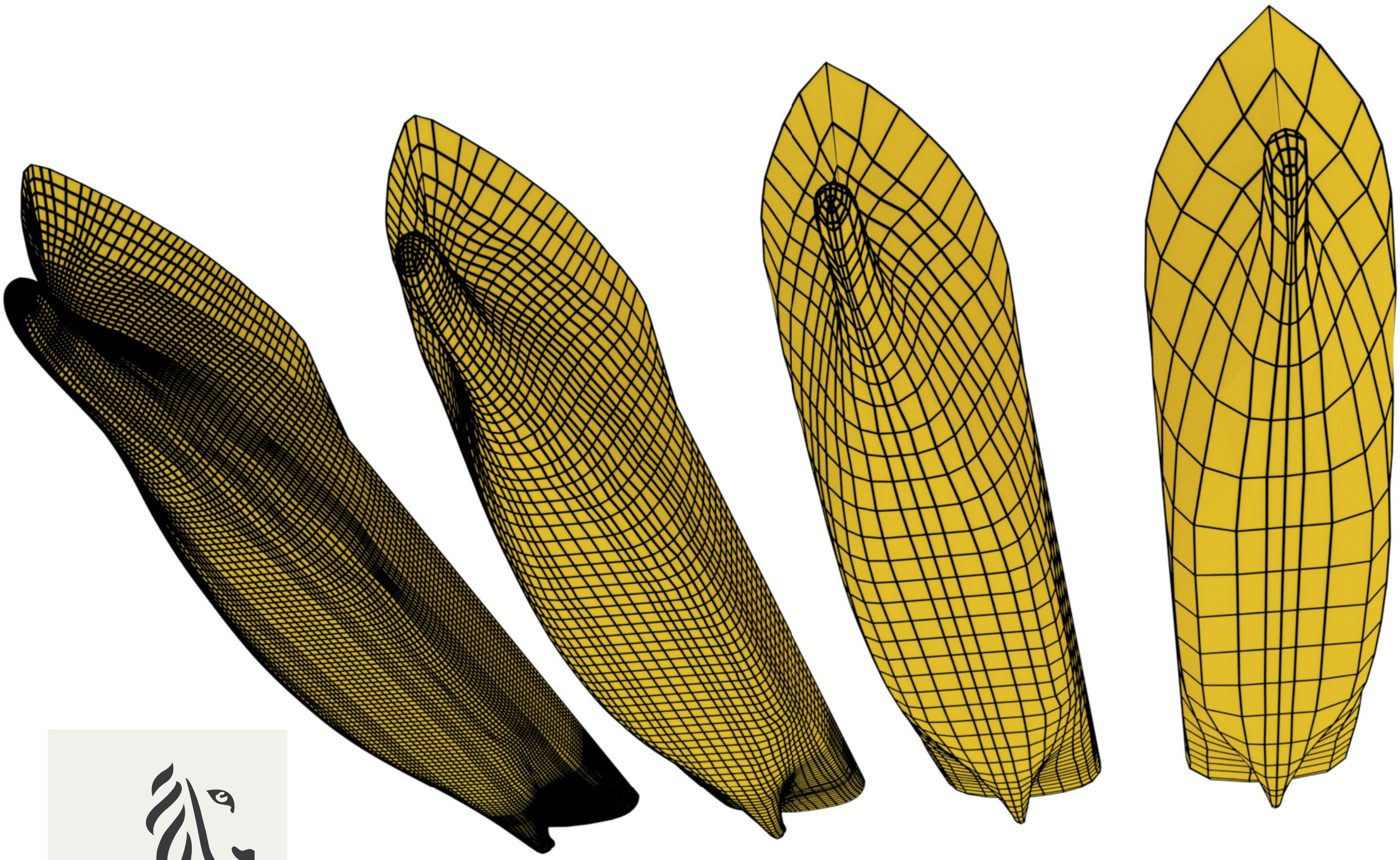


Flanders
State of
the Art

PA032_2
FHR reports

Beheer en Onderhoud CFD modellen

Implementation of a CFD CAD model database

DEPARTMENT
MOBILITY & PUBLIC WORKS

www.flandershydraulicsresearch.be

Beheer en Onderhoud CFD modellen

Implementation of a CFD CAD model database

Van Hoydonck, W.; Van Zwijnsvoorde, Th.; Lopez Castaño, S.; Villagómez, J.; Mostaert, F.

Flanders
Hydraulics Research

Flanders
State of the Art

Legal notice

Flanders Hydraulics Research is of the opinion that the information and positions in this report are substantiated by the available data and knowledge at the time of writing.

The positions taken in this report are those of Flanders Hydraulics Research and do not reflect necessarily the opinion of the Government of Flanders or any of its institutions.

Flanders Hydraulics Research nor any person or company acting on behalf of Flanders Hydraulics Research is responsible for any loss or damage arising from the use of the information in this report.

Copyright and citation

D/2020/3241/196

This publication should be cited as follows:

Van Hoydonck, W.; Van Zwijnsvoorde, Th.; Lopez Castaño, S.; Villagómez, J.; Mostaert, F. (2020). Beheer en Onderhoud CFD modellen: Implementation of a CFD CAD model database. Version 2.0. FHR Reports, PA032_2. Flanders Hydraulics Research: Antwerp

Document identification

Customer:	Flanders Hydraulics Research	Ref.:	WL2020RPA032_2
Keywords (3-5):	CFD, CAD, database, Confluence wiki, Subversion, Python		
Knowledge domains:	Harbours and waterways > Manoeuvring behaviour > Open Water > Numerical calculations		
Text (p.):	13	Appendices (p.):	3
Confidentiality:	No	☒ Available online	

Author(s):	Van Hoydonck, W.

