D4.2.1. CMDOWS

Author, company:
- Jente Sonneveld, TU Delft
- Anne-Liza Bruggeman, TU Delft
- Gianfranco La Rocca, TU Delft

Version:
1.0

Date:
February 6, 2024

Status:
Final

Confidentiality:
Public

Change log

Revision	Date	Prepared by	Checked by	Description
0.1	22/11/2023	Jente Sonneveld	Bastiaan Beijer	The first setup of this deliverable
0.2	25/01/2024	Anne-Liza Bruggeman	Bastiaan Beijer	Added a general description of CMDOWS + the switch, branches and subworkflow extensions.
0.3	5/02/2024	Gianfranco La Rocca	Bastiaan Beijer	Review and editing work
1.0	6/02/2024	Jente Sonneveld	Bastiaan Beijer	Final review and check

Contents

Acronyms

Acronym	Definition
API	Application Programming Interface
CMDOWS	Common MDO Workflow Schema
DEFAINE	Design Exploration Framework based on AI for froNt-loaded Engineering
MDAO	Multi-disciplinary Design Analysis and Optimization
PIDO	Process Integration and Design Optimization

Acknowledgements

This research is partly funded by the ITEA 3 Call 6 project DEFAINE of the European Union.

References

[1] DEFAINE Consortium, "D4.1.1. First release of Workflow (re-)formulation tool(s)," ITEA3 Call 6, 2022.

[2] DEFAINE Consortium, "D4.1.2. Second release of Workflow (re-)formulation tool(s)," ITEA3 Call 6, 2022.

[3] DEFAINE Consortium, "D4.1.3. Final release of workflow (re-) formulation tool(s)," ITEA3 Call 6, 2023.

[4] I. van Gent, G. La Rocca and M. F. M. Hoogreef, "CMDOWS: a proposed new standard to store and exchange MDO systems," *CEAS Aeronautical Journal,* vol. 9, p. 607–627, 2018.

[5] AGILE, "AGILE Aircraft 3rd Generation MDO for Innovative Collaboration of Heterogeneous Teams of Experts," H2020-EU.3.4., [Online]. Available: https://www.agile-project.eu/. [Accessed 23 12 221].

[6] I. van Gent, R. Lombardi, G. la Rocca and R. d'Ippolito, "A Fully Automated Chain from MDAO Problem Formulation to Workflow," in *EUROGEN 2017*, Madrid, 2017.

1. Introduction

WP (Work Package) 4 aims to achieve automated (re-)formulation of simulation workflows. This deliverable focusses on the means to store and exchange of the generated workflow formulations. In deliverables 4.1.1 [1], 4.1.2 [2] and 4.1.3 [3] the MDO reformulation tool KADMOS and its developments within the DEFAINE project are described. In order to store and exchange workflow formulations generated by KADMOS in a neutral format (i.e., independent both from KADMOS and any Process Integration and Design Optimization tool selected to execute the workflow) the Common MDO Workflow Schema (CMDOWS) [4] data format was proposed. CMDOWS is an open-source XML-based format, specifically defined to store and exchange MDAO workflow formulations between different applications, including multiple PIDO tools as well as applications for the visualization and inspection of the given formulation. To accommodate the developments towards dynamic reformulation of workflows within the DEFAINE project, the following developments related to the exchange of workflow formulations are described in this deliverable:

- A new version of CMDOWS (v0.10) to store dynamic MDAO workflow formulations
- A new CMDOWS importer targeting the Noesis Solutions' commercial PIDO tool 'Optimus', to enable fully automated materialization and execution of formulated (static and dynamic) workflows.

The second development goes beyond the objectives originally set in the project proposal, but it was deemed necessary to the integration of the WP2 project demonstrators, otherwise not achievable using existing open-source tools.

1.1. Intended use and purpose of this deliverable

This deliverable is of type 'document' in the form of an updated version of the CMDOWS standard for MDAO system formulation exchange, in XSD file format. The purpose of this deliverable is to provide some background and an overview of the proposed changes to the CMDOWS standard. Chapter 2 provides an overview of the CMDOWS data standard (based on existing literature), Chapter 3 details the extension to the CMDOWS standard implemented within DEFAINE, Chapter 4 describes the CMDOWS-Optimus importer's development, and Chapter 5 provides some conclusions.

