a hydroobject have a cross section?"

When a cross section is not defined on a hydroobject, a surface water model cannot make use of the hydroobject. This rule could be checked by constructing a line through the points of the cross section, and then check if a hydroobject has an intersection with a cross section. For a part of the watercourses, standard cross sections can be defined to improve the number of hydroobjects with a cross section.

Edit rule: $hydroobject = dwarsprofiellijnen.geom \bigwedge hydroobject.geometrielijn = false$

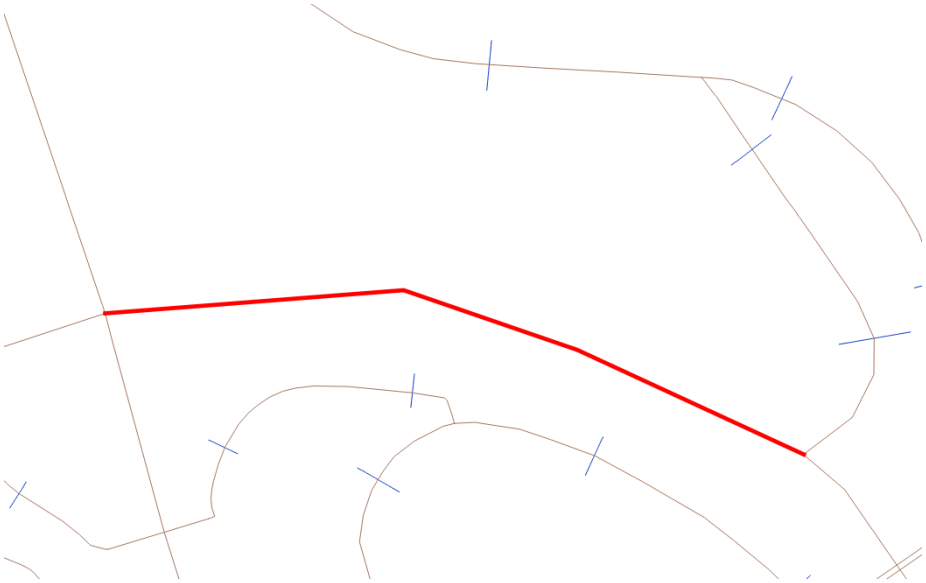


Figure A.1: Hydroobject without cross section

"Lines that do not have bends or intersections, should be defined as one line, not multiple."

The when the data is rasterised, multiple lines do not work well, these should be defined as one line. Multiple lines that are connected but do not have a difference in direction or properties should therefore always be one line. This could be checked by checking all lines that are singularly connected on one side, and have a line connected on that side, that has the same properties. To implement this rule in the correct way, the model builders should be involved, as they have a good understanding what works best with there models.

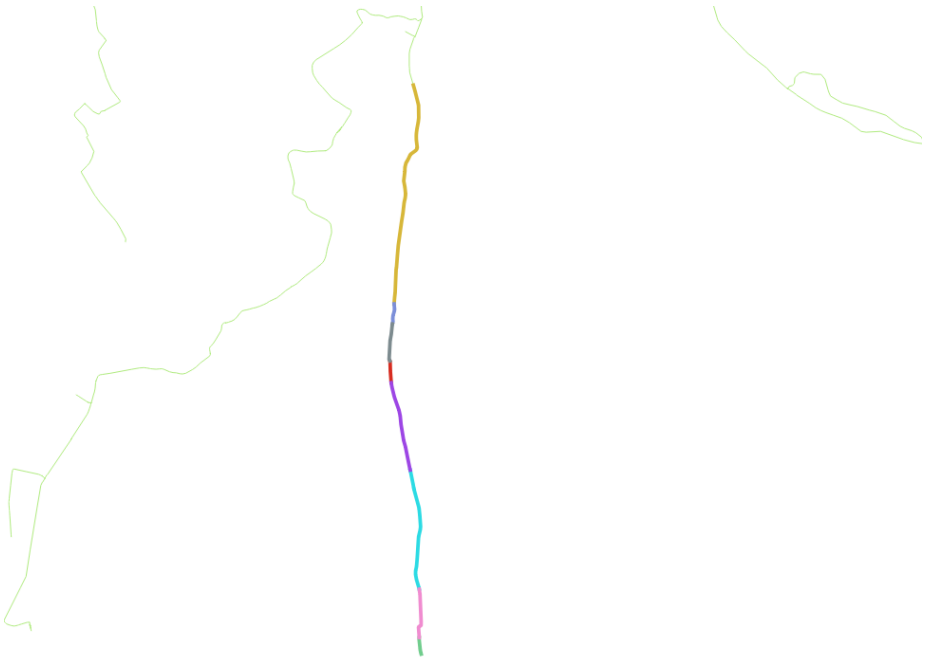


Figure A.2: Multiple segments for one watercourse

"Does a cross section have a hydroobject?"

If a cross section is defined somewhere, where there is no hydroobject, it can not be used by a model. This can be checked by the same rules as the first rule, but the check should change to first look at the cross section.

Edit rule: $dwarsprofiellijnen.geom = dwarsprofiellijnen.geom \wedge hydroobject.geometrielijn = false$

This is an population completeness error, there are record(s) missing in the dwarsprofiellijnen table.

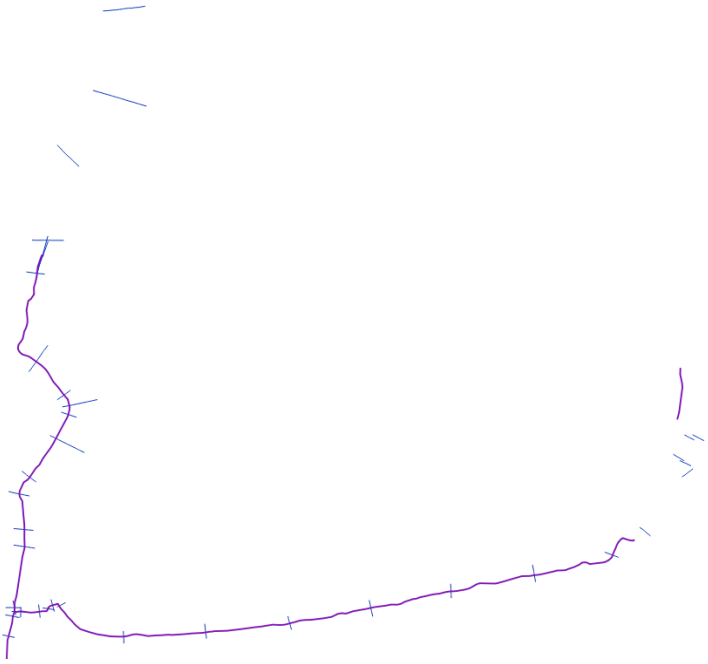


Figure A.3: Cross section without hydroobject

"Does a cross section intersect more then one hydroobject?"