Control

	Name	Signature
Revisor(s):	Van Zwijnsvoorde, Th.; Lopez Castaño, S.; Villagómez, J.	Thibaut Van Zwijnsvoorde (Signature) Digitally signed by Thibaut Van Zwijnsvoorde (Signature) Date: 2020.10.19 08:38:23 +02'00' Getekend door: Santiago Lopez Castaño Getekend op: 2020-10-19 12:51:12 +00:00 Reden: Ik keur dit document goed Santiago Lopez Castaño Getekend door:Jose Villagomez Rosales Getekend op:2020-12-15 16:15:28 +00:0 Reden:Ik keur dit document goed Jose Villagomez R.
Project leader:	Van Hoydonck, W.	Getekend door: Wim Van Hoydonck Getekend op: 2020-10-19 06:56:17 +00:00 Reden: Ik keur dit document goed Wim Van Hoydonck

Approval

Head of division:	Mostaert, F.	Getekend door: Frank Mostaert (Signature) Getekend op: 2020-10-19 07:01:56 +00:00 Reden: Ik keur dit document goed Frank Mostaert

CERTIFIED ISO 9001

Abstract

The objective of this report is to document the implementation of a system to store and document Computer Aided Design (CAD) models utilised in Computational Fluid Dynamics (CFD) software and potential panel methods at Flanders Hydraulics Research (FHR). The system makes use of available (software) facilities at FHR: the CAD data is added in the version control software (Subversion) and a visual front-end is created in the Confluence Wiki software. Topics discussed in this report are the storage of information about a specific CAD model, the process of updating the front-end pages on the wiki and retrieval of characteristics of the CAD models from other databases.

Contents

List of Figures

Nomenclature

Abbreviations

CAD	Computer Aided Design
CFD	Computational Fluid Dynamics
FHR	Flanders Hydraulics Research
NURBS	Non-Uniform Rational B-Spline
XML	eXtensible Markup Language
XSD	XML Schema Definition

1 Introduction

1.1 Background

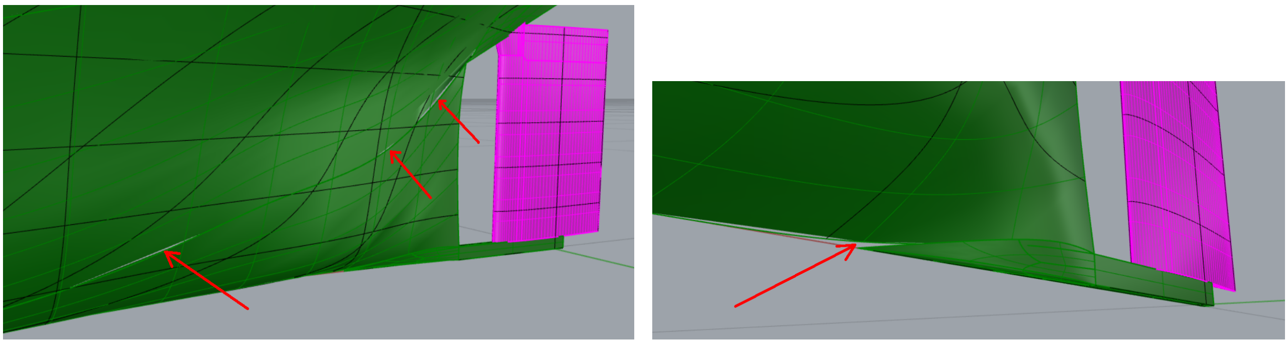
In the past, CAD files of scale model ships used in towing tank experiments at FHR have been used as a basis for CFD computations. These CAD files are stored in the lines plans section of the towing tank database here: `https://wlsow.vlaanderen.be/shpgenerator/Lists/Lijnenplannen/Forms/AllItems.aspx`. Currently, the CAD files are stored in `3DM` (Rhino 3D Model) format, whereas in the past formats such as `iges`, `dwg`, `fbm`, ...files have been used. The majority of internally created CAD models of scale ships have been created using Delftship[1]. As-is, the CAD files in the towing tank database are not suitable for CFD computations due to a combination of two requirements that are generally not met with the towing tank CAD files:

- the CAD geometry must be watertight;
- the CAD geometry should make use of a Non-Uniform Rational B-Spline (NURBS) formulation for the underlying geometry[2].

Internally, Delftship uses subdivision surfaces to model ship hulls. This technique is a generalisation of non-rational B-spline surface discretisation to arbitrary topology, which makes it significantly easier to create a watertight hull form. However, subdivision surfaces cannot represent circles (and the other conic sections) exactly unlike a NURBS formulation. This is one of the reasons why NURBS are used more than subdivision surfaces as a basis for file formats to exchange models between different CAD packages. Exporting a model from Delftship to IGES format converts the model from subdivision surface(s) to NURBS surfaces. If the topology of the original model does not allow for a direct conversion, the surface is split in patches and these patches are converted to NURBS format. If the NURBS curves at the connection do not have the same mathematical description (e.g. due to widely varying scales), gaps will be present.

As an example, Fig. 1 shows two close-ups of the bottom of the hull near the rudder of the CAD model of tanker `T0E`. Significant gaps are present between adjacent NURBS patches, which makes this model not suitable for CFD computation without extensive modifications. Another example of issues with the CAD models in the towing tank database is shown in Fig. 2, where the aft section of the hull of the inland vessel Myzako is shown. The propeller duct protrudes the hull, which makes this hull model (with appendages) not suitable for use in CFD computations.

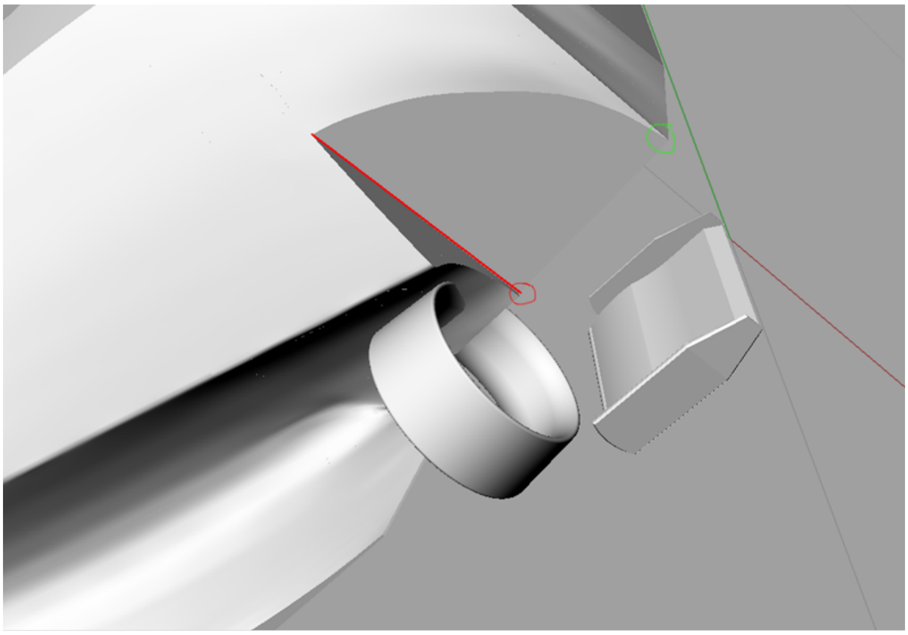
Figure 1 – Close-up of the bottom hull of tanker `T0E` near the rudder which shows significant gaps between adjacent NURBS patches.