2. The CMDOWS data standard

CMDOWS [4] is a standard data schema that supports the formalization and storage of MDAO workflows. It is an XML-based data schema that, besides storage, supports the exchange of MDAO workflows between MDAO process tools as shown in Figure 1. CMDOWS was originally developed by TU Delft, within the EU project AGILE [5]

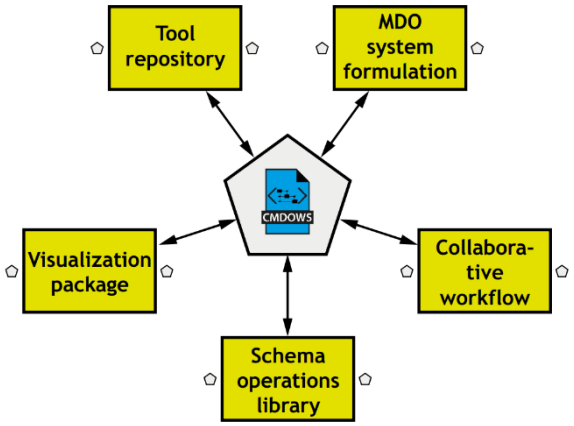

Figure 1: CMDOWS schema to support the exchange of MDAO workflows (adapted from [4])

A high-level overview of CMDOWS is shown in Figure 2. CMDOWS stores all information related to MDAO workflows. organized into three categories, namely *Information*, *Nodes* and *Connections*.

In the Information category, general information describing the CMDOWS file is stored. Under header, information like the creator, a description of the CMDOWS file and file version is stored. The *problemDefinition* element contains information about the formulated MDAO problem. This includes amongst others, the adopted MDAO solution strategy (*problemFormulation*), which include, for example, the IDFand MDFMDO architectures, or DoE (Design of Experiments) formulations or design convergence workflow based on the Gauss-Seidel or Jacobi schema. Furthermore, it describes the problemRoles, i.e. the role of every component of the given workflow formulation, such as design variables, constraints, objectives and other quantities of interest.

