ROUND 3 TECHNICAL KNOWLEDGE TRAINING

DATA MIGRATION

Docs-As-Code

Bosch
Global
Software
Technologies
alt_future

Mr. Nguyen Lam Thanh

Senior Software Engineer

Experiences at Bosch:

- 2022-2023: Software Engineer / Senior Software Engineer

Area of Expertise: Tool development and scripting

Mr. Vu Cong Luat

Senior Software Engineer

Experiences at Bosch:

- 3 years experience in telecommunication domain as software engineer with roles such as analyzing requirement, designing, implementing and release products
- Professional Certificate: OCP11 (Oracle certificated professional, java se 11 developer).

Area of Expertise: **Java** software engineer (**Full stack** developer)

Data Migration – Technical Knowledge Training

Bosch Global Software Technologies
alt_future

Agenda

01.

What is Docs-As-Code?

02.

Why is Docs-As-Code needed?

03.

Requirement management with Docs-As-Code

04.

Test management with Docs-As-Code

05.

Documentation with Docs-As-Code example

© Bosch Global Software Technologies Company Limited 2022. All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

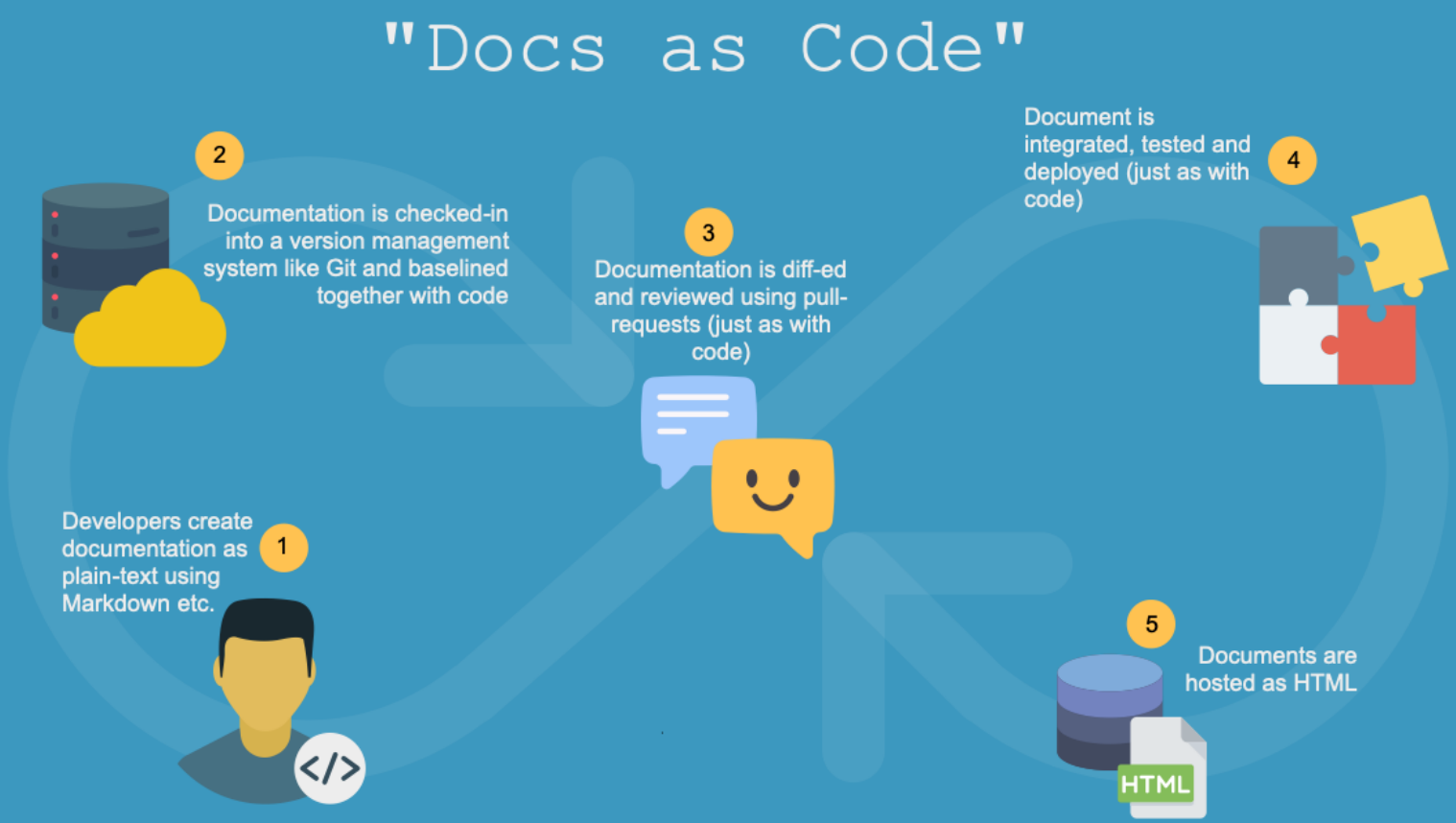
BOSCH

1. What is Docs-As-Code?

- Stands for "***Documentation as Code***".
- Use the ***same tools*** & ***workflows*** as ***code***.
- Easily ***Versioning/baselining*** with VCS as code.
- ***Easier to review*** the changes with ***pull-requests*** feature.
- Able to apply ***CI/CD*** for ***testing*** and ***deploying*** documentation.
- ***Better interface*** for documentation.

BOSCH

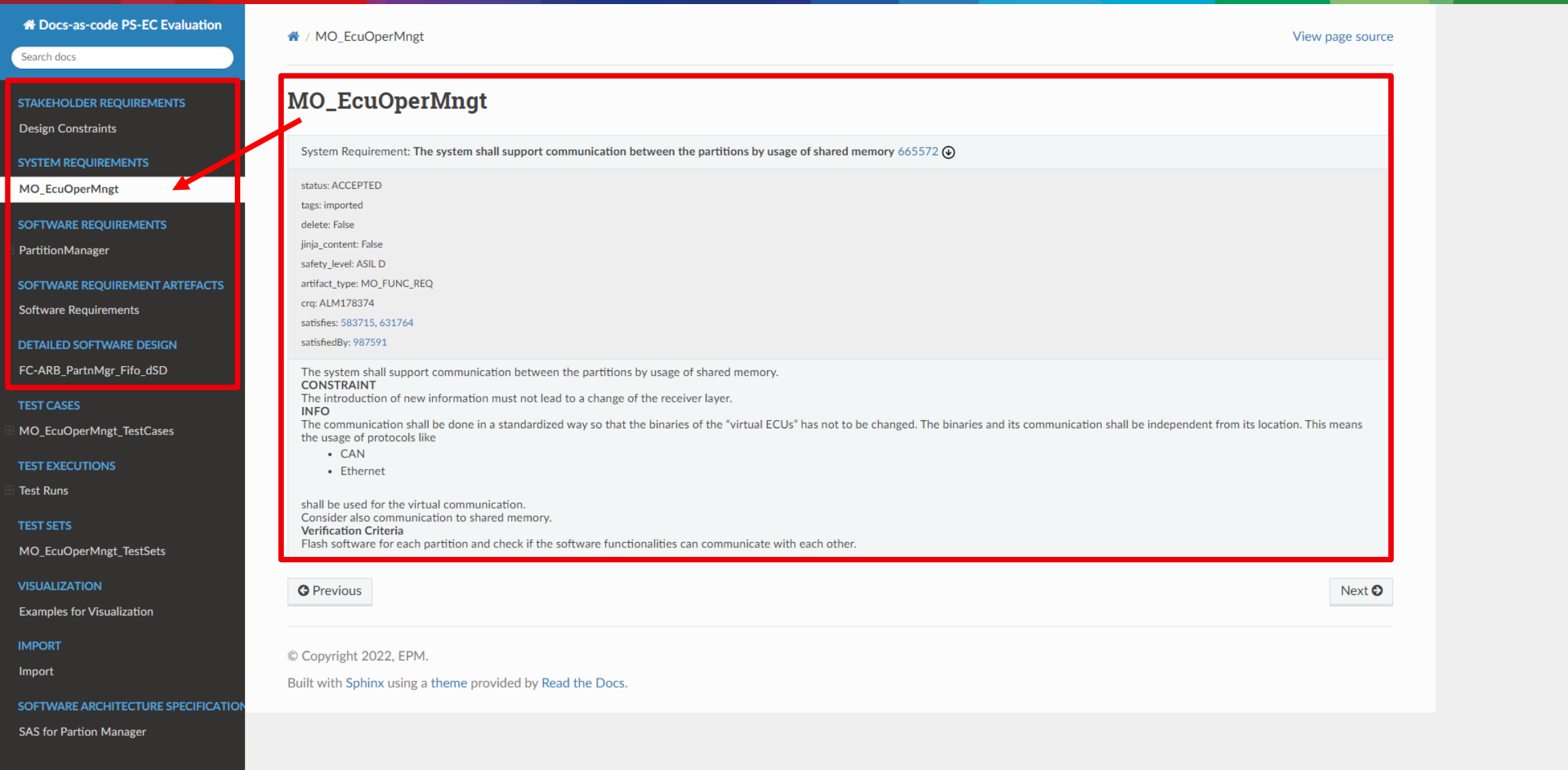
1. What is Docs-As-Code?