The cross section can only belong to one hydroobject to be valid for the model, so it should have one intersection with a hydroobject. To check for this, the cross section can be checked if it has only one intersection.

Edit rule: $dwarsprofiellijnen.geom = dwarsprofiellijnen.geom \bigwedge hydroobject.geometrielijn > 1$

This can be a semantic accuracy issue, the cross section could be drawn larger then it actually is.

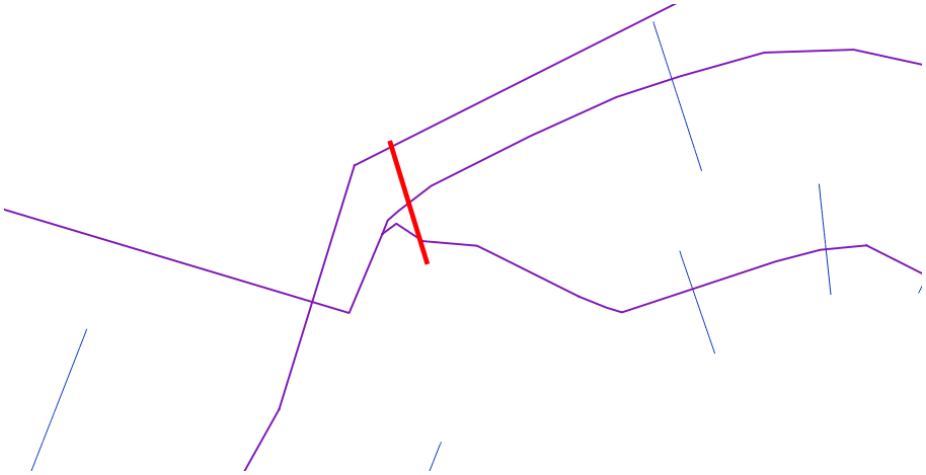


Figure A.4: Cross section with multiple watercourses

“Directions of a watercourse should face the same direction.”

The direction of the vectors of the line segments of a hydroobject, define the positive or negative flow in a model. The line segments should have the same direction, when having only one connection. This can be checked for by by looking at the vector direction of boundary connections between two line segments.

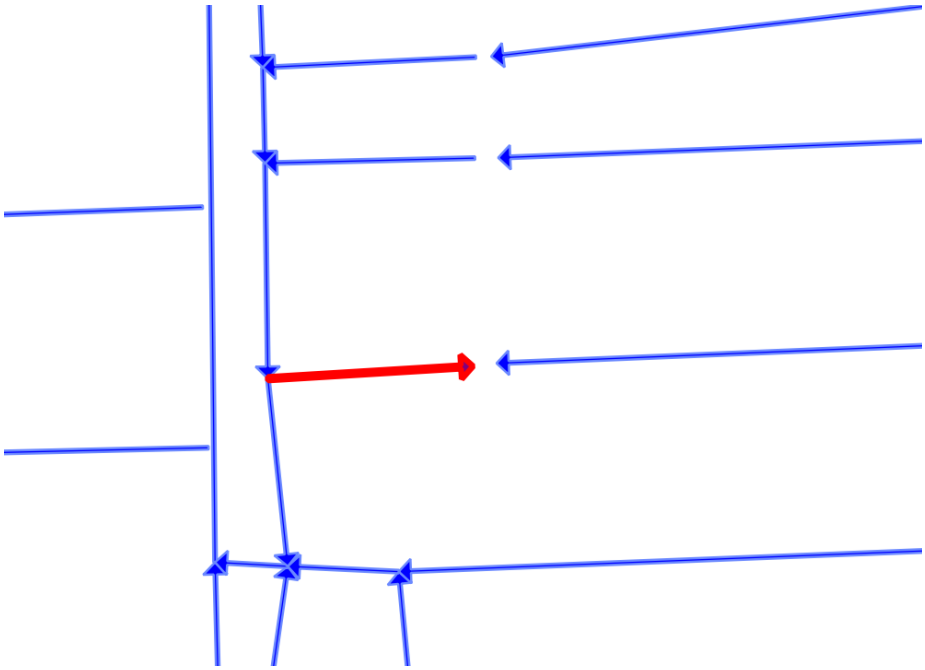


Figure A.5: hydroobject with wrong direction

"A culvert should lie on a hydroobject line."

The culvert is different from the line segments of the hydroobjects. Hydroobjects are combined with the culverts in the model, so they should lie close to each other, or on top of them.

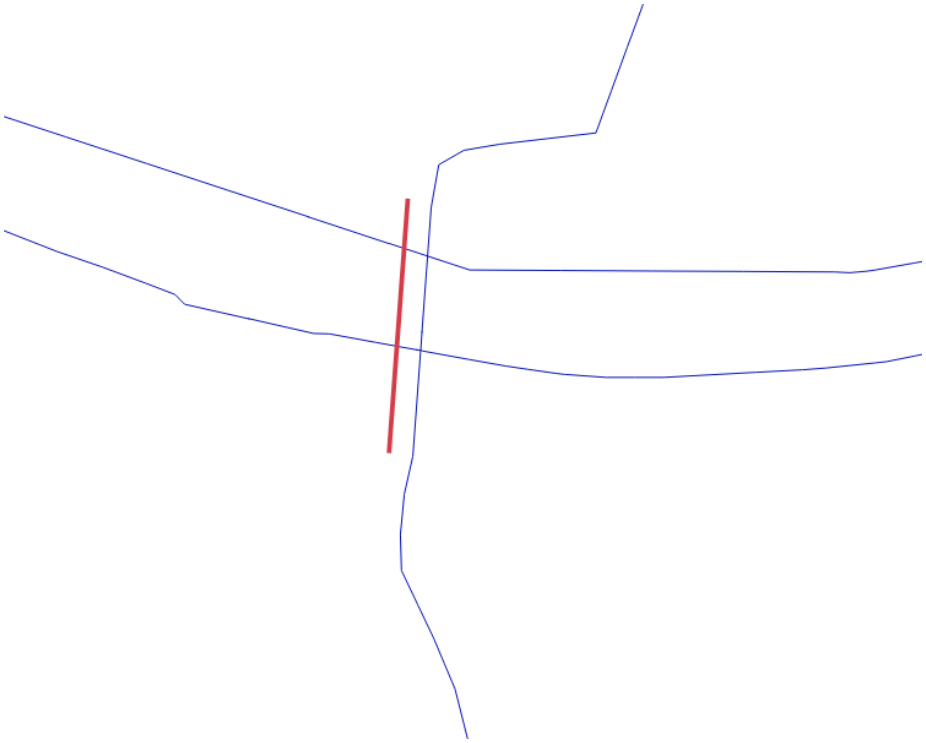


Figure A.6: Culvert that is not on hydroobject

"Do the artificial objects lie near a hydroobject?"