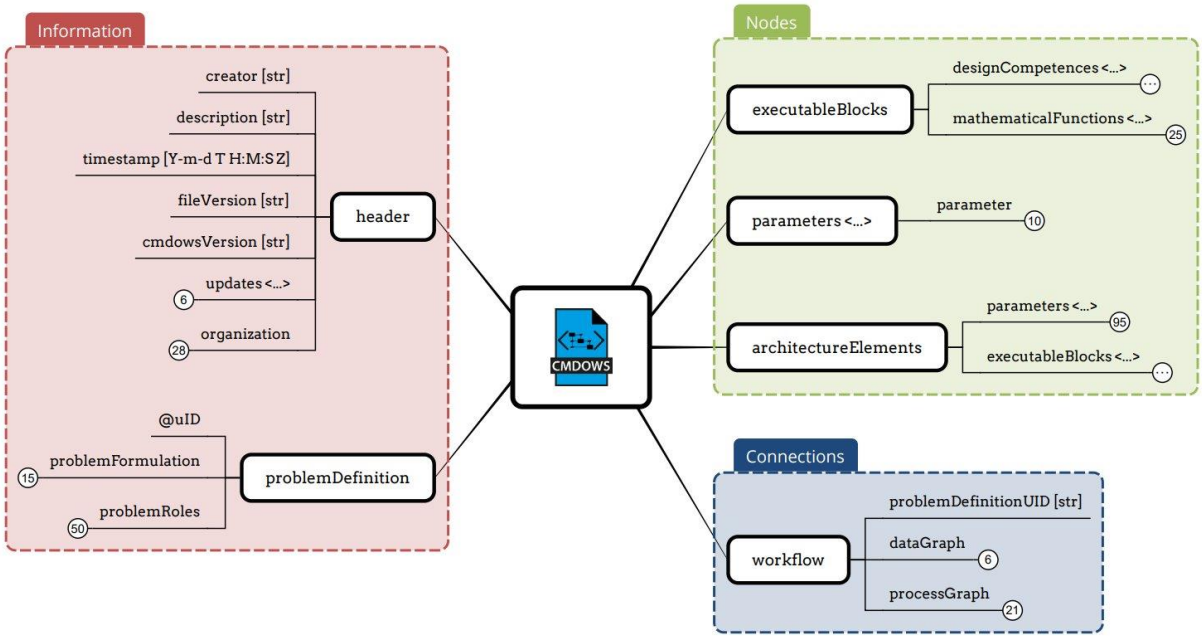

Figure 2: High level overview of the CMDOWS format and its three main element categories (based on [4])

The Nodes category store all the nodes present in the MDAO workflow. The *executableBlocks* element stores the simulation and analysis tools that are required to solve the MDAO problem. Executable blocks can either be design competences (e.g. simulation tools) or mathematical functions. Besides the executable tools, also the parameters presents in the MDAO workflow are stored. These consists of all the variables that are input or output for at least one executable block. Lastly, the *architectureElements* store all the non-discipline specific components of the given MDAO system formulation, such as optimizers, convergers and DoEs, as well as parameters like the initial guesses for coupling variables, or the final optimized values of the design variables.

The Connections category stores all connections between the executable blocks and parameters. In the *dataGraph* element, all data connections between executable blocks and parameters are stored. This includes information such as which parameters are connected as inputs and which are connected as outputs to the executable blocks. The *processGraph* element stores the execution sequence of the executable blocks.

A detailed description of the CMDOWS schema can be found in van Gent, la Rocca and Hoogreef [4]. The following chapters will focus on the adaptations made to the CMDOWS schema in DEFAINE to support the formalization and storage of dynamic MDAO workflows.

3. CMDOWS adaptations to enable dynamic workflows

The original definition of CMDOWS from [4] was able to store only static MDAO workflows. Within Defaine, new dynamic workflows have been developed, i.e., workflows where constraints, disciplinary tools and design variables change at execution time, depending on the current design vector. To enable the formalization, storage and exchange of these dynamic workflows, four new elements needed to be introduced in CMDOWS, namely switches, branches, subworkflows and variable placeholders. Each of them will be shortly explained in the following sections.

3.1. Switches & branches

A switch activates/deactivates different alternative branches of an MDAO workflow. Based on the input provided to the switch and the execution conditions specified for each branch, the switch determines which branch is going to be executed. The branches of a switch are mutually exclusive, which means that only one of the branches is executed per iteration. The executed branches can different per iteration, according to the execution conditions evaluated by the switch.

The elements displayed with red fonts in Figure 3 show the new elements added to the CMDOWS schema. Switches are stored under the *architectureElements/executableBlocks* node. The reason for this is that the switch supports in solving the problem, similar to for example convergers or optimizers. The switch is not part of the problem itself, meaning it is not a simulation or analysis tool that is required to solve the problem.

Each switch element has a *uID* and a *label*. Furthermore, it has two or more branches. A branch is an executableBlock that is activated or deactivated depending on the evaluation of some condition. Therefore, each branch has three child elements, namely *conditions*, *relatedExecutableBlockUID* and *decisionVariable*.

The *conditions* element specifies when a branch will be executed. One branch can have multiple conditions. A condition consists of a parameter, a constraint operator (equal to, greater then, smaller or equal then, etc.) and a value. In case of an equality condition also a required equality precision can be provided. The second element, *relatedExecutableBlockUID* points to one executableBlocks element as indicated with the green arrow in Figure 3, meaning that a branch consists of either a simulation/analysis tool, a mathematical equation or a subworkflow, as explained in the next section.

