



Open
Geospatial
Consortium

Training Session on Authoring OGC Standards with Metanorma

Part 1

Introduction to Metanorma for OGC

Meeting sponsor



2022-09-28



Overview

- Introduction
- Motivation
- Supported OGC document types
- Data models
- Metanorma AsciiDoc

Introduction

- Metanorma is an open-source framework for writing & publishing standardization documents with the focus on semantic authoring and flexible output support.
- Maintained by Ribose in the metanorma GitHub organization
 - <https://github.com/metanorma>
- Supports the authoring of standards by several Standards Development Organizations (see <https://www.metanorma.org>), enabling cross-publishing



Motivation: Some of the problems solved by Metarnoma

- Single source of content for multiple output styles
- Long reviewing times of documents written in typical word processing software
- Documents written in typical word processing software are human error prone
- Conversion of documents to different formats can result in formatting errors
- Duplication of effort in maintenance of bibliographies

Metanorma for OGC

- “Metanorma for OGC” is the implementation of Metanorma for OGC.
- It has been approved as an official way to publish new OGC Standard documents since 2021-09-17, with Metanorma-based document templates subsequently approved by the OGC Document Team (DocTeam) SubCommittee on 2022-02-25.
- Metanorma for OGC documents are created in the Metanorma AsciiDoc format. Metanorma AsciiDoc is a textual syntax for preparing an [ISO/AWI 36100](#) compliant document model tree which can be rendered in a variety of presentation formats.
- In OGC, the supported rendering formats are PDF, HTML, Microsoft Word, and ISO/AWI 36100 XML.

Difference between Metanorma for OGC and other Metanorma flavours

- Supports specification of OGC Standards metadata, including:
 - Document types
 - Stages
 - Identifiers
 - Authorship
- Supports specification of OGC ModSpec (OGC 08-131r3) elements through a specialized syntax
 - Requirements Classes
 - Requirements
 - Conformance Classes
 - Conformance Tests

Including **validation** of linkages between Requirements Classes and Requirements etc.

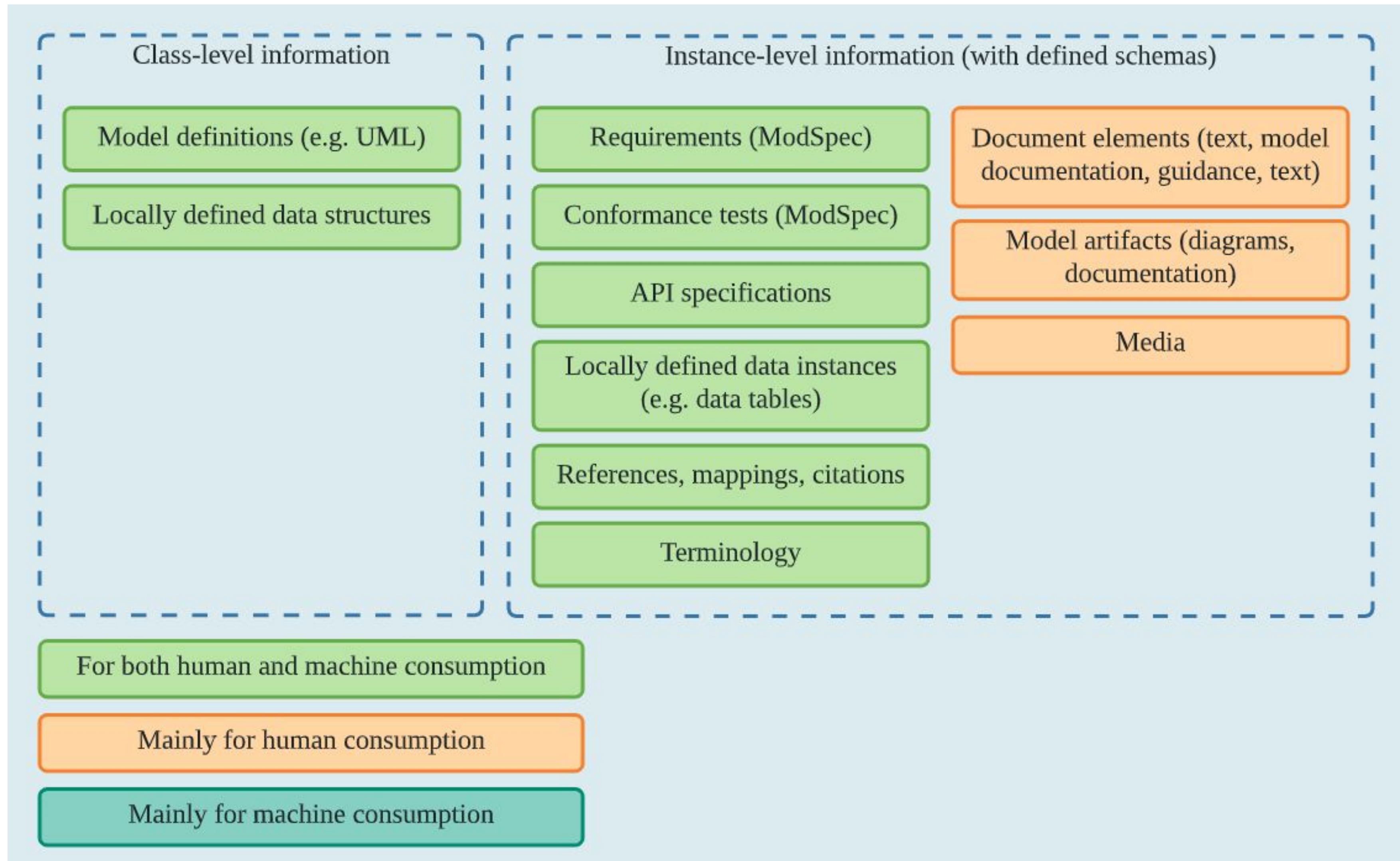
Difference between AsciiDoctor and Metanorma for OGC