The object that regulates the watercourse, should lie near a hydroobject. Because the width of the line segments of the hydroobjects is not defined, the artificial objects could lie close, but not on the hydroobject. This could be checked by defining a buffer around a artificial object of a certain size that intersects a watercourse, or look for the nearest watercourse and then checking for the distance.

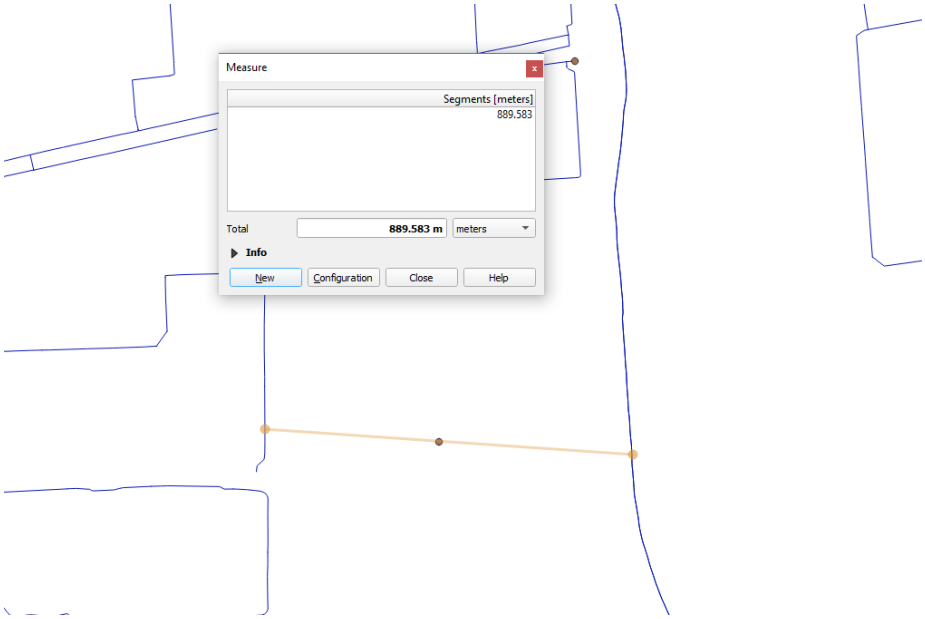


Figure A.7: Unconnected pumping station

"Are all the properties filled out and in valid ranges?"

When adding the data of the waterboards to the NHI database, not every property is checked. This can also be done in the context of this project. Rules can be added to check for empty property fields. Next to an empty check, validation on ranges can also be applied. For instance, the length and diameter of a culvert should not be lower then $1cm$.

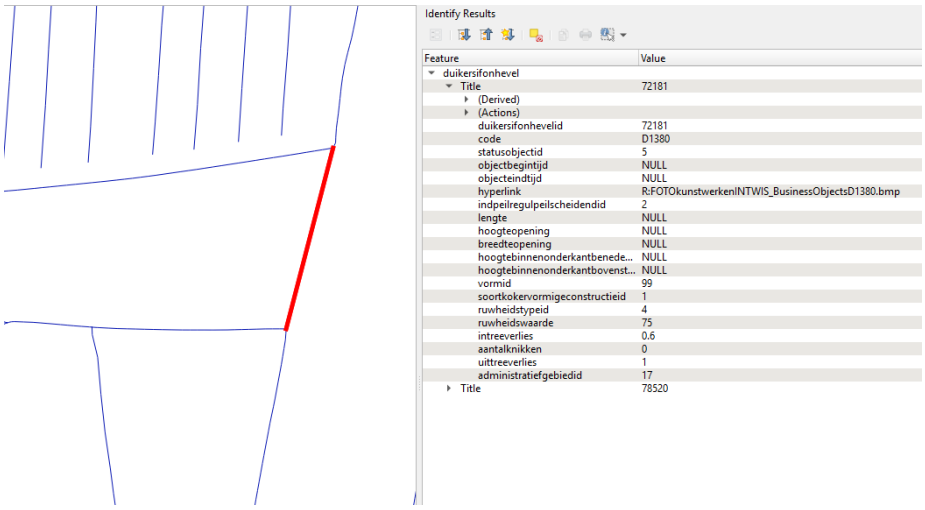


Figure A.8: Culvert with missing properties

A.2 Communication

The errors will not be corrected in the NHI database, but will be communicated to the waterboards. This way, the errors in the data will not return, because the source database is corrected. To make sure the waterboards are able to correct their mistakes, clear communication, preferably within the same tools that they are working with, is needed. A choice will have to be made if Deltares communicates the errors openly via the data portal and as extra tables in the PostGIS database, or that the communication of errors is more restricted.

The waterboards might also have more information on the tertiary watercourses that are not currently available in the database. To add the tertiary watercourse data, 'standard' watercourses could be used.

Appendix B

Meeting with Joachim Hunink & Gerrit Hendriksen

Meeting time: 17 may 2019 08.30

The models MIPWA and MODFLOW are both used by Joachim Hunink to model groundwater systems in the north east of the Netherlands. These models can use data from HyDAMO, to prescribe boundary conditions, and as model input. Other data used are the 'peilvakken', the water level above the ground level and water-course height.