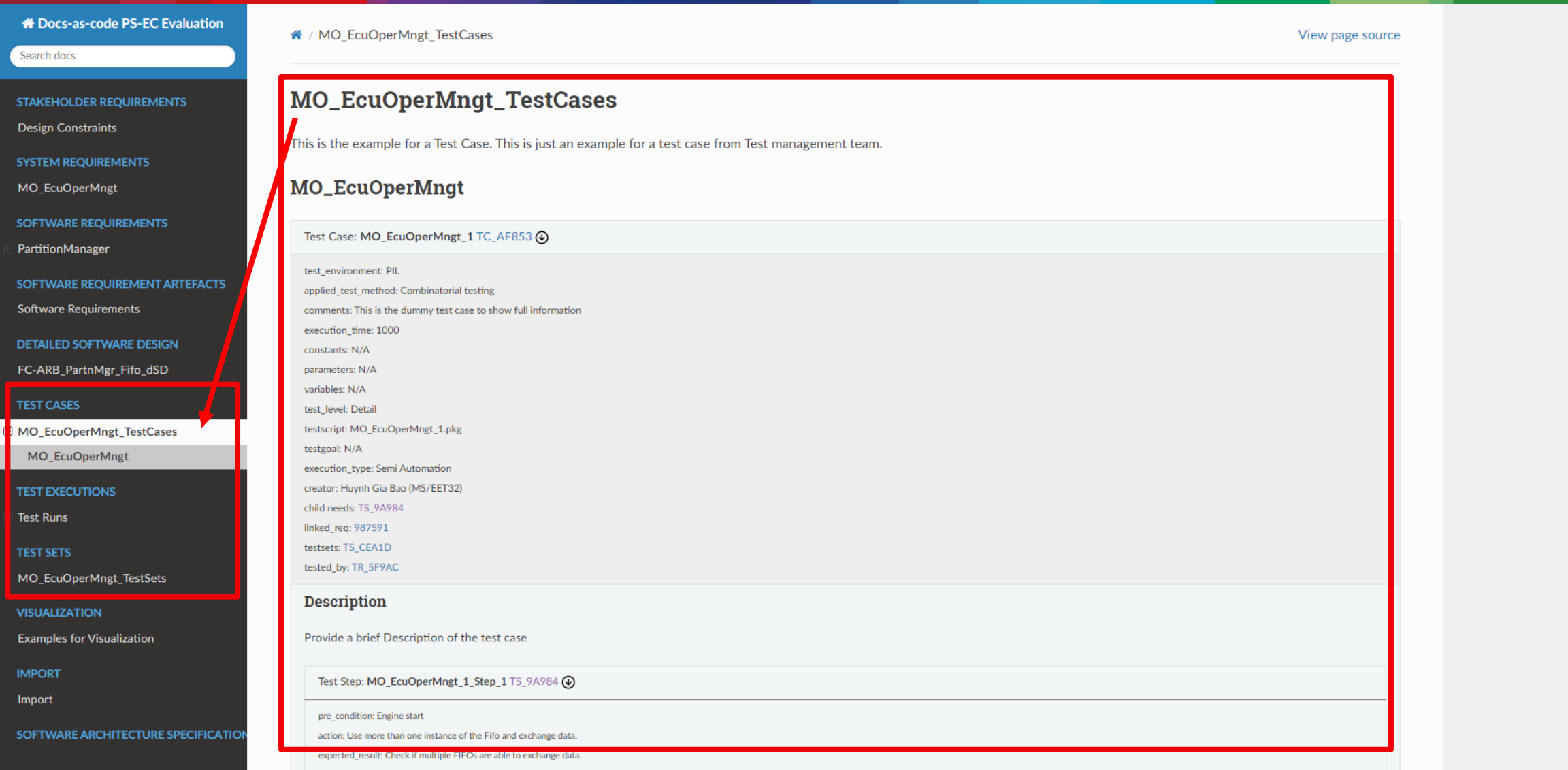
2. Why is Docs-As-Code needed?

- Documentation is done in ***various tools***.
- ***Not easy*** to ***verify*** code, test, and documentation.
- ***Take time*** to ***release*** new ***requirements***.