Feature	AsciiDoctor	Metanorma for OGC
Document metadata (e.g. docnumber, keywords, etc)	Added as values in table cells	Added as document attributes (More info)
Requirements Classes	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Requirements	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Conformance Classes	Presentation and content specified as tables	Created using a definition list, and then automatically rendered as tables. (More info)
Conformance Tests	Presentation and content specified as tables	Content added as a definition list, and then automatically rendered as tables for Presentation. (More info)

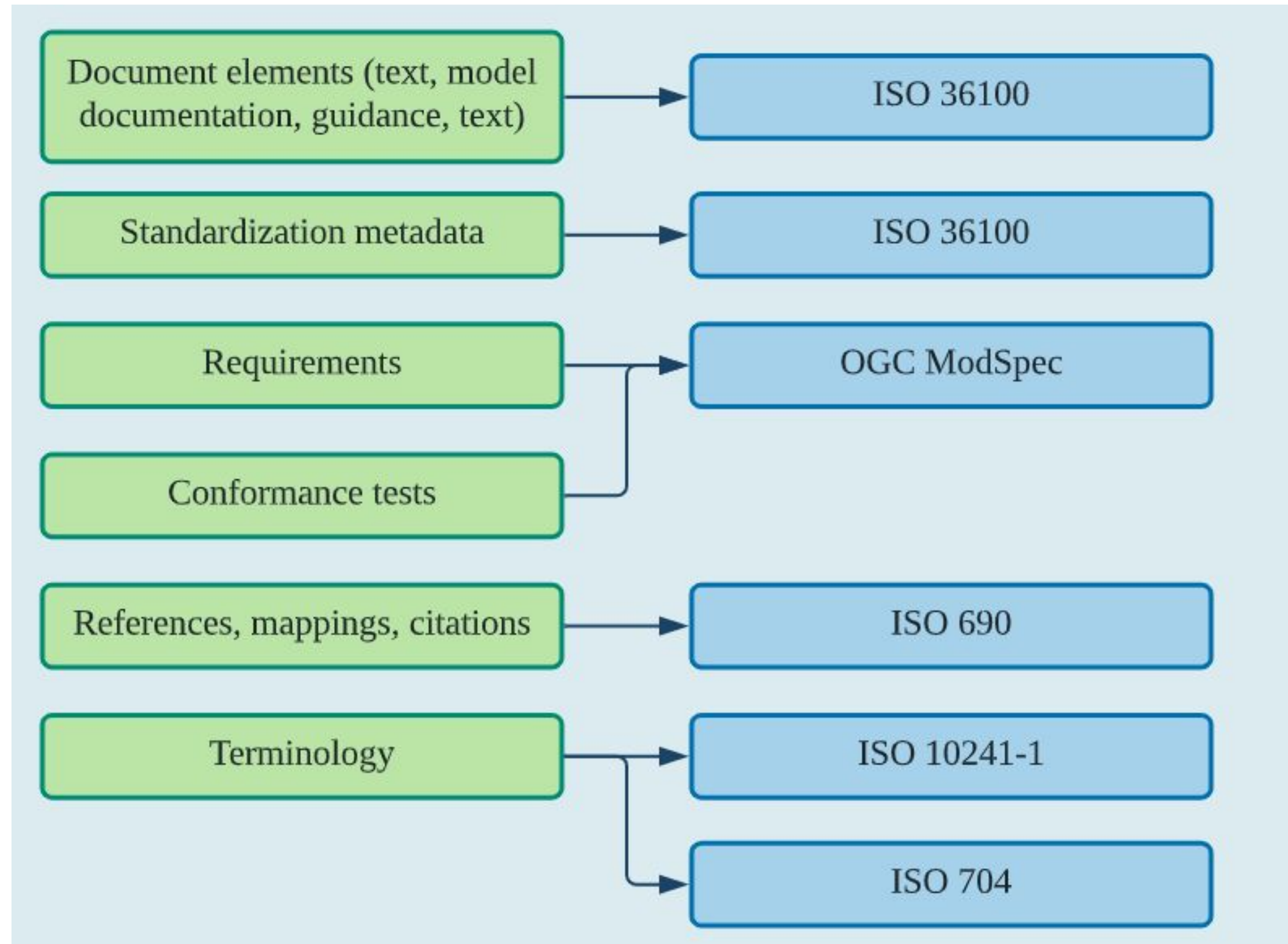
OGC document types supported by Metanorma