[1] `https://www.delftship.net`

[2] This second requirement is not strictly necessary; very simple geometries consisting of flat surfaces can be represented accurately with a polygon format that uses triangular or quadrilateral faces such as `stl` or `obj`.

Figure 2 – Aft part of the hull of the Myzako that shows the duct of the propeller protruding the hull.



1.2 System requirements

The examples in the previous section show that without significant cleaning (which may take minutes to multiple days), the CAD models in the towing tank database are not suitable for use in CFD computations. Furthermore, there is no possibility to add a quality control system to the models in that database.

This document contains a proposal for a system for the long-term management and maintenance of CAD models for use in CFD computations. CAD models used in CFD studies are often reused between studies[3], whereas the complete setup including the domain box and solver settings are generally not reused.

The purpose of the proposed system is to create a database of validated CAD models that are directly usable in CFD computations. The following is a set of requirements for the system:

- the CAD models are stored in a central location under version control;
- for each CAD model, an xml file is added with information about the model, including its origin, available variants, relevant dimensions, remarks such as warnings and notes related to the quality, applicability of a model, source of the original information, etc.;
- for each CAD model, a wiki page is created using the data in version control and the information stored in the eXtensible Markup Language (XML) file;
- a CAD model can be available in multiple formats, with a conversion history that shows the relationship between the formats;
- CAD formats are subdivided in three categories `input`, `cfd` and `panel`;
- CAD models are grouped according to their application area;
- the database should be accessible for researchers outside FHR;
- the database should be searchable.

The last two requirements stem from the fact that the nautical research group cooperates with Ghent University in the Knowledge Center Manoeuvring in Shallow and Confined Water and that both parties have a partially overlapping collection of CAD models for use in numerical computations. Ideally, these collections should be merged in a single database that is accessible for both research groups. At the moment, the version control system in use at FHR is not accessible from outside the laboratory. However, in the future, an extern-

[3]This is also one of the reasons for the existence of various ship hulls such as the KVLCC2 and KCS for benchmarking purposes.

ally hosted repository (likely based on `git`[4]) might become available that will be accessible to external parties. The last requirement should make it easy to find a suitable (hull) model based on some characteristics once the database becomes very large. There are multiple possibilities to achieve this:

- (automatically) save relevant characteristics from the wiki pages into an excel sheet where advanced filters can be used to search a suitable hull model;
- collect the relevant characteristics in a table in a separate wiki page and use the standard table sorting features to find a suitable model[5].

These last two requirements are not discussed or solved in this report. The implementation of the other requirements is discussed in Chapter 2 while a short user guide is given in Appendix A1.

1.3 Terminology

The CAD model database is used for nautical and hydraulic applications. For these disparate research fields, a common set of terms has to be used that can be applied to both fields. The terms used in this report are *model* and *variant*.

When referring to nautical applications, the term *model* is used to refer to computer representations of a geometric shape. Such a shape (a hull or rudder for example) may have certain characteristic properties (such as a combination of a *length*, a *width* and a *height*) that are used to distinguish it from other similar shapes. Within a certain model, multiple *variants* may be present. These could pertain to scaled versions of the same shape, e.g. a model scale and full scale geometry of a specific container ship, but also to panel discretisations of the hull of a specific ship.

For hydraulic applications, *models* pertain to the geometry used in CFD computations of a specific project. Variants are either specific parts of the hydraulic structure (e.g. the downstream part of a fishway where the attraction current enters a river, or a set of individual basins of a complete fishway), or alternative geometries created to improve the performance of a hydraulic structure (e.g. adding rounded corners with guiding vanes to reduce flow separation in bends in pipes). A variant may need multiple files and the files themselves may be used in different variants.

[4] `https://git-scm.com`

[5] The *Table Filter and Charts for Confluence* app in the Atlassian Marketplace (`https://marketplace.atlassian.com/apps/27447/table-filter-and-charts-for-confluence`) would make this even easier.

2 Implementation

2.1 Revision control system

2.1.1 Location of the repository

At FHR, the agreed-upon revision control system in use for numerical work is Subversion[6]. The repository is accessible at `https://wl-subversion.vlaanderen.be`. One of the top-level repositories is dedicated to version control of numerical models (repoSpNumMod). The CFD CAD model database is located inside this repository at `https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/`.

2.1.2 Directory structure of the repository

Top-level directories are created for different types of geometry, roughly linked to the research groups at the laboratory that utilize CFD methods in their research. At the time of writing, these research groups are *nautica* and *kustwatcon*. For the former group, a second level of directories is added based on geometry type (such as hulls, propellers and rudders), while for the latter, it was opted that the second level of directories should be grouped based on project number. This means that over time, the number of directories for the second level of the hydraulic models will increase, while for nautical models, the third level of directories will gradually increase as more models are added. A third first-level directory is present (*coastal*) for (future) storage of CAD geometry related to coastal research. At the time of writing, the following structure for the SVN repository is present:

```
├── nautical
│   ├── hulls
│   ├── propellers
│   ├── rudders
│   ├── terrain
│   └── other
├── hydraulic
│   ├── 14_050_onderzoek_breekbalken
│   └── 18_134_Dessel-Schoten
└── coastal
```

2.1.3 Directory structure of a CAD model

Inside the CAD model directory, one `xml` file is present with all information about the geometry: `info.xml`. The CAD files themselves are stored in subfolders, where each type is stored in a separate folder. This is necessary because certain CAD file formats store their data in more than one file. One such example is the color STL format used by FINE/Marine: next to the STL file, a separate file encodes colours for each triangle present in the STL file.

CAD files for use in (potential) panel methods are stored in the `panel` folder while the CAD files for CFD applications are stored in the `cfd` folder. Both input and intermediate CAD files (such as `blend` files used to create a

[6] `https://www.subversion.org`

panel discretisation from an accurate triangulation of a hull shape) are stored inside the `input` folder. Finally, the `vis` folder contains visualisations of the different variants.