- Some checks are already done on the data before use in the models.
- Manual detection of errors is not practical
- Data is restructured to a grid, that is usually coarser then the data.
- The raster for the data is often 25 by 25 meters, but can be any size.
- D-flow and SOBEK models are more sensitive to data quality, because they use more detailed data.
- 'Peilvakkaarten' are missing in HyDAMO

B.1 Pre-processing

Before the data can be input into a model, it first needs some pre-processing. Even when the data is standardised, different models like SOBEK and MODFLOW will need different types of simplification. As an example, SOBEK uses the watercourses as vectors, whereas MODFLOW will recalculate the the object into a raster. Next to these differences, models will always need to be a simplification of reality. So pre-processing will be needed even if the data is standard. When using the same

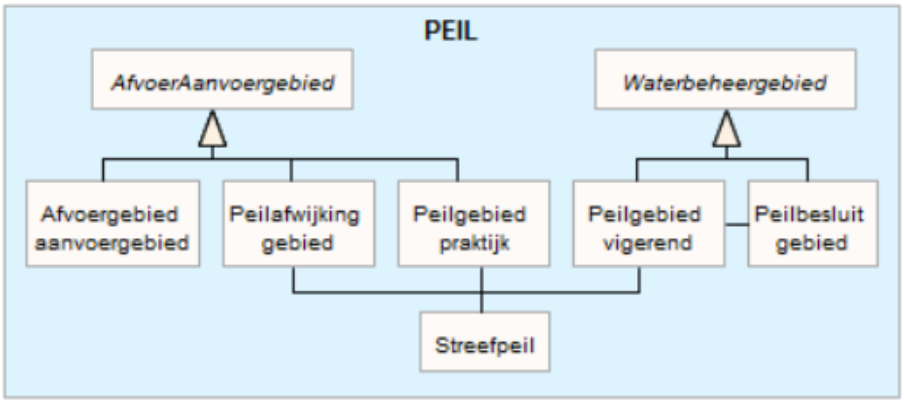


Figure B.1: DAMO waterlevel class diagrams

model for different waterboards, the pre-processing tools can be reused, when all the data from the waterboards, is standardised by the HyDAMO data model.

For the input of the model for the cross sections and watercourses, the AHN can also be used.

The pre-processing also removes errors in the data. Examples of these errors are decimal errors (points vs comma's), type errors (integers, floats), level of the cross sections, missing values that are in the data and are not signified with 'null' (999, 0). The waterlevel is also checked that it is below the ground level. Except for these examples, other errors often come up, and makes that the code that pre-processes the data needs to be rewritten for every waterboard where the MODFLOW model is implemented. A clear example is that different waterboards use different values for the no-data signifier (-9999, 0, null, etc.).

B.2 Schema completeness

During the meeting, the topic of missing data came up. For the MODFLOW and MIPWA models to work, more data types should be added to the database. In the context of data quality, this means that the data is not 'fit for use' and the quality dimension that is connected with this problem is schema completeness. In the meeting the objects that where mentioned where the water level objects from DAMO.

In the DAMO data model, there are some water level objects that could be added to HyDAMO, see figure B.1.

B.3 Modflow parameters

MODFLOW uses waterlevel, groundlevel, cross section resistance, whetted surface and more to model the surface water. Every cell in the grid has a whetted surface and a constant area. The infiltration area is hard to calculate, waterboards mostly use a map with the water supply in an area. The main water system (large rivers), is also used to calculate the infiltration, data on the main water system is much better than the local water systems. In the west of the Netherlands the waterlevel maps are best to use, in the east of the Netherlands, where there are no areas that are kept on one level, the watercourse is taken as a boundary condition.

- Peilvakkaarten
- Watercourses as boundary conditions
- Water supply from main watercourses (from Rijkswaterstaat)

Appendix C

Meeting with Gerry Roelofs (waterboard Rijn & Ijssel)

Roughness coefficient values for the Strickler the inverse of the actual roughness. This means a lower coefficient should be in the 'high' attribute, and the higher coefficient should be in the 'low' attribute. Some waterboards (about half of the current users of HyDAMO) have done the exact opposite. Where they have put the high absolute value of the coefficient in the high attribute. This is an easy error to make if one is not knowledgeable about how the Manning formula works.

For this research, use has been made of QGIS and a PostGIS database. In practise, most waterboards use ArcGIS with an Oracle spatial database. For tracking suggestions like is done in this research, ArcGIS has a review system. This review system is currently in use at the waterboard Rijn & Ijssel When publishing the suggestion, an important step towards integrating the suggestion with the review process of the waterboards can be, the transformation of the suggestion table from a PostGIS database format to the format that ArcGIS uses for the reviews.

The waterboards use FME tools to process a combination of datasources into the HyDAMO format. This means that a direct import/export from the databases will not work. The database that is used at many waterboards is the Oracle spatial database.

In most cases, waterboards would first like to check their data in-house, before uploading this to the HyDAMO online database.

Lateral knots should lie within the accompanying catchment area.

Although the waterboards are not expected to be accepting of the publishing of the suggestions online, there might be more support for the placing of comments online. These comments would be linked to the spatial object and would give users outside of waterboards the chance to signal errors.

The suggestions that are generated by the code produced for this report, would have the best chance of being used, when these are available online, but behind a login. This way the waterboards can easily review the suggestions on the data, but only the waterboards themselves would have access to the data.

To facilitate the checking by the waterboards, the code should be easily adjustable for parameters use in the checking. An example would be the range a weir can be away from a hydroobject before it is detected as too far away. The code should be well documented, and available in open source. Preferably hosted at a repository on GitHub. There is a NHI site where the code can be published.

In the future, facilitating of communication between data users and data consumers could be an added value of the NHI data portal. Adding comments to the

When handing in data for the BAG, a term for data that is not of sufficient quality is "gerede twijfel". This way of communicating to the waterboards that a record is erroneous, might go over better with them, because they are already used to this terminology.

When the suggestion is read by an employee of the waterboard that manages the data, they may conclude that the suggestion is not correct. To be able to exclude the object from the data rules, a flag should be able to be added to suggestions. This flag (true or false bit value) signals that the data object should not be again checked for the rule.

Appendix D

Additional errors

Next to the errors found via the rules, using the NHI database has also made some other issues come to light. This section consist of all the extra errors that do not fit into any other sections.

Unknown values in database

In the object model of the current HyDAMO implementation, unknown values are indicated with an often indicated with an integer value of 99. A better way to represent missing or unknown values in the database, is assigning them the 'null' value. As this explicitly states that there is no value, and doesn't depend on the object model. This way filtering of unknown values in HyDAMO can be as simple as checking for the null values in a column, any column. The error associated would be the syntactic accuracy, as the values should not fall within the accepted range of values in the object model.