3. Manage requirements with Docs-As-Code



4. Manage Tests with Docs-As-Code

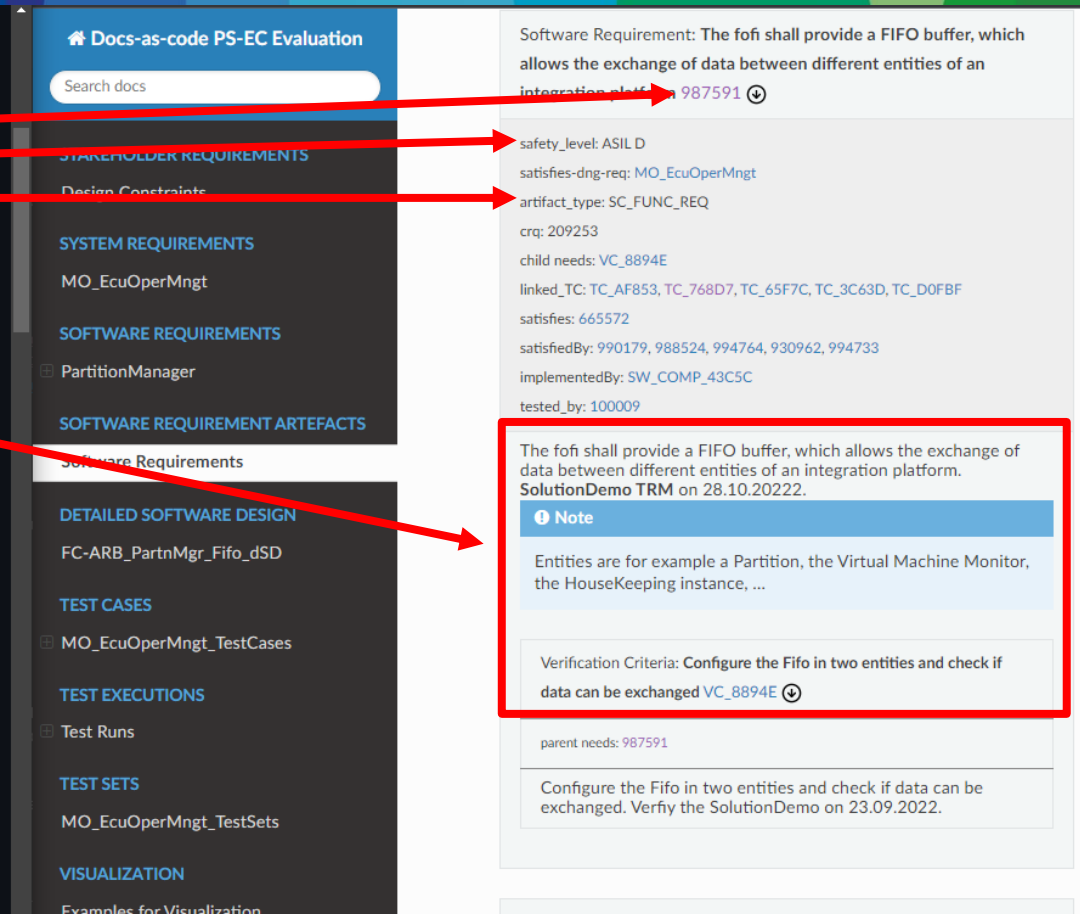


5. Documents with Docs-As-Code example

Software Requirements

```
.. sw_req::
   :id: 987591
   :safety_level: ASIL D
   :artifact_type: SC_FUNC_REQ
   :satisfies-dng-req: BI_8CaJVgBHI8qnM1otgXF7Ew
   :crq: 209253
   :satisfies: 665572

   The fofi shall provide a FIFO buffer, which allows the exchange of data between different entities of an
   **SolutionDemo TRM** on 28.10.20222.

   .. note:: Entities are for example a Partition, the Virtual Machine Monitor, the HouseKeeping instance, …

   .. verify::

      Configure the Fifo in two entities and check if data can be exchanged.
      Verfiy the SolutionDemo on 23.09.2022.
```

```
.. sw_req::
   :id: 1277298
   :status: NEW/CHANGED
   :safety_level: ASIL C
   :artifact_type: SC_FUNC_REQ
   :satisfies-dng-req: BI_8CaJVgBHI8qnM1otgXF7Ew

   It shall be possible to use the FIFO in a system with managed memory access rights.
   Solution Demo Si 12.10.2022.

   .. note:: It has to be possible to use the FIFO in systems where only 1 part (sender / receiver) has write
```



Clean code

CODERACE 2023 EXCEED THE LIMITLESS MIND

Clean code

What is it?