Hence, the complete top-level directory structure of a CAD geometry contains the following four folders:

```
├── cfd
├── input
├── panel
└── vis
```

The `panel` directory is mainly used in nautical applications and may not be present with hydraulic models.

Inside these top-level model folders, CAD files are stored per filetype in separate subfolders. For example, the `cfd` folder may contain two subfolders (`parasolid` and `stl`) that store geometry in these file formats. It may also happen that CAD models in a certain file format are present in two folders. A list of model types with a short description is shown below:

stl stereolithograhy: triangular discretisation of geometries;

blend native Blender file format, often used to create panel representations from accurate STL triangulations;

obj Wavefront geometry definition file similar to STL, but allows for faces with arbitrary vertices;

parasolid native file format for the Parasolid geometry kernel developed by Siemens;

iges standardized file format for the exchange of 2D and 3D drawing (surface) data between CAD software packages;

step conceived as successor of IGES file format that can store both surface and solid model geometry;

rhino native files of the CAD modelling software Rhinoceros 3D, for which FHR has commercial licenses;

hydrostar native panel format of Bureau Veritas' HydroSTAR potential hydrodynamics software used at FHR;

ropes native panel format of ROPES, a potential hydrodynamics software package;

2.2 xml file format

Initially, the workflow to add a new CAD model to the database was to collect the different CAD files in the folders, create some visualisations and commit that data to the version control system. Thereafter, the wiki page was manually created. This however proved to be a very tedious and error-prone process due to the addition of a table with hyperlinks to the different CAD files. Initially, this was solved by creating a template page that defined the structure of the wiki page for the user with some variables (such as the base location of the model in Subversion) that could be filled in. This reduced the time to create a wiki page but still required the user to manually construct hyperlinks to files.

A solution was found in the `atlassian-python-api` Python package that contains a simple interface for interacting with Atlassian software products based on the official public REST API[7]. Amongst other things, this package allows the creation and modification of wiki pages directly from Python, which opens the way to completely automate the wiki page creation process by storing the data for the wiki page in a simple XML file. A Python script was developed that parses the XML file, scans the model folder and based on that data, constructs a complete wiki page including hyperlinks to the CAD files, other related CAD models and possibly external sources. This reduces the time spend by the user to filling in an XML file with a simple structure.

[7]See `https://github.com/atlassian-api/atlassian-python-api` for the source code and `https://pypi.org/project/atlassian-python-api/` for some examples of its usage.

To enable the user to check the validity of the XML format, an XML schema was created which defines the structure of the XML format.

In Listing 1, an overview is given of the the structure of the XML file. The top level element is called <cad_model_info>.
Then, there are nine first level elements:

<author> The author of the CAD models if not mentioned as an attribute to the <file> element.

<description> Contains a model name, a type and a reference.

<variants> Contains a list of <variant> elements each with a description of the variant. Optional subelements may contain characteristic dimensions (only implemented for nautical/hulls), discussed below. The <variant> elements require a name and description attribute and optionally a datasource attribute.

<remarks> Contains a list of remarks related to the model(s). Subelements can be of type <note>, <info>, <warning> and <tip> , where each subelement may have an variant attribute that contains the name of the variant as defined in the name attribute of the variant element. These four elements are supported by the wiki software used at FHR.

<origin> Contains a description of the origin of the CAD geometry, e.g. "own work".

<projects> Contains a list of project elements with a number and name attribute in which the geometry was used.

<references> Contains a list of (external) reference elements with a href attribute, e.g. a report.

<related_models> Contains a list of related_model elements with a href attribute that points to another geometry in the database.

<files> Contains a list of file elements with an href attribute that defines the relative path of the CAD file in the subversion repository. The file element can optionally contain a variant, name and date attribute.

The variants, remarks, projects, references, related_models and files elements can contain an unlimited number of subelements (including zero).

2.2.1 variant element

The variant element contains a name attribute with the name of the variant. This value is used in other places in the XML file as a reference (although at the moment they are not checked).

For hull types such as tanker or container, this element may contain a datasource attribute. If the value of this attribute equals wlsowMS, a connection is established with the wlsow sharepoint site that contains characteristic dimensions of scale models tested in the towing tank. The connection is made based on the directory name of the CAD geometry, not the value of the description element. If a hull with the same name exists, values for L_{pp}, L_{oa}, B and D are extracted and these will be shown on the wiki page[8]. If the value of the attribute starts with wlsowMS_s and ends in a number (e.g. wlsowMS_s75), then (scale model) values are also extracted, and they are multiplied with the trailing number for a variant of a different size[9] If the value of the datasource attribute does not fall into either category, no attempt is undertaken to extract data from the towing tank database and instead, subelements lpp, loa, beam and depth are queried for values.

[8] Note that the block coefficient is not available as a parameter of the hull shape in the towing tank database, that is only available for a specific loading condition, and then not even as part of a List so it cannot be extracted using shareplum.

[9] Note that at the moment, variants of the same model can only be scaled uniformly.

Listing 1 – Structure of the `info.xml` files.

```
<?xml version="1.0" encoding="utf-8"?>
<cad_model_info>
  <author>initials</author>
  <description type="modeltype" href="reference">model name</description>
  <variants>
    <variant name="variantname" comment="description" datasource="wlsowMS|wlsowMS_s<xx>">
    </variant>
    <variant name="variantname" comment="description" datasource="wlsowMS_s75">
    </variant>
  </variants>
  <remarks>
    <warning variant="variantname">description</warning>
    <warning variant="variantname">description</warning>
  </remarks>
  <origin>where it comes from</origin>
  <projects>
    <project number="xx_xxx" name="project name" />
  </projects>
  <references>
    <reference href="external reference">description</reference>
  </references>
  <related_models>
    <related_model href="absolute url to related model">related model name</related_model>
  </related_models>
  <files>
    <file href="relative path to cad file">description</file>
    <file href="relative path to cad file" variant="variantname" author="initials"
     ↪  date="YYYY-MM-DD">description</file>
    <file href="relative path to cad file" date="2020-02-04">description</file>
    <file href="relative path to cad file" variant="variantname"
     ↪  date="YYYY-MM-DD">description</file>
    <file href="relative path to cad file" variant="variantname"
     ↪  date="YYYY-MM-DD">description</file>
  </files>
</cad_model_info>
```

2.2.2 Example `info.xml` file for model `COD`

Listing 2 shows the `info.xml` file for the CAD model of the container ship `COD`. This CAD model contains a model scale bare hull shape and four full-scale panel geometries that can be used in Hydrostar and ROPES. For the variant with name `COD_panel_marc`, a warning is present related to the poor discretisation of the model. A note has been added related to the `COD_bare_hull` model. The CAD model originates from the towing tank database and the model has been used in the past in two projects (00_057 and 15_011). There is one model (`COD_rudder`) in the database that is related to this one. There are a total of 11 CAD files related to this model of which nine are directly linked to the variants. The other two are the original `iges` files and an intermediate `blend` file used to create the panel geometries.