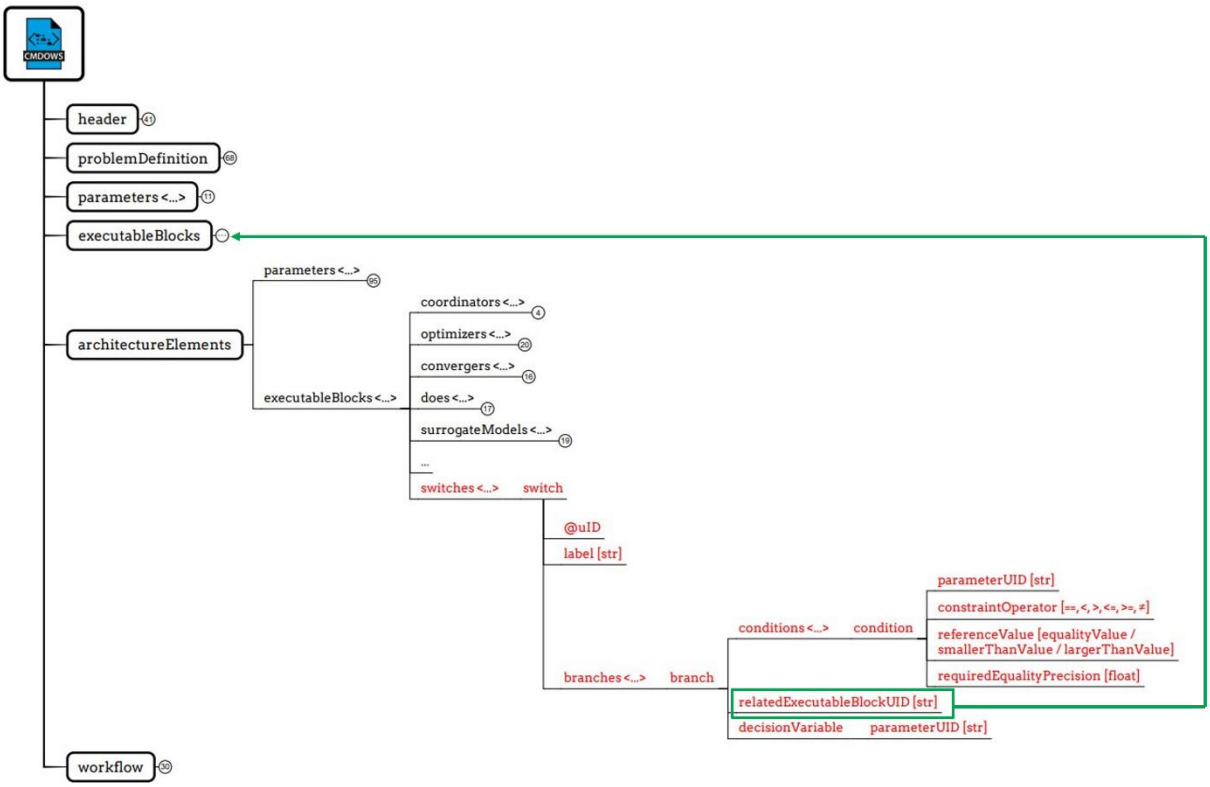

Figure 3: CMDOWS 'architectureElements' element extended with switches element which include a reference to an 'executableBlock' element (see Figure 4)

Finally, the decision variable is a Boolean that can either be True or False. A decision variable is calculated for each branch by the switch based on the conditions specified for each branch. If all the conditions of a branch are satisfied, its corresponding decision variable will be set to True, meaning that this branch will be executed. When the decision variable of a branch is set to False, the branch will not be executed. As mentioned before, the branches of a switch are mutually exclusive, meaning that only one decision variable can be True per iteration.

3.2. Subworkflows

The third new element that needed to be added to CMDOWS is the subworkflow element. A subworkflow is a workflow in a workflow. It is required to solve a problem, therefore the subworkflow has been added to the executableBlocks element as shown in Figure 4.

Each subworkflow has a uID and label. Furthermore, its inputs and outputs are specified, just as for the other executableBlock types (design competence and mathematical equation). A subworkflow is a workflow in itself, so either a different CMDOWS file or a new CMDOWS element can be used to define the workflow. The latter is indicated by the green arrow in Figure 4.