- Abstract Specification Topic
- Best Practice
- Community Standard
- Community Practice
- Discussion Paper
- Policy
- Release Notes
- Whitepaper
- Standard
- change-request-supporting-document
- Engineering Report
- Reference Model
- Test Suite
- User Guide

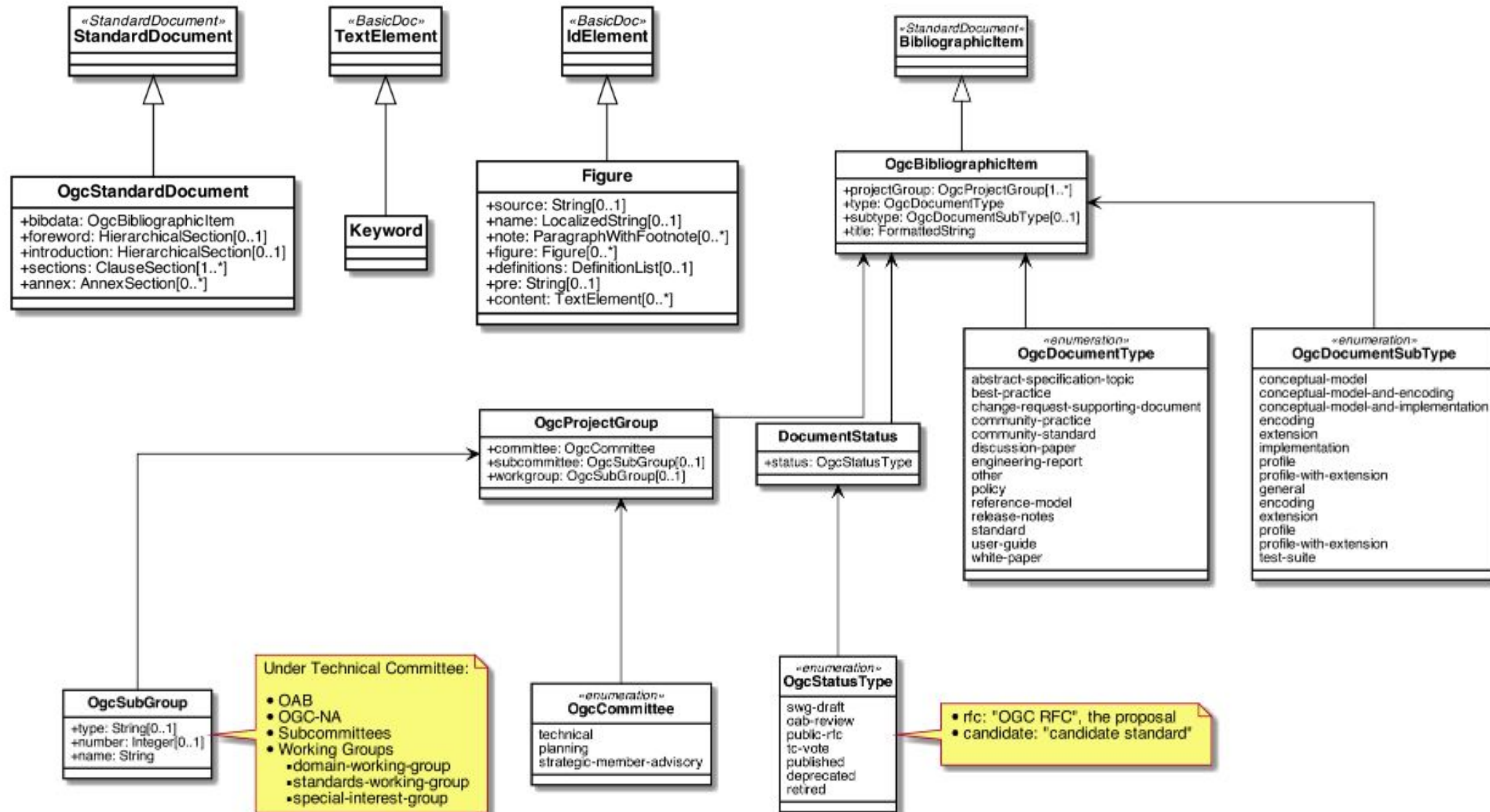
Components of an OGC document



Models used in Metanorma



Data Model for OGC documents in Metanorma



Understanding AsciiDoc

- AsciiDoc is a plain text markup language for writing technical content.
- Documents written in AsciiDoc can be converted to HTML, PDF, and other formats

test.adoc	test.adoc Preview
<pre>1 == Hello Section 2 3 Hello world. 4 5 === Hello Sub Section 6 7 My name is... 8</pre>	<p>Hello Section</p> <p>Hello world.</p> <p>Hello Sub Section</p> <p>My name is...</p>