Listing 2 – Example `info.xml` for the CAD model of container ship COD.

```
<?xml version="1.0" encoding="utf-8"?>
<cad_model_info>
  <author>WVH</author>
  <description type="containership"
  ↪ href="https://wlsow.vlaanderen.be/shpgenerator/Lists/Prototypeschepen/AllItems.aspx"> Dionysia
  ↪ </description>
  <variants>
    <variant name="COD_bare_hull" comment="model scale bare hull geometry for CFD computations"
     ↪ datasource="wlsowMS">
    </variant>
    <variant name="COD_panel_coarse" comment="full scale coarse panel geometry for potential
     ↪ computations" datasource="wlsowMS_s75">
    </variant>
    <variant name="COD_panel_medium" comment="full scale medium panel geometry for potential
     ↪ computations" datasource="wlsowMS_s75">
    </variant>
    <variant name="COD_panel_fine" comment="full scale fine panel geometry for potential
     ↪ computations" datasource="wlsowMS_s75">
    </variant>
    <variant name="COD_panel_marc" comment="Model of COD obtained from Marc Vantorre"
     ↪ datasource="wlsowMS_s75">
    </variant>
  </variants>
  <remarks>
    <note variant="COD_bare_hull">This is the cad model of the model scale tested in the towing
     ↪ tank.</note>
    <warning variant="COD_panel_marc">Very coarse discretisation near bow and stern and on hull
     ↪ bottom (not recommended for general use).</warning>
  </remarks>
  <origin>cad model towing tank database</origin>
  <projects>
    <project number="00_057" name="berekeningen met FINE/marine van CFD software" />
    <project number="15_011" name="evaluatie ROPES" />
  </projects>
  <references>
    <reference href="https://wlsow.vlaanderen.be/shpgenerator/Lists/Prototypeschepen/AllItems.aspx">
     ↪ COD </reference>
  </references>
  <related_models>
    <related_model href="http://wlwiki.vlaanderen.be/wiki/display/wlwiki/rudder+COD"> rudder of COD
     ↪ </related_model>
  </related_models>
  <files>
    <file href="./input/iges/MS_COD_T200v3.igs">Original model used in the towing tank, iges is
     ↪ missing from the 'lijnenplannen' list</file>
    <file href="./cfd/parasolid/MS_COD_T200v3_cf_180912_1_cf.x_t" variant="COD_bare_hull"
     ↪ author="MLO" date="2013-01-29">model created from iges file in cadfix</file>
    <file href="./input/blend/COD.blend" date="2020-02-04">contains the stl triangulation and the
     ↪ panel geometry</file>
    <file href="./panel/ropes/COD_coarse.dat" variant="COD_panel_coarse" date="2019-03-27">coarse
     ↪ version of the panel geometry</file>
    <file href="./panel/ropes/COD_medium.dat" variant="COD_panel_medium" date="2019-03-27">medium
     ↪ version of the panel geometry</file>
    <file href="./panel/ropes/COD_fine.dat" variant="COD_panel_fine" date="2019-03-27">fine version
     ↪ of the panel geometry</file>
    <file href="./panel/ropes/COD_marc.dat" variant="COD_panel_marc" date="2019-03-27">coarse
     ↪ version obtained from Marc Vantorre</file>
    <file href="./panel/hydrostar/COD_coarse.hst" variant="COD_panel_coarse"
     ↪ date="2020-03-02">converted from panel/ropes/COD_coarse.dat via COD_coarse.obj</file>
    <file href="./panel/hydrostar/COD_medium.hst" variant="COD_panel_medium"
     ↪ date="2020-03-02">converted from panel/ropes/COD_medium.dat via COD_medium.obj</file>
    <file href="./panel/hydrostar/COD_fine.hst" variant="COD_panel_fine" date="2020-03-02">converted
     ↪ from panel/ropes/COD_fine.dat via COD_fine.obj</file>
    <file href="./panel/hydrostar/COD_marc.hst" variant="COD_panel_marc" date="2020-03-02">converted
     ↪ from panel/ropes/COD_marc.dat via COD_marc.obj</file>
  </files>
</cad_model_info>
```

2.3 Wiki pages

Using the data stored in the XML file and figures found in the `vis` subfolder of a CAD model, a wiki page is constructed with the Python script `create_wlwiki_page.py`. This script is stored in the top-level directory of the database (`https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/create_wlwiki_page.py`) and thus becomes available once a checkout of the complete repository is created (see Appendix A1.1). It is advised to run the script in a separate environment because some non-default packages are required. An environment file (*confluence_api.yml*) is included in the repository that contains a list of all packages in this environment.

When an XML file is parsed, it is first validated against the XML Schema Definition (XSD) schema definition. This way, the user quickly knows whether or not the XML syntax is correct. The XSD schema file is also added to the repository: if the syntax of the XML file is changed, it can be updated accordingly. At the time of writing this report, the Python script uses the XSD file in the checkout in the project folder on the P-drive.

Apart from presenting the data stored in the XML file, the wiki pages will also display visualisations of the different variants if they are found in the `vis` subfolder of a CAD model by the Python script. To make this work, the visualisation files must contain the `variant` names defined in the XML file in addition to a string that defines the view that is show. Supported strings are `ISO`, `front`, `side`, `top`, `bottom`, `back`, `dimensions`, `overview`, `detail` and `perspective` with an optional trailing number. Both parts should be separated with an underscore, hence: `variantname_ISO.png` and `variantname_ISO1.png` would be valid filenames. In addition, for Hydrostar geometry input files, the number of panels are displayed as well.

For Listing 2, screenshots of the resulting wiki page are shown in Figs. 3 to 6.

Figure 3 – Wiki page of the CAD model of C0D - model description and variant characteristics.