Afvoercoëfficient not correct in database

In the object model of HyDAMO, the 'stuw afvoercoëfficiënt' is defined as a double, but in the database the column is defined as a integer. The value is mandatory, but there is not always a value available. The value -9999 is used as a indicator that a value is missing, but this should be 'null'.

```
<simpleType name="vormCodeSimpleType">
        <restriction base="integer">
                <enumeration value="1"/><!--rond -->
                <enumeration value="2"/><!--driehoekig -->
                <enumeration value="3"/><!--rechthoekig -->
                <enumeration value="4"/><!--eivormig -->
                <enumeration value="5"/><!--ellipsvormig -->
                <enumeration value="6"/><!--paraboolvormig -->
                <enumeration value="7"/><!--trapeziumvormig-->
                <enumeration value="8"/><!--heulprofiel-->
                <enumeration value="9"/><!--muilprofiel-->
                <enumeration value="10"/><!--langwerpig-->
                <enumeration value="11"/><!--scherp-->
                <enumeration value="99"/><!--onbekend-->
        </restriction>
```

Figure D.1: Enumeration integer used as unknown value.

```
        on delete cascade,
laagstedoorstroombreedte double precision,
hoogstedoorstroombreedte double precision,
kruinbreedte             double precision,
laagstedoorstroomhoogte  double precision,
hoogstedoorstroomhoogte  double precision,
afvoercoefficient        integer not null,
administratiefgebiedid   integer not null
    constraint stuw_administratiefgebiedid_fkey
        references administratiefgebied (administratiefgebiedid)
        on delete cascade,
```

Figure D.2: Afvoercoëfficiënt defined as integer in the database.

QGIS

For layers that are constructed with a database view in Postgresql, there needs to be a unique key selected. If the layer is added via the directory browser it doesn't work. The data source manager must be used to select a unique feature id. Because a single feature can have multiple errors, the primary key from the nhi database table is not unique. To fix this, the id of the join table is added to make sure every feature has a unique id. This then has to be selected in the data manager, so the layer can be added to the QGIS view.

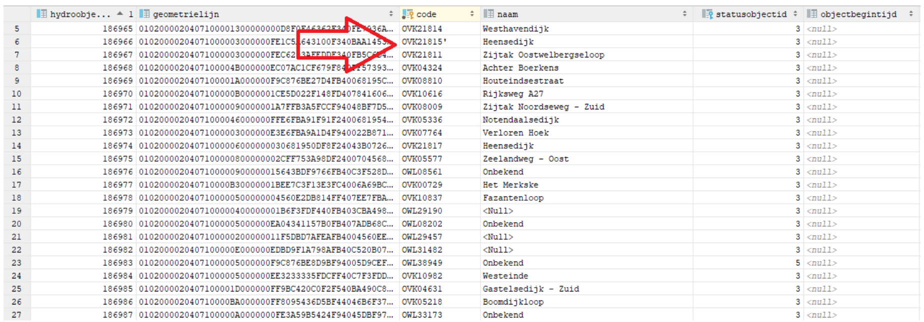
Hydroobjects

In the table of the hydroobjects, there are some inconsistencies or schema errors, that should be addressed. As a general remark; the primary key should probably be the 'hydroobjectid' column, as this column is a unique integer set. For the

'dwarsprofiel' table the primary id on the code column was apparently already a problem, because instead of a primary key on the code column alone, a composite primary key between the 'administratiefgebiedid' and 'code' was used. This would indicate that the codes are not unique between the waterboards.

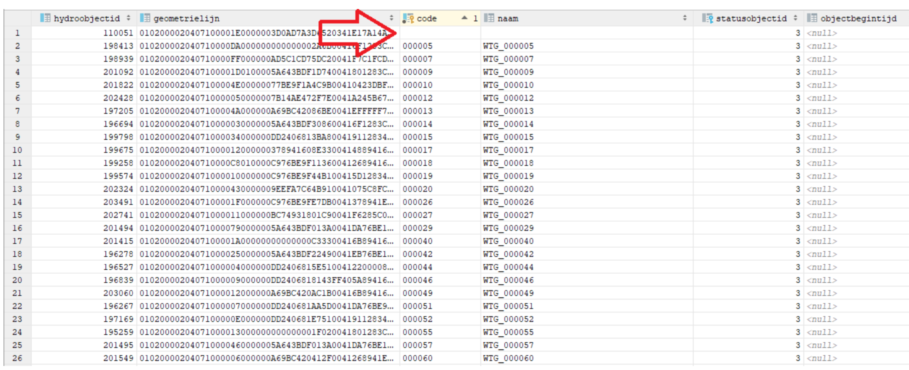
- Primary key on code
- Inconsistency of the code column

The following figures are examples of entries that should not be in the code column.



	hydroobje...	geometrielijn	code	naam	statusobjectid	objectbegintijd
5	186965	0102000020407100001300000000D8F0F46362F3[illegible]FE[illegible]936A...	OVK21814	Westhavendijk	3	<null>
6	186966	010200002040710000030000000FE1C5[illegible]643100F340BAA145[illegible]	OVK21815'	Heensedijk	3	<null>
7	186967	010200002040710000030000000FEC62[illegible]3AFEDDF340FB5C[illegible]4...	OVK21811	Zijtak Oostwelbergseloop	3	<null>
8	186968	0102000020407100004B000000EC07AC1CF679F8[illegible]F57393...	OVK04324	Achter Boerkens	3	<null>
9	186969	0102000020407100001A000000F9C876BE27D4FB40068195C...	OVK08810	Houteindsestraat	3	<null>
10	186970	0102000020407100000B0000001CE5D022F148FD407841606...	OVK10616	Rijksweg A27	3	<null>
11	186971	010200002040710000090000001A7FFB3A5FCCF94048BF7D5...	OVK08009	Zijtak Noordseweg - Zuid	3	<null>
12	186972	01020000204071000046000000FFE6FBA91F91F2400681954...	OVK05336	Notendaalsedijk	3	<null>
13	186973	01020000204071000003000000E3E6FBA9A1D4F940022B871...	OVK07764	Verloren Hoek	3	<null>
14	186974	0102000020407100000600000030681950DF8F24043B0726...	OVK21817	Heensedijk	3	<null>
15	186975	010200002040710000080000002CFF753A98DF2400704568...	OVK05577	Zeelandweg - Oost	3	<null>
16	186976	0102000020407100000900000015643BDF9766FB40C3F528D...	OWL08561	Onbekend	3	<null>
17	186977	0102000020407100000B30000001BEE7C3F13E3FC4006A69BC...	OVK00729	Het Merkske	3	<null>
18	186978	010200002040710000050000004560E2DB814FF407EE7FBA...	OVK10837	Fazantenloop	3	<null>
19	186979	010200002040710000040000001B6F3FDF440FB403CBA498...	OWL29190	<Null>	3	<null>
20	186980	01020000204071000005000000EA04341157B0FB407ADB68C...	OWL08202	Onbekend	3	<null>
21	186981	0102000020407100000200000011F5DBD7AFEAFB4004560EE...	OWL29457	<Null>	3	<null>
22	186982	0102000020407100000E000000EDBD9F1A798AFB40C520B07...	OWL31482	<Null>	3	<null>
23	186983	01020000204071000005000000F9C876BE8D9BF94005D9CEF...	OWL38949	Onbekend	5	<null>
24	186984	01020000204071000005000000EE3233335FDCFF40C7F3FDD...	OVK10982	Westeinde	3	<null>
25	186985	0102000020407100001D000000FF9BC420C0F2F540BA490C8...	OVK04631	Gastelsedijk - Zuid	3	<null>
26	186986	010200002040710000BA000000FF8095436D5BF44046B6F37...	OVK05218	Boomdijkloop	3	<null>
27	186987	0102000020407100000A0000000FE3A59B5424F94045DBF97...	OWL33173	Onbekend	3	<null>