Sample of tools supporting the AsciiDoc language

- AsciiDoctor
- Metanorma
- AsciiDocFX
- AsciiDoctorJ
- AsciiDoctor.js

NOTE

OGC templates require Metanorma for compilation. Do not use any other tool for compiling OGC documents.

Metanorma AsciiDoc – Basic Syntax

- Metanorma AsciiDoc is based on the [AsciiDoc syntax](#) used by AsciiDoctor, and inherits most of its syntax.
- AsciiDoctor is a reference implementation of AsciiDoc

Metanorma Enhancements to AsciiDoc

- Metanorma auto-numbering extends beyond typical AsciiDoc, applying to document elements including: tables, figures, formulae, and notes
- Controlled rendering of ModSpec elements (requirements, conf classes etc)
- Access to centralized bibliography of OGC, ISO, IETF, and other standards.
- Support for annotations (e.g. Editor, Reviewer, and To-do notes)

AsciiDoc features unsupported by Metanorma

- Sidebars (asides) are not supported. Instead, see [annotations](#).
- Page breaks ("thematic break") are not supported, as Metanorma documents are focuses on semantic encoding.
- ASCII art, or preformatted text ("literal content"), are only supported in selected Metanorma flavors, as they are disallowed in most standardization organizations.

Hands-on Exercise

Pre-requisite

- Install docker, if you have not already done so. Instructions are at <https://docs.docker.com/engine/install/>
- Docker is a containerization software product that reserves and isolates resources on a host machine for use by an application.
- Several applications, including metanorma, can be pulled in from Docker Hub – an online registry of docker images
- Note that the following slides use the color blue to identify parts of a docker command that carries the actual metanorma commands

Installing Metanorma through docker

1. Open a command line terminal
2. Pull the latest docker image of metanorma by running this command from the terminal

```
docker pull metanorma/metanorma:latest
```

```
% docker pull metanorma/metanorma:latest
latest: Pulling from metanorma/metanorma
c229119241af: Pull complete
c14a196c62ee: Pull complete
365bbf24384c: Pull complete
dc13cfc653f2: Pull complete
0c1861070bf3: Pull complete
4f4fb700ef54: Pull complete
a4f7448c4c8a: Pull complete
52817fab15ee: Pull complete
d3efc0b95905: Pull complete
25c8d0468f24: Pull complete
8a904758280f: Pull complete
bf79b9964e2e: Pull complete |
Digest: sha256:d0ac1b888d3fed9cded677de329fb73bce306999c97190b6d9f35fd4a1404f6a
Status: Downloaded newer image for metanorma/metanorma:latest
docker.io/metanorma/metanorma:latest
```

Creating a copy of the OGC Standards template

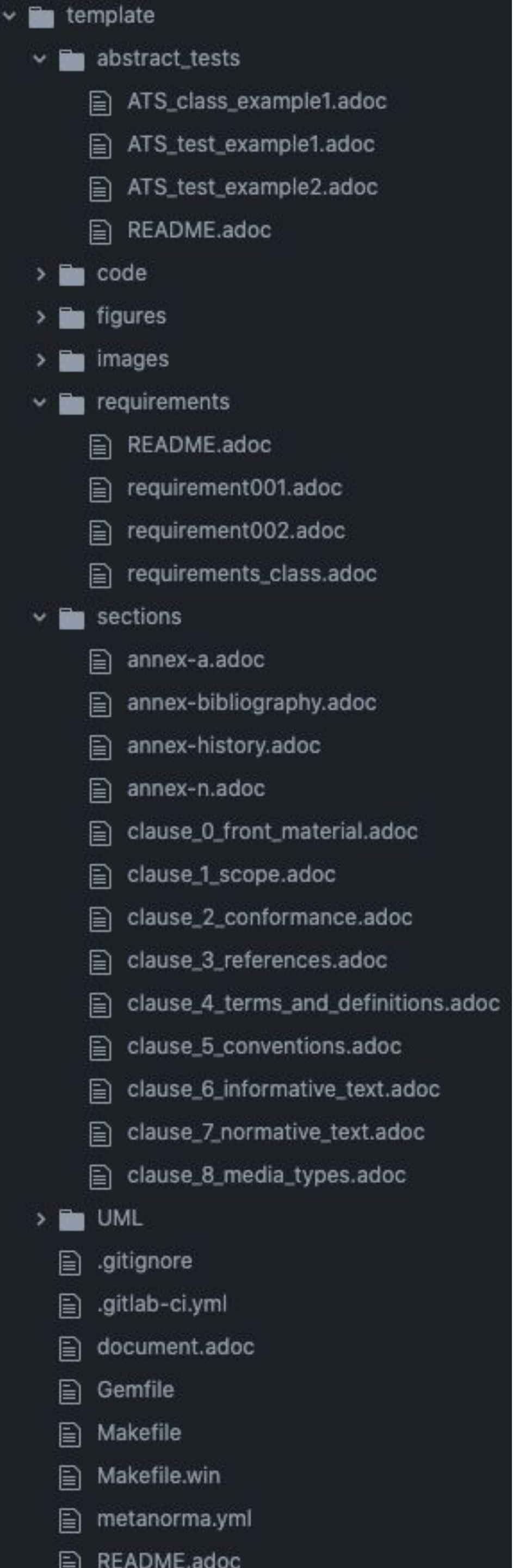
1. To create a copy of the template, run the following command from a terminal (i.e. from the command prompt).

```
docker run -v "$(pwd) " :/metanorma metanorma/metanorma metanorma  
new -d standard -t ogc -l  
https://github.com/metanorma/mn-templates-ogc  
folder_for_standard
```