C0D

Created by Wim Van Hoydonck, last modified about 5 hours ago

Model description

Model name	C0D
Description	Dionysia
type	containership
SVN Location	nautical/hulls/C0D
Variants	• **C0D_bare_hull** (model scale bare hull geometry for CFD computations) • **C0D_panel_coarse** (full scale coarse panel geometry for potential computations) • **C0D_panel_medium** (full scale medium panel geometry for potential computations) • **C0D_panel_fine** (full scale fine panel geometry for potential computations) • **C0D_panel_marc** (Model of C0D obtained from Marc Vantorre)
Origin	cad model towing tank database
Project(s)	• 00_057: berekeningen met FINE/marine van CFD software • 15_011: evaluatie ROPES
References	• C0D
Related models	• rudder of C0D

Variant characteristics

Variant	lpp	loa	beam	depth
C0D_bare_hull	3.864	4.02	0.54	0.304
C0D_panel_coarse	289.8	301.49999999999994	40.5	22.8
C0D_panel_medium	289.8	301.49999999999994	40.5	22.8
C0D_panel_fine	289.8	301.49999999999994	40.5	22.8
C0D_panel_marc	289.8	301.49999999999994	40.5	22.8

Figure 4 – Wiki page of the CAD model of `COD` - remarks and visualisations of variant `COD_bare_hull`.

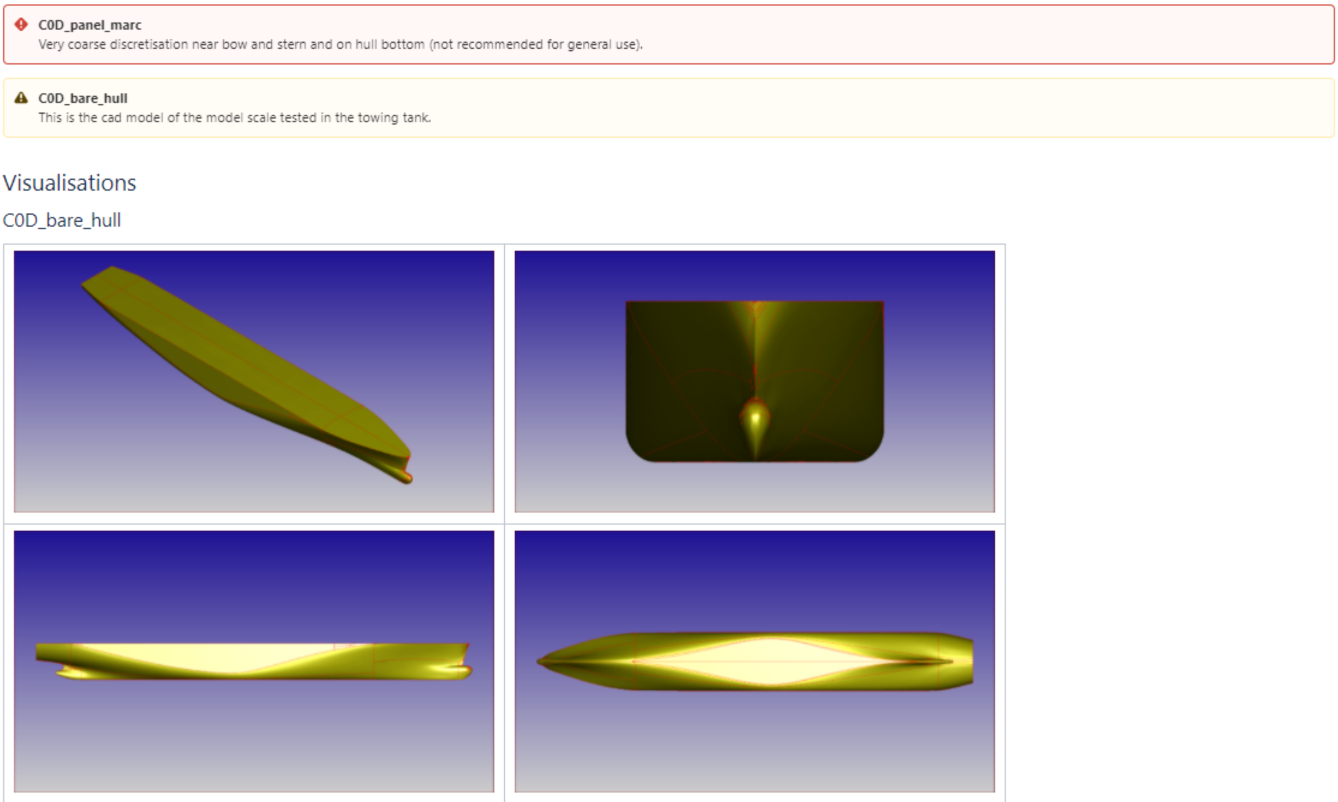


Figure 5 – Wiki page of the CAD model of `COD` - visualisations of three panel geometries.

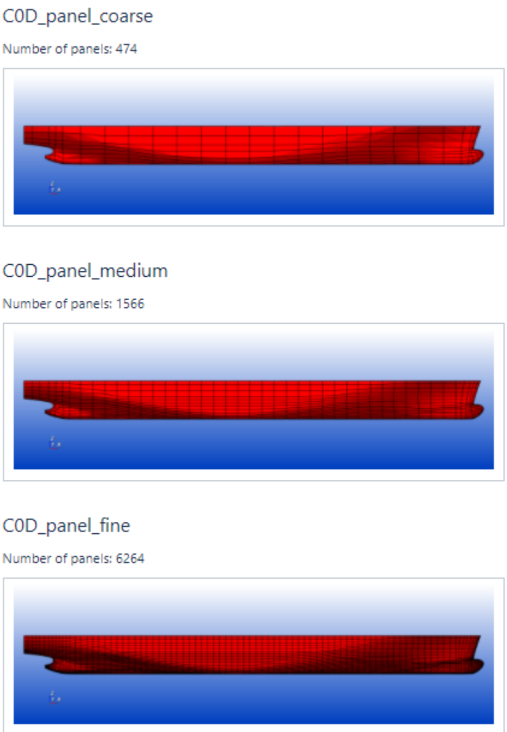
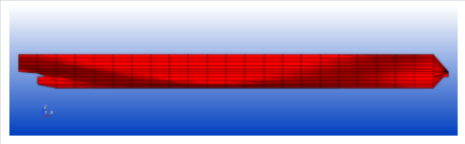


Figure 6 – Wiki page of the CAD model of C0D - visualisations of last panel geometry and description of the different CAD files.