Figure D.3: Unescaped quotation mark in the code column.



	hydroobjectid	geometrielijn	code	naam	statusobjectid	objectbegintijd
1	110051	0102000020407100001E0000003D0AD7A3D[illegible]520341E17A14A...			3	<null>
2	198413	010200002040710000DA0000000000000002A[illegible]F12[illegible]3C...	000005	WTG_000005	3	<null>
3	198939	010200002040710000FF000000AD5C1CD75DC20041F[illegible]C1FCD...	000007	WTG_000007	3	<null>
4	201092	0102000020407100001D0100005A643BDF1D740041801283C...	000009	WTG_000009	3	<null>
5	201822	0102000020407100004E00000077BE9F1A4C9B00410423DBF...	000010	WTG_000010	3	<null>
6	202428	010200002040710000050000007B14AE472F7E0041A245B67...	000012	WTG_000012	3	<null>
7	197205	0102000020407100004A000000A69BC42086BE0041EFFFFF7...	000013	WTG_000013	3	<null>
8	196694	010200002040710000030000005A643BDF308600416F1283C...	000014	WTG_000014	3	<null>
9	199798	01020000204071000034000000DD2406813BA800419112834...	000015	WTG_000015	3	<null>
10	199675	01020000204071000012000000378941608E3300414889416...	000017	WTG_000017	3	<null>
11	199258	010200002040710000C8010000C976BE9F11360041268941 6...	000018	WTG_000018	3	<null>
12	199574	01020000204071000010000000C976BE9F44B100415D12834...	000019	WTG_000019	3	<null>
13	202324	010200002040710000430000009EEFA7C64B910041075C8FC...	000020	WTG_000020	3	<null>
14	203491	0102000020407100001F000000C976BE9FE7DB0041378941E...	000026	WTG_000026	3	<null>
15	202741	01020000204071000011000000BC74931801C90041F6285C0...	000027	WTG_000027	3	<null>
16	201494	010200002040710000790000005A643BDF013A0041DA76BE1...	000029	WTG_000029	3	<null>
17	201415	0102000020407100001A000000000000000C33300416B89416...	000040	WTG_000040	3	<null>
18	196278	010200002040710000250000005A643BDF22490041EB76BE1...	000042	WTG_000042	3	<null>
19	196527	01020000204071000004000000DD2406815E5100412200008...	000044	WTG_000044	3	<null>
20	196839	01020000204071000009000000DD24068181 43FF405A89416...	000046	WTG_000046	3	<null>
21	203060	01020000204071000012000000A69BC420AC1B00416B89416...	000049	WTG_000049	3	<null>
22	196267	01020000204071000007000000DD240681AA5D0041DA76BE9...	000051	WTG_000051	3	<null>
23	197169	0102000020407100000E000000DD240681E75100419112834...	000052	WTG_000052	3	<null>
24	195259	010200002040710000130000000000000001F020041801283C...	000055	WTG_000055	3	<null>
25	201495	010200002040710000460000005A643BDF013A0041DA76BE1...	000057	WTG_000057	3	<null>
26	201549	01020000204071000006000000A69BC420412F0041268941E...	000060	WTG_000060	3	<null>

Figure D.4: Empty string as entry of the code column.

Double entries

In the schema objects are checked if they appear only once in a table using the value of the code column. This is not the only way to check this, because the geometry can also be used to check for multiple entries of the same object. Between tables

there can also be a doubling of objects. An example of this can be seen in figure D.5. In the land area of the waterboard Stichtse Rijnlanden, all the pumps (pompen) and pumping stations (gemalen) are the same. In this case the

To check for the double input of features, the code can be used, but the usage of geometry will give better results. A code can be easily input wrong, or defined differently, but the geographic location can be easily checked on a map, and in the real world, only one object can occupy the same space.

Figure D.5: The same feature in two tables

Appendix E

Code implementation

In this section the implementation details of the code that sets up and runs the detection rules are described. First the structure of the code is presented. Second the way the rules are implemented in PostGIS and Python is described. Third the multi-threading approach to send multiple queries to the database via Python is described. Including this description is the explanation of how the interface with the AHN server works. All the code used in this research is available via the open source portal of Deltares, and has a GPL license [10], so can be used in any open source project. This way the research done can comply with the initiative of an open government [21].

E.1 Code structure

The rules for checking the errors, and the errors that are detected, are implemented in Python and PostGIS. The code is written to take advantage of classes, inheritance, abstract classes and threading. For grouping the functions that are used to check the data that should be checked are taken as a class. For example, the object that is checked for missing values is the 'gemaal' object. The function that is used to check it, can be found in the GemaalDetector class. As there are many objects, functions that are more generic can be reused. Here class inheritance comes in to play. When a function finds erroneous objects, it needs to save the error information into the database. The function that saves these objects, is the same for every detector, so this can be placed in the parent class of all the detectors. The parent of the detectors is called Detector. A few more general functions are contained in this class, and some need values of properties that the child class has. For example the table name of the child class needs to be known for the saving function to save the objects to the correct database. To make sure these values are implemented in the child class, abstract 'get' functions are used. This will force the child class to implement a function and value for the table.

Next to the detectors, there are a datasources. These datasources are used by the detectors to connect to the datasources. This can be the PostGIS database of the HyDAMO database, but also other webservices. Datasources should have a configuration class that provides the datasources with login information and configuration when needed. This way datasources can be reused across the NHI project.

The code has two procedural files (scripts), one to set up the database schema, views and tables for the error detection, and the script that manages the running of the detection. The setup file calls the detectors to create their tables and views. Last in the list is are the cross section lines generated from the dwarsprofielen. This is done because they take a long time to generate and are needed in multiple checks. These cross section lines are mostly the same as the dwarsprofiellijnen in the HyDAMO database, only an effort has been made to generate less faults in the lines. After the setup has been run, the detection of errors can begin.

The detect errors script calls all the detectors with their detection rules. To speed up the detection, every detector uses its own threads. For the AHN detection a separate function is used with its own queue.