- NOTE: The `-d standard -t ogc` flags instruct metanorma that the template is for OGC Standards.
- NOTE: Users of Microsoft Windows should replace `$(pwd)` with `%cd%`
- NOTE: The `folder_for_standard` value can be replaced with whatever you would like to be the name of the folder that contains the copy of the template.

Understanding the template

- The template for Standards documents is organized as a folder of asciidoc files, with nested folders for sections, abstract tests, requirements and other resources.
- The main parts are:
 - document.adoc (it can be renamed)
 - images folder
 - sections folder
 - requirements folder
 - abstract_tests folder



Compiling a draft OGC Standard with a docker-containerized Metanorma instance

- To convert the draft standard from AsciiDoc format to HTML and PDF formats, we use the metanorma software to compile the document.

1. From the folder containing the document .adoc file, run the following command.

```
docker run -v "$(pwd)":"/metanorma -v  
${HOME}/.fontist/fonts:/config/fonts metanorma/metanorma  
metanorma compile --agree-to-terms -t ogc -x  
xml,html,doc,pdf document.adoc
```

- NOTE: Users of Microsoft Windows should replace \$(pwd) with %cd%

Screenshot of generated HTML document

DRAFT

OGC STANDARD

CONTENTS

I. ABSTRACT

II. KEYWORDS

III. PREFACE

IV. SECURITY CONSIDERATIONS

V. SUBMITTING ORGANIZATIONS

1. SCOPE

2. CONFORMANCE

3. NORMATIVE REFERENCES

4. TERMS AND DEFINITIONS

5. CONVENTIONS

5.1. Identifiers

6. CLAUSES NOT CONTAINING NORMATIVE MATERIAL

6.1. Clauses not containing normative material sub-clause 1

6.2. Clauses not containing normative material sub-clause 2

7. CLAUSE CONTAINING NORMATIVE MATERIAL

7.1. Requirement Class A or Requirement A Example

8. MEDIA TYPES FOR ANY DATA ENCODING(S)


ANNEX A (INFORMATIVE) CONFORMANCE CLASS ABSTRACT TEST SUITE (NORMATIVE)

A.1. Conformance Class A

ANNEX B (INFORMATIVE) TITLE

ANNEX C (INFORMATIVE) REVISION HISTORY

OGC (add title text)



Open
Geospatial
Consortium

Editor One

Editor

Editor Two

Editor

Submission Date:

2029-03-30

Approval Date:

2029-03-30

Publication Date:

2029-03-30

External identifier of this OGC® document:

http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Internal identifier of this OGC® document:

YY-999

Version: 1.0

Additional Formats: [XML](#) [PDF](#) [DOC](#)