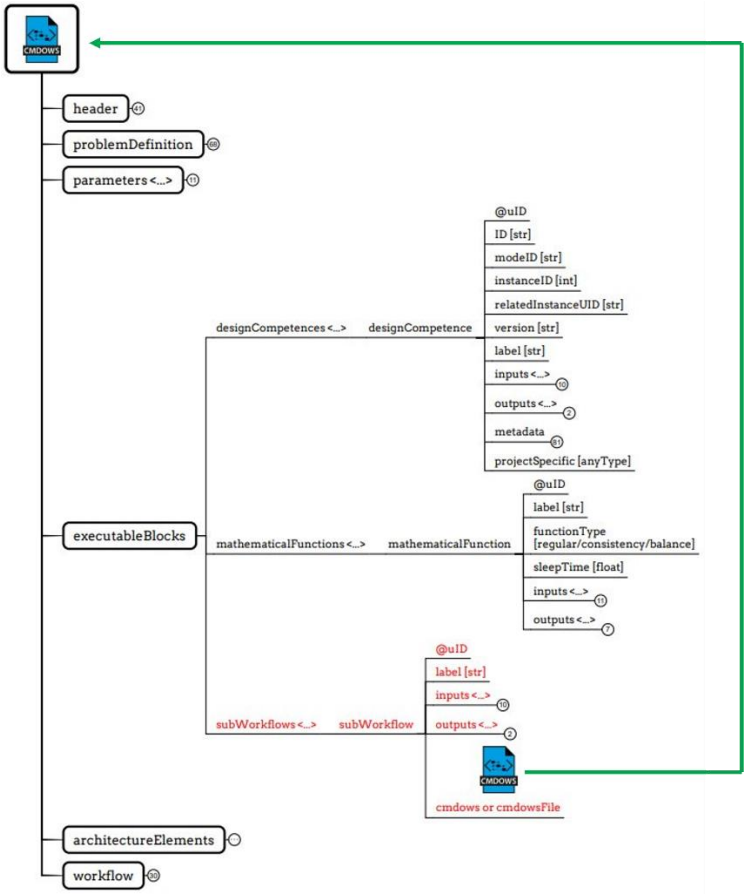

Figure 4: CMDOWS 'executableBlocks' element extended with the new 'subWorkflows' element, including a reference to a sub workflow formalized as another nested CMDOWS.

3.3. Variable placeholder

In some cases, the quantity and nature of variables only become known during the execution of the workflow. An example of these are the so-called hierarchical variables, i.e. variables whose quantity and type depend on the value assumed by other variables. To enable the formulation of a workflow featuring such variables, a new variable type was introduced: the *variable placeholder*. This variable type 'holds the place' of a variable until its true quantity and type becomes known. To this purpose *variablePlaceholder* bool element has been added to the CMDOWS branches *designVariable*, *objectiveVariable*, *constraintVariable* and *stateVariable* elements, as shown in Figure 5. Its value can either be True or False.