E.2 Rules implementation

```
def check_if_object_has_cross_section(self):
    # Every hydroobject needs to have a cross section. So check
    # this with the intersection function if there are intersections
    # between the datasets
    with self.get_datasource().get_connection() as connection:
        results = connection.execute('''
            SELECT hydroobject.*, {quality_schema}.cross_section_lines.*
            FROM hydroobject
                LEFT JOIN {quality_schema}.cross_section_lines
                    ON st_intersects(
                        hydroobject.geometrielijn,
                        {quality_schema}.cross_section_lines.cross_section
                    )
                WHERE {quality_schema}.cross_section_lines.profielcode IS NULL
        '''.format(quality_schema=self.get_quality_schema()))

    self.insert_error_records(results, self.get_cross_section_suggestion())
```

E.3 Multi-threading and connection to the AHN server

To be able to improve performance of the rules checking, the processing must be able to scale with an increase in cores. As in the last decade single core performance of cpus has not increased substantially, but the number of cores in a processor has. Usually computationally intensive task can be parallelised via programs as openMPI. And if the computational task is homogeneous, this is a good solution. In the case of the rules that are checked in this code, the task are very heterogeneous. Some rules need data from the AHN, some are bound by disk access and some are waiting for a database query to finish. To be able to complete these tasks in parallel, multi-threading must be used.

The Postgresql database can work in parallel when solving a query. When the database has to run multiple connections are made to it. Thus no effort has to be put in trying to make a query itself run parallel. To run multiple queries at once, multiple connections must be made to the database. When using a single thread, only a single connection can be made at a time. The execution would halt to wait for the result of a query. To make multiple connections and queries, multiple threads are needed. Every thread can make its own connection to the database, and does not have to wait for a query in another thread to finish.

For every rule a thread is started, this thread makes a connection to the database, and executes the query for that rule. When the results of the query are received, an insert query is executed, that adds all the erroneous records to the appropriate suggestions table in the database. This way, every query can be executed in parallel, and the queries do not have to wait for each other to finish.

The code makes use of the PDOK.nl webservice for the AHN height map. These height map are very data intensive, a map of the Netherlands has a size of 1,2 terabytes. To query this data efficiently, not all data should be accessed at once. Limiting the data usage and keeping the performance of the code acceptable has succeeded, but it required two iterations.

The first iteration was based on the fact that for the cross sections, only point values where needed. So instead of querying large sections of the height map, only the points on which a HyDAMO data point lies will be queried. An illustration how this works in practise is found in figure E.1. The points are queried from the HyDAMO database and then put in a queue for the threads to read from. When a thread gets a point from the queue, a request to the AHN web-service is made. The height value from the AHN height map and the point from the HyDAMO database is compared, and if they do not match, a suggestion is saved in the HyDAMO database.

The issue with this approach, is that the number of points is rather large (2,2 million) in the current database. This results in the AHN server to have to return small values very often, when the server is optimised for returning maps. Runtime for the code was 16 hours to check all the points, and the AHN server sometimes

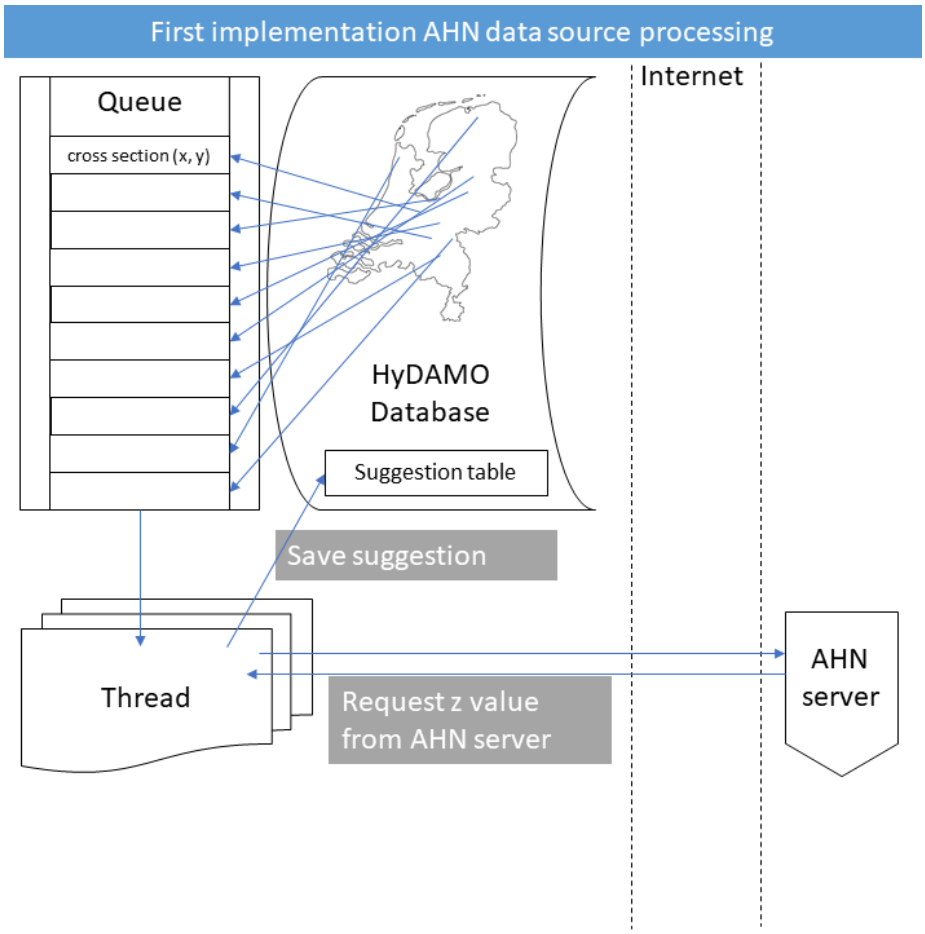


Figure E.1: AHN server data collection, version 1

blocked the requests. This could have been expected, because the way the server is accessed when asking for a large amount of points, might be analogues to a denial of service attack. This implementation was deemed to slow, and potentially wasteful for the AHN server resources.

The second iteration of the AHN connection code is done with the assumption that the AHN server is efficient at sending maps instead of points. First step is dividing the bounding box of the Netherlands into tiles, that can be queried to the HyDAMO database and the AHN server. The size of these tiles is 2 by 2 kilometres, because this is a constraint of the AHN server. The coordinated of these tiles are put into a queue, and the threads take a queue entry. The thread then checks the HyDAMO database if there are any points on the tile. If there are no points on the tile, the tile is skipped. If it has points, the tile is downloaded, and if required saved to disk.