C0D_panel_marc

Number of panels: 234



CAD files

CAD format	Location	Variant	Description	Conversion date	author
input/iges	./input/iges/MS_C0D_T200v3.igs	-	Original model used in the towing tank, iges is missing from the 'lijnenplannen' list		WVH
cfd/parasolid	./cfd/parasolid/MS_C0D_T200v3_cf_180912_1_cf.x_t	C0D_bare_hull	model created from iges file in cadfix	29 Jan 2013	MLO
input/blend	./input/blend/C0D.blend	-	contains the stl triangulation and the panel geometry	04 Feb 2020	WVH
panel/ropes	./panel/ropes/C0D_coarse.dat	C0D_panel_coarse	coarse version of the panel geometry	27 Mar 2019	WVH
panel/ropes	./panel/ropes/C0D_medium.dat	C0D_panel_medium	medium version of the panel geometry	27 Mar 2019	WVH
panel/ropes	./panel/ropes/C0D_fine.dat	C0D_panel_fine	fine version of the panel geometry	27 Mar 2019	WVH
panel/ropes	./panel/ropes/C0D_marc.dat	C0D_panel_marc	coarse version obtained from Marc Vantorre	27 Mar 2019	WVH
panel/hydrostar	./panel/hydrostar/C0D_coarse.hst	C0D_panel_coarse	converted from panel/ropes/C0D_coarse.dat via C0D_coarse.obj	02 Mar 2020	WVH
panel/hydrostar	./panel/hydrostar/C0D_medium.hst	C0D_panel_medium	converted from panel/ropes/C0D_medium.dat via C0D_medium.obj	02 Mar 2020	WVH
panel/hydrostar	./panel/hydrostar/C0D_fine.hst	C0D_panel_fine	converted from panel/ropes/C0D_fine.dat via C0D_fine.obj	02 Mar 2020	WVH
panel/hydrostar	./panel/hydrostar/C0D_marc.hst	C0D_panel_marc	converted from panel/ropes/C0D_marc.dat via C0D_marc.obj	02 Mar 2020	WVH

This page is automatically created with a python script from info.xml and data from the subdirectories below https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/nautical/hulls/C0D.

2.4 Retrieval of ship data from the towing tank database

For CAD models that have a model scale hull in the towing tank database, characteristic dimensions are extracted from the database instead of duplicating them inside the XML files. This saves time for the user because less values have to be filled in. Extracting data from lists in a Sharepoint site proved to be very easy with the help of the Shareplum Python package[10]. An example to extract four characteristic dimensions of the hull of C0D is shown in Listing 3. There is one catch however, and that is that Shareplum can only handle data stored in lists inside Sharepoint. Data stored in forms are not accessible with the version (0.5.1) used during development.

[10]See https://pypi.org/project/SharePlum/ for documentation and basic examples and https://github.com/jasonrollins/shareplum for the source code.

Listing 3 – Python script to extract data from a Sharepoint list using Shareplum.

```python
def main():
    """
    test function to access wlsow shpgenerator sharepoint site and extract data from it
    using SharePlum
    """

    from shareplum import Site
    from requests_ntlm import HttpNtlmAuth
    import keyring
    import urllib3
    urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)

    pw = keyring.get_password('shpgenerator','vhoydowi')

    auth = HttpNtlmAuth('ALFA\\vhoydowi', pw)

    # verify_ssl should be False, otherwise it does not seem to work
    site = Site('https://wlsow.vlaanderen.be/shpgenerator', auth=auth, verify_ssl=False)

    ship_hull_list = site.List('Scheepsrompen')  # scale models
    # use a query to get the correct ship, and a field to get the correct columns
    query = {'Where':[('Eq','Titel','COD')]}
    fields = ['MLOA','MLPP','MB','MD']
    ship_data = ship_hull_list.GetListItems(fields=fields, query=query)
    print(ship_data)

if __name__ == "__main__":
    main()
```

3 Conclusions

This report documents the implementation of a system that was conceived to store and document CAD files for use in CFD software and potential panel methods in use at Flanders Hydraulics Research. The system makes use of available software at FHR (Subversion version control and Confluence wiki software). Initially, the wiki pages — that act as a visual front-end for the data in version control — had to be created by hand. This proved to be very tedious and error-prone. An alternative was found with the use of the *atlassian-python-api* Python package that provides a simple interface to interact with Atlassian software products. All relevant information related to a CAD model is stored in a simple XML file. A Python script was developed that extracts data from the XML file and from the subversion repository and based on that, automatically creates a wiki page based on said data. At the moment, this script has to be executed by the user every time a change is made to the `info.xml` file. Ideally, it could be executed automatically after a commit as a post-commit hook if FHR had easy access to the server. This is however not the case at the moment[11], which means that for the time being, the script has to be executed by the user that adds new geometry to the database.

At the time of writing this report, the database contains 14 CAD models of ship hulls, six CAD models of rudders, one propeller model and two other models in the nautical section. The hydraulic section contains the relevant CAD models used in projects 14_050 and 18_134. For the latter group, there is a significant backlog of CFD projects whose CAD geometries must be sorted out and added to version control.

[11] The server is maintained by the IT subcontractor of the Flemish government.