Editing the document metadata

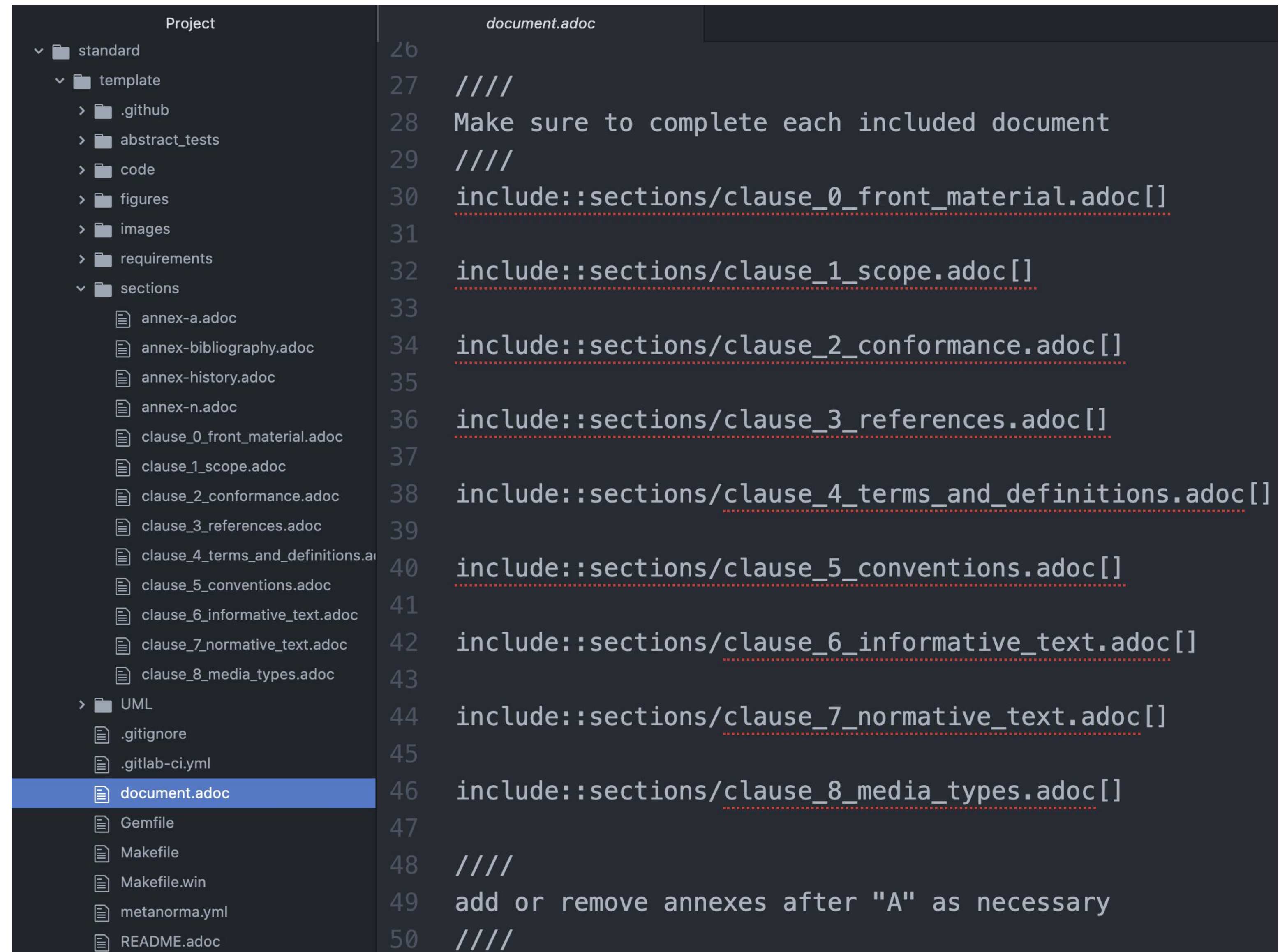
- Title (=)
- Metanorma document class ([mn-document-class](#))
- Document number ([docnumber](#))
- Document type ([doctype](#)), and optionally subtype ([docsubtype](#))
- Document stage ([status](#))
- Committee ([committee](#))
- Author or editor ([fullname](#))
- Version number ([draft](#))
- Submitted date ([received-date](#))
- Approval date ([issued-date](#))
- Publication date ([published-date](#))
- Keywords ([keywords](#))
- Submitting organizations ([submitting-organizations](#))

```
document.adoc
1  = OGC (add title text)
2  :doctype: standard
3  :encoding: utf-8
4  :lang: en
5  :status: draft
6  :committee: technical
7  :draft: 3.0
8  :external-id: http://www.opengis.net/doc
9  :docnumber: YY-999
10 :received-date: 2029-03-30
11 :issued-date: 2029-03-30
12 :published-date: 2029-03-30
13 :fullname: Editor One
14 :fullname_2: Editor Two
15 :docsubtype: Interface
16 :keywords: ogcdoc, OGC document, API, op
17 :submitting-organizations: Organization
18 :mn-document-class: ogc
19 :mn-output-extensions: xml,html,doc,pdf
20 :local-cache-only:
21 :data-uri-image:
22 :pdf-uri: ./document.pdf
23 :xml-uri: ./document.xml
24 :doc-uri: ./document.doc
```


Organizing the document to your liking

Use the **include** directive to import source files into the index/main asciidoc file

NOTE: "include:" lines must be separated with blank lines unless the sections can be stuck together. This is a requirement of AsciiDoc (inherited by Metanorma).