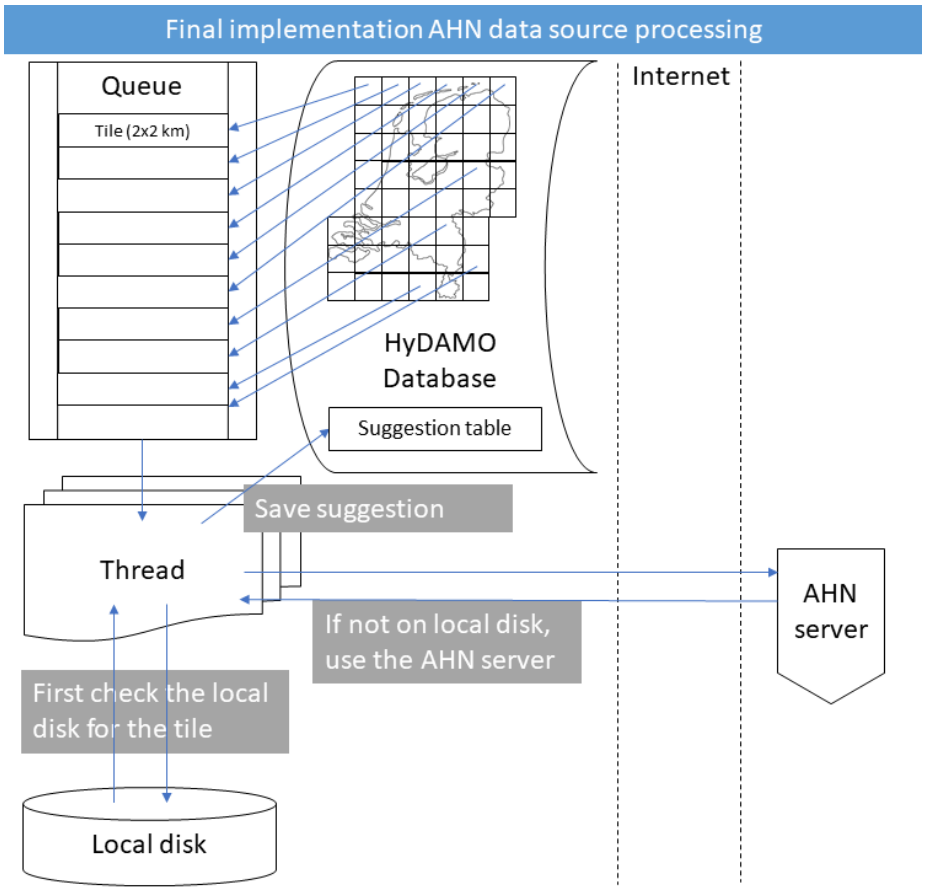


Figure E.2: AHN server data collection, version 2

This disk saved tile can then be used in another run instead of the AHN server to speed up the checking of the rules. The thread then checks all the points that are on the tile and saves the suggestions to the HyDAMO database. A graph of the code can be found in figure E.2. This implementations runtime was 4 hours when using the AHN server, and 10 minutes when the files are downloaded and used from a local harddisk. As an added benefit, because the code does not have to connect to the server for every point, the scaling of this solution is better.

Appendix F

INSPIRE data quality elements

The INSPIRE Directive aims to create a European Union spatial data infrastructure for the purposes of EU environmental policies and policies or activities which may have an impact on the environment. This European Spatial Data Infrastructure will enable the sharing of environmental spatial information among public sector organisations, facilitate public access to spatial information across Europe and assist in policy-making across boundaries. INSPIRE is based on the infrastructures for spatial information established and operated by the Member States of the European Union. The Directive addresses 34 spatial data themes needed for environmental applications [8]. A number of rules for data quality are defined in the hydrography specification of the INSPIRE directive.

In table F.1 a short version of the rules that are in the INSPIRE documentation is presented. Some of the rules have been directly implemented already, so a column of that matches the INSPIRE rules with the rules found within this paper, is added. Rules like the undershoots and overshoots, and geometric accuracy in general, are not straightforward to automate. Relating geometric objects to each other does not give a conclusive result on what the the reason for an under or overshoot. Which one of the objects is in the wrong place, is not easily detected, also, both object placements can be wrong. For these rules in the INSPIRE rule catalogue, more research is needed to build an implementation that uses multiple sources of data, to try and build a complete picture of which object is in the wrong place. Sources like the height maps, databases from different sources and satellite imagery could be used to automate the checking of correct placement off these features.

Table F.1: INSPIRE data quality rules

Id	NHI code	Name	Rule definition
3	-	Rate of excess items	Number of excess items in the dataset in relation to the number of items that should have been present.
7	1301, 1701	Rate of missing items	Number of missing items in the dataset in relation to the number of items that should have been present.
10		Number of items not compliant with the rules of the conceptual schema	Count of all items in the dataset that are not compliant with the rules of the conceptual schema
11	1001	Number of invalid overlaps of surfaces	Total number of Erroneous overlaps within the data.
17	1401	Value domain conformance rate	Number of items in the dataset that are in conformance with their value domain in relation to the total number of items in the dataset.
21	-	Number of faulty point-curve connections	Number of faulty point-curve connections in the dataset.
23	-	Number of missing connections due to undershoots	Count of items in the dataset that are mismatched due to undershoots, given the parameter Connectivity tolerance.
24	-	Number of missing connections due to overshoots	Count of items in the dataset that are mismatched due to overshoots, given the parameter Connectivity tolerance.
26		Number of invalid self-intersect errors	Count of all items in the data that illegally intersect with themselves.
27		Number of invalid self-overlap errors	Count of all items in the data that illegally self overlap.
-		Number of watercourse links below threshold length	Count of all watercourse link items in the data that are below the threshold length.
-		Number of closed watercourse links	Count of all watercourse link items in the data that are closed.
-		Number of multi-part watercourse links	Count of all watercourse link items in the data that are composed of multi-parts.
28		Mean value of positional uncertainties (1D, 2D)	Mean value of the positional uncertainties for a set of positions where the positional uncertainties are defined as the distance between a measured position and what is considered as the corresponding true position.
53		Relative horizontal error	Evaluation of the random errors in the horizontal position of one feature to another in the same dataset or on the same map/chart.
65		Number of incorrect attribute values	Total number of erroneous attribute values within the relevant part of the dataset.
71		Attribute value uncertainty at 95 % significance level	Half length of the interval defined by an upper and a lower limit, in which the true value for the quantitative attribute lies with probability 95 %.