A1 User guide

A1.1 Subversion access

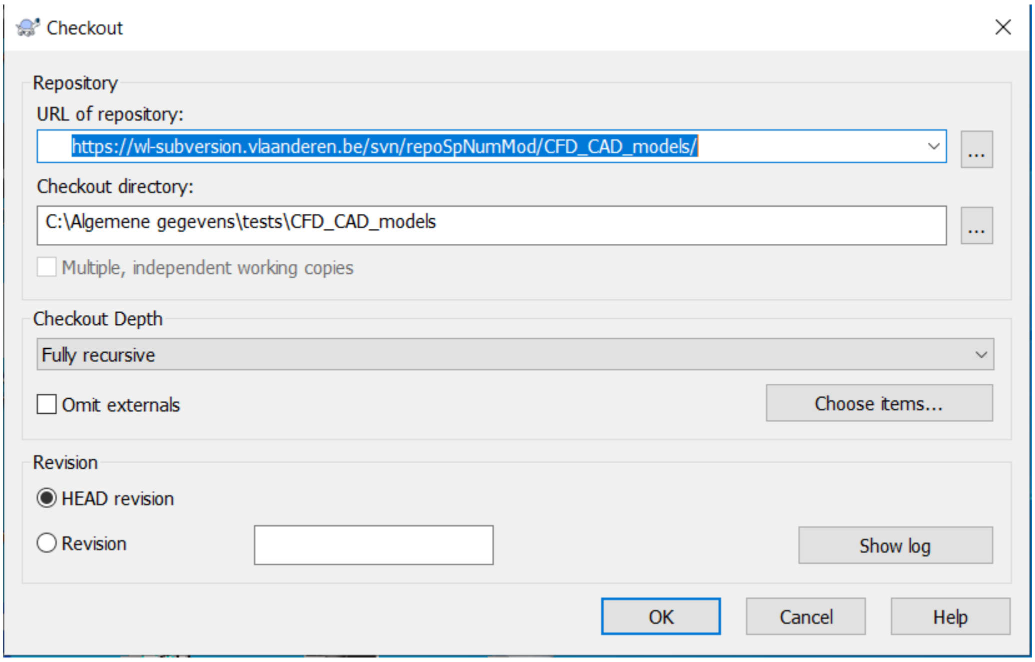
The subversion repository can be accessed directly[12] but the access through the wiki is more convenient because it will show all information related to a CAD model in a single webpage: `https://wlwiki.vlaanderen.be/display/wlwiki/CFD+CAD+model`.

For adding a new model to the database, a checkout of the subversion repository is required. This can be achieved using either the command-line subversion client as follows:

```
svn co https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/ CFD_CAD_models
```

or using the TortoiseSVN GUI client as shown in Fig. 7.

Figure 7 – Checkout of the Subversion repository using the TortoiseSVN GUI client.



Once this is done, a new directory (without spaces) can be added with an `info.xml` file that contains all information about a new CAD model and the changes can be committed.

A1.2 Anaconda Python Environment

The Python script that was developed to automatically create wiki pages requires packages that are not available in the standard channels of Anaconda but requires them to be installed using `pip`. When a new model is added to version control, a wiki page must be created with this Python script.

The environment file (`confluence_api.yml`) that is added to the repository at `https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/` contains all the required packages for creating a

[12] At `https://wl-subversion.vlaanderen.be/svn/repoSpNumMod/CFD_CAD_models/`

new environment in Anaconda to run the Python script. The installation of Python itself will not be discussed here, the `wlwiki` contains all necessary information: `https://wlwiki.vlaanderen.be/display/wlwiki/Installing+Python+and+its+libraries`.

Due to the installation of packages using `pip` and the fact that a proxy server is required, the following file must be added `%APPDATA%\pip\pip.ini` with the following contents:

```
[global]
proxy=https://vipproxy.vlaanderen.be:8080
```

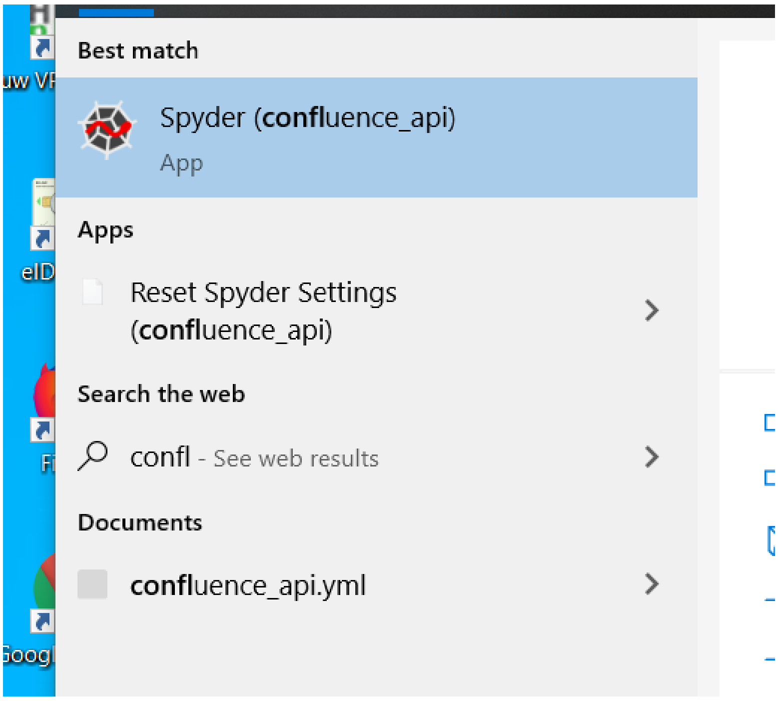
Then, installing the new environment should be as simple as:

```
conda env create -f confluence_api.yml
```

Without the proxy settings, the `conda` installation will hang once the pip packages need to be installed.

At this point, the `spyder` editor should be available for the `confluence_api` environment from the start menu (see Fig. 8).

Figure 8 – Spyder editor in `confluence_api` environment.



A1.3 Creating a new wiki page

Log on to the wiki and create a new blank page with the same name as the new directory in the subversion checkout[13]. This step is required because at the moment the Python script can only adapt an existing wiki page, it cannot create a new page from scratch. This feature may be added in the future.

Editing a wiki page requires a user account that is protected with a password. This username and its associated password are also required by the Python script. The method to store and retrieve the password associated with a certain user is documented on the wiki: `https://wlwiki.vlaanderen.be/display/wlwiki/Storing+and+using+passwords+in+Python+scripts`.

[13]One disadvantage of the Confluence wiki software is that no two pages can have the same name as they are stored in flat file structure.

Open the Spyder editor (in the correct conda environment) and load the Python script. The `main` function at the end must be modified to point to the correct page in the wiki. Three variables must be set in order to run the script:

`wikiusername` the name of the account on the wiki;

`model_name` the name of the wiki page that will be modified, this one must be the same as a directory name in the subversion repository;

`parent_name` the name of the parent page of the wiki page.

If all goes as it should (e.g. a password for the wiki is found, the XML file contains no syntax nor format errors, the wiki page exists, …), executing the Python script should result in a new or updated wiki page.

DEPARTMENT **MOBILITY & PUBLIC WORKS**
Flanders hydraulics Research

Berchemlei 115, 2140 Antwerp
T +32 (0)3 224 60 35
F +32 (0)3 224 60 36
waterbouwkundiglabo@vlaanderen.be
www.flandershydraulicsresearch.be