The screenshot displays a code editor with two panels. The left panel shows a project file tree with the following structure:

- Project
 - standard
 - template
 - .github
 - abstract_tests
 - code
 - figures
 - images
 - requirements
 - sections
 - annex-a.adoc
 - annex-bibliography.adoc
 - annex-history.adoc
 - annex-n.adoc
 - clause_0_front_material.adoc
 - clause_1_scope.adoc
 - clause_2_conformance.adoc
 - clause_3_references.adoc
 - clause_4_terms_and_definitions.adoc
 - clause_5_conventions.adoc
 - clause_6_informative_text.adoc
 - clause_7_normative_text.adoc
 - clause_8_media_types.adoc
 - UML
 - .gitignore
 - .gitlab-ci.yml
 - document.adoc (highlighted)
 - Gemfile
 - Makefile
 - Makefile.win
 - metanorma.yml
 - README.adoc

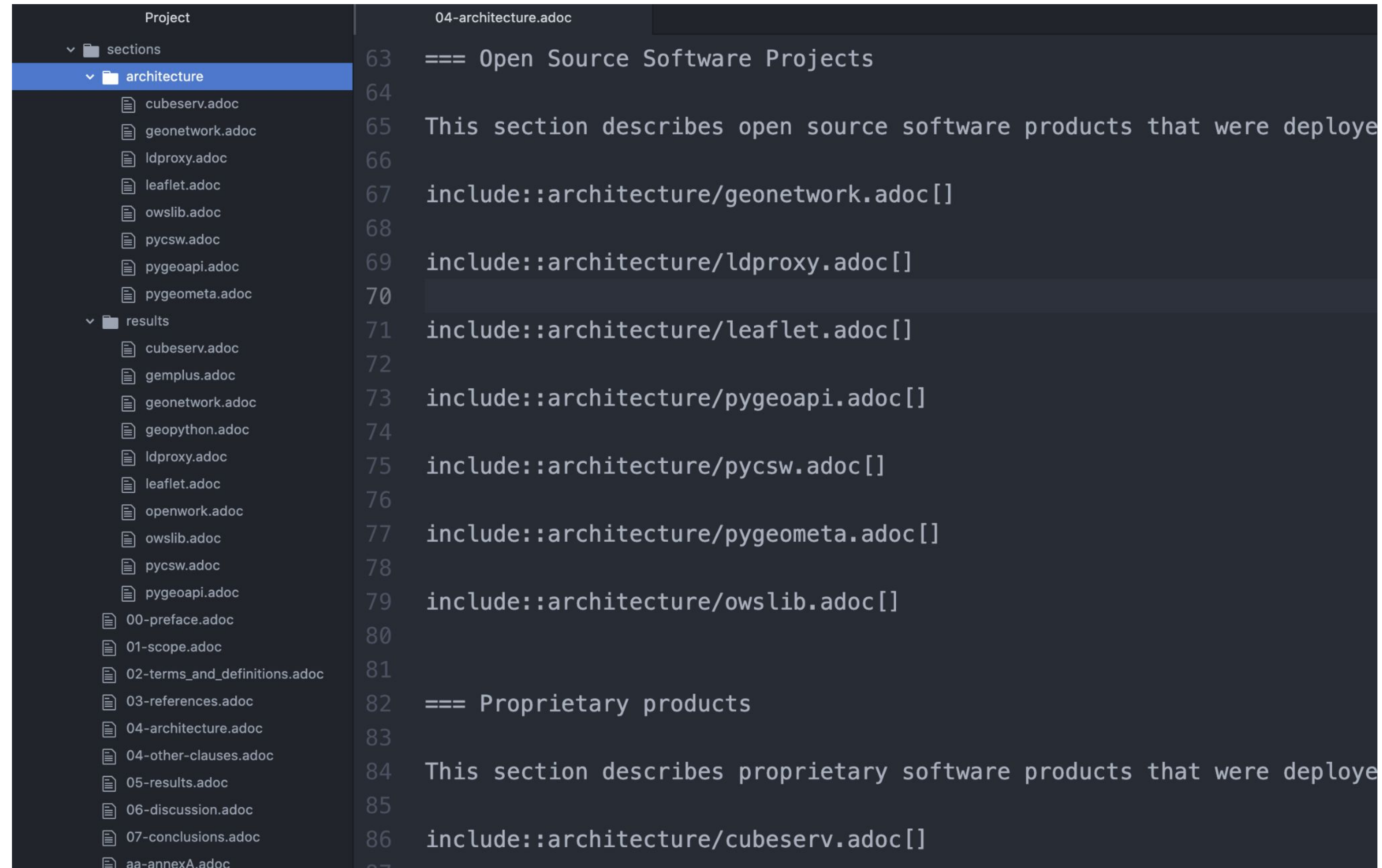
The right panel shows the content of `document.adoc`:

```
26
27  ////
28  Make sure to complete each included document
29  ////
30  include::sections/clause_0_front_material.adoc[]
31
32  include::sections/clause_1_scope.adoc[]
33
34  include::sections/clause_2_conformance.adoc[]
35
36  include::sections/clause_3_references.adoc[]
37
38  include::sections/clause_4_terms_and_definitions.adoc[]
39
40  include::sections/clause_5_conventions.adoc[]
41
42  include::sections/clause_6_informative_text.adoc[]
43
44  include::sections/clause_7_normative_text.adoc[]
45
46  include::sections/clause_8_media_types.adoc[]
47
48  ////
49  add or remove annexes after "A" as necessary
50  ////
```