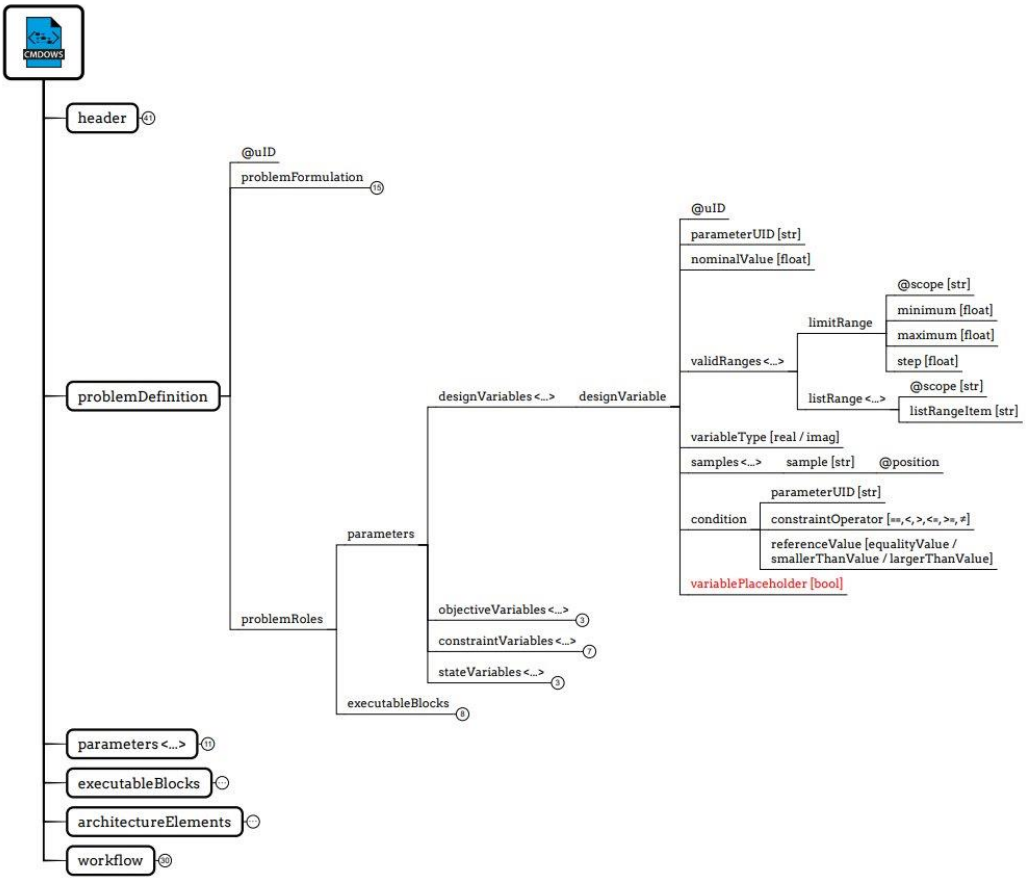

Figure 5: CMDOWS extended with the 'variablePlaceholder' Boolean element.

The adaptations documented in this report are implemented in the latest version of CMDOWS (v0.10) available at: https://gitlab.tudelft.nl/lr-fpp-mdo/cmdows.

4. CMDOWS to Optimus plugin

In the past, a number of 'translators' have been developed to translate CMDOWS files into executable workflows. The process of automatic generation of an executable workflow starting from a CMDOWS file is here addressed as workflow *materialization*. Examples are the translators for the following PIDO tools: DLR's RCE, NOESIS Solutions' Optimus [6] and NASA's open source initiative OpenMDAO.

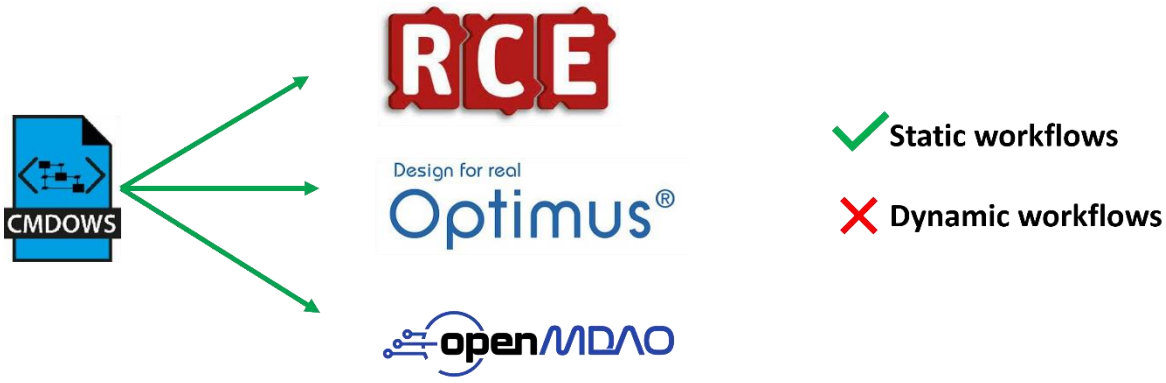

Figure 6: Existing CMDOWS to PIDO tool materialization software

These translators only materialize static workflows, and do not support the changes made to the CMDOWS standard to support dynamic workflows. To enable the automatic materialization and execution of dynamic workflows, a new CMDOWS translation tool was developed in DEFAINE.