TIP: Organizing the document for editing as a team

Split the informative/normative section files into multiple files (e.g. one for each subsection)

Makes it possible for multiple editors to add content simultaneously, in parallel



The screenshot displays a document editor interface. On the left, a sidebar titled 'Project' shows a hierarchical structure. Under 'sections', the 'architecture' folder is expanded, listing files: cubeserv.adoc, geonetwork.adoc, ldproxy.adoc, leaflet.adoc, owslib.adoc, pycsw.adoc, pygeoapi.adoc, and pygeometa.adoc. Below this, a 'results' folder lists: cubeserv.adoc, gemplus.adoc, geonetwork.adoc, geopython.adoc, ldproxy.adoc, leaflet.adoc, openwork.adoc, owslib.adoc, pycsw.adoc, pygeoapi.adoc, and pygeometa.adoc. At the bottom of the sidebar, a list of document sections is shown: 00-preface.adoc, 01-scope.adoc, 02-terms_and_definitions.adoc, 03-references.adoc, 04-architecture.adoc, 04-other-clauses.adoc, 05-results.adoc, 06-discussion.adoc, 07-conclusions.adoc, and aa-annexA.adoc. The main editor area shows the content of '04-architecture.adoc'. It begins with a section header '=== Open Source Software Projects' at line 63. Line 64 is empty. Line 65 contains the text 'This section describes open source software products that were deployed'. Lines 66-67 show an include statement: `include::architecture/geonetwork.adoc[]`. Line 68 is empty. Lines 69-70 show another include statement: `include::architecture/ldproxy.adoc[]`. Line 71 is empty. Lines 72-73 show an include statement: `include::architecture/leaflet.adoc[]`. Line 74 is empty. Lines 75-76 show an include statement: `include::architecture/pycsw.adoc[]`. Line 77 is empty. Lines 78-79 show an include statement: `include::architecture/pygeometa.adoc[]`. Line 80 is empty. Lines 81-82 show a section header '=== Proprietary products'. Line 83 is empty. Line 84 contains the text 'This section describes proprietary software products that were deployed'. Line 85 is empty. Lines 86-87 show an include statement: `include::architecture/cubeserv.adoc[]`.

```
Project
├── sections
│   └── architecture
│       ├── cubeserv.adoc
│       ├── geonetwork.adoc
│       ├── ldproxy.adoc
│       ├── leaflet.adoc
│       ├── owslib.adoc
│       ├── pycsw.adoc
│       ├── pygeoapi.adoc
│       └── pygeometa.adoc
└── results
    ├── cubeserv.adoc
    ├── gemplus.adoc
    ├── geonetwork.adoc
    ├── geopython.adoc
    ├── ldproxy.adoc
    ├── leaflet.adoc
    ├── openwork.adoc
    ├── owslib.adoc
    ├── pycsw.adoc
    ├── pygeoapi.adoc
    └── pygeometa.adoc
00-preface.adoc
01-scope.adoc
02-terms_and_definitions.adoc
03-references.adoc
04-architecture.adoc
04-other-clauses.adoc
05-results.adoc
06-discussion.adoc
07-conclusions.adoc
aa-annexA.adoc
```

```
04-architecture.adoc
63 === Open Source Software Projects
64
65 This section describes open source software products that were deployed
66
67 include::architecture/geonetwork.adoc[]
68
69 include::architecture/ldproxy.adoc[]
70
71 include::architecture/leaflet.adoc[]
72
73 include::architecture/pygeoapi.adoc[]
74
75 include::architecture/pycsw.adoc[]
76
77 include::architecture/pygeometa.adoc[]
78
79 include::architecture/owslib.adoc[]
80
81
82 === Proprietary products
83
84 This section describes proprietary software products that were deployed
85
86 include::architecture/cubeserv.adoc[]
87
```




Thank You

Community

500+ International Members
110+ Member Meetings
60+ Alliance and Liaison partners
50+ Standards Working Groups
45+ Domain Working Groups
25+ Years of Not for Profit Work
10+ Regional and Country Forums

Innovation

120+ Innovation Initiatives
380+ Technical reports
Quarterly Tech Trends monitoring

Standards

65+ Adopted Standards
300+ products with 1000+ certified implementations
1,700,000+ Operational Data Sets
Using OGC Standards