The targeted tool to materialize dynamic workflow is the commercial PIDO tool Optimus. Although not originally developed to support dynamic MDO workflows, Optimus natively offers some functionalities which have been exploited to enable the construction of dynamic workflows. These include an interface (called Opt-in-Opt) to integrate Optimus workflows within another workflow, useful for the sub-workflow, and a native switch element. In addition, Optimus features a Python API that enables fully automated workflow materialization.

This has resulted in the new 'Optimus-CMDOWS plugin', capable of materializing static workflows and support the materialization and execution of dynamic workflows, as described in more detail in deliverable D4.1.3 [3]. The plugin and its documentation (see snapshot in Figure 7) are not (yet) open source, but can be made available at request.

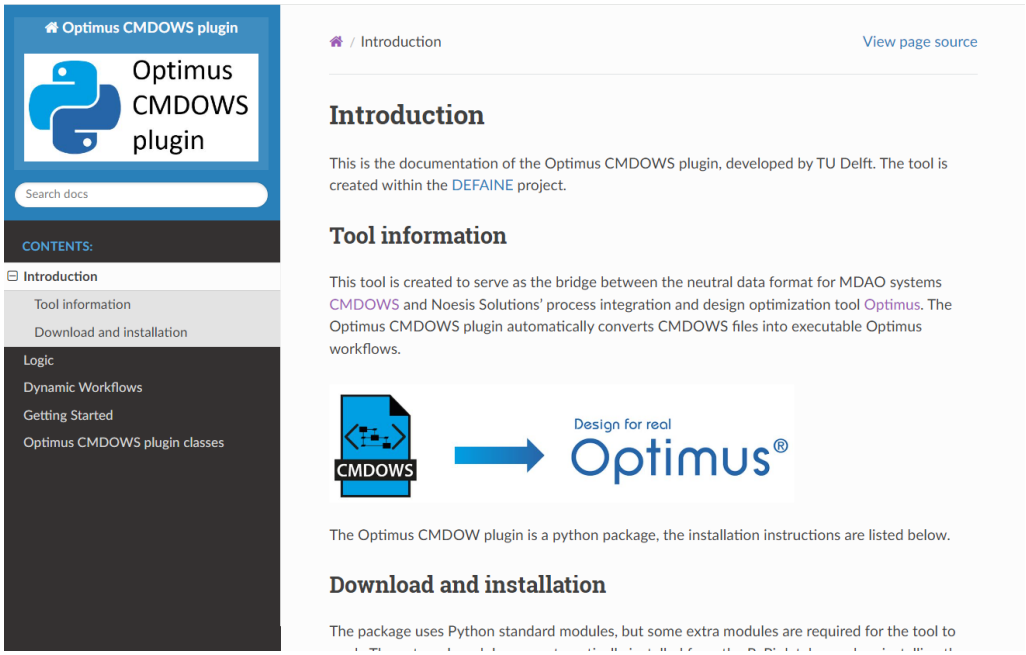

Figure 7: Screenshot of Sphynx documentation made for the Optimus CMDOWS plugin

5. Conclusions

This deliverable discusses the release of a new version of the MDAO workflow exchange format CMDOWS, to accommodate the developments towards dynamic workflows made in the DEFAINE project.

The CMDOWS standard extended to support dynamic workflows, is available as open source at: https://gitlab.tudelft.nl/lr-fpp-mdo/cmdows.

To support the materialization and execution of (static and) dynamic workflows, a new CMDOWS to Optimus translator has been developed and exploited to set up and execute some of the DEFAINE demonstrator from WP3.

While CMDOWS is filling a void in literature, by offering a much needed standard format to store and exchange complete formulations of MDAO systems, the plug-in enable designers to seamlessly design, integrate and execute static and dynamic MDAO workflows, completely automating a process typically requiring a lot of expertise and lengthy, repetitive and error prone manual activities.