What

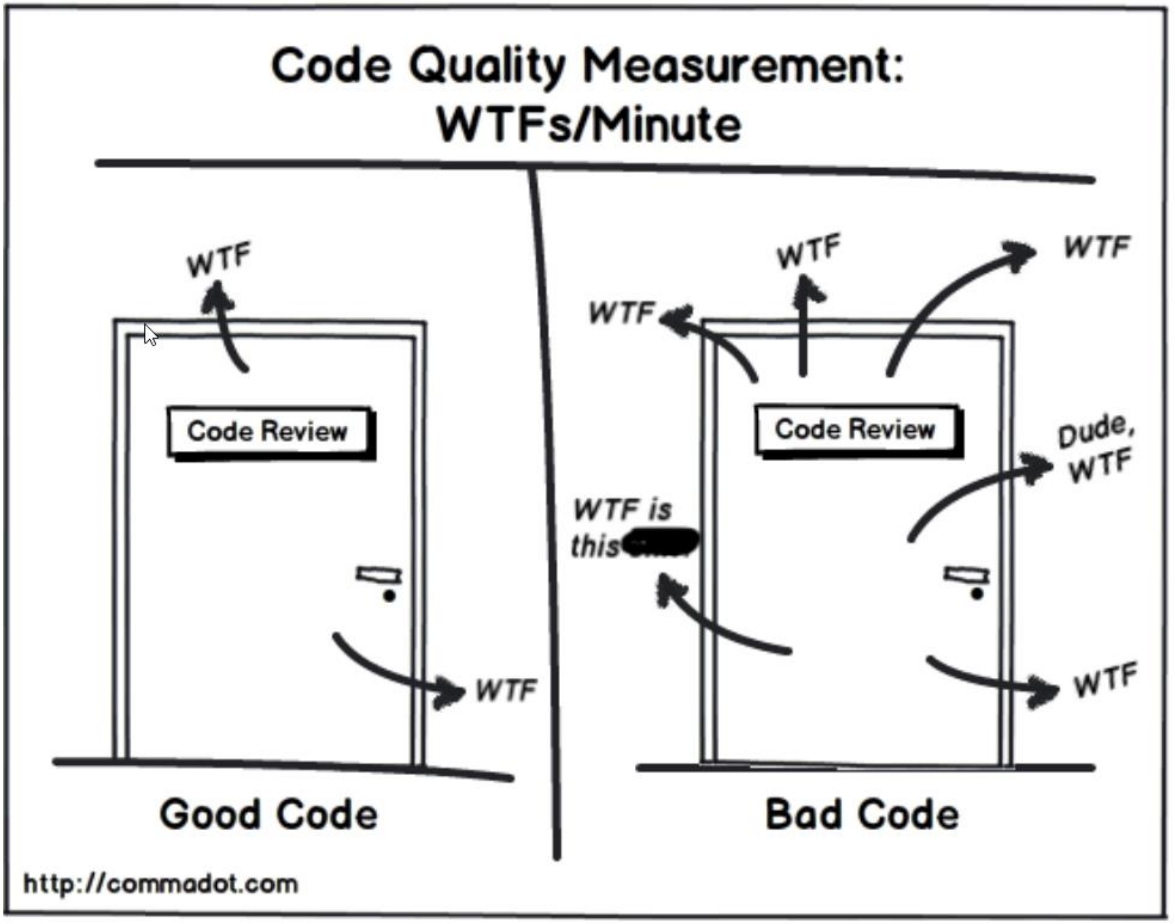
Code is clean if it can be understood easily – by everyone on the team. Clean code can be read and enhanced by a developer other than its original author. With understandability comes readability, changeability, extensibility and maintainability

How

Clean coding is not a skill that can be acquired overnight. It is a habit that needs to be developed by keeping these principles in mind and applying them whenever you write code

Clean code

Clean code principle



- Naming convention
- Functions rules
- Comments rules

Clean code

Naming convention

Bosch Global Software Technologies
alt_future

Should	Should not
Using long descriptive names that are easy to read	Avoid using ambiguous shorthand ○ Do not use letters like x, y, a or b as variable names unless there is a good reason (loop variables are an exception)
Using pronounceable names	Avoid noise words

Clean code

Naming convention

Bosch Global Software Technologies
alt_future

Using long descriptive names that are easy to read

```
# Not recommended
# The au variable is the number of active users
au = 105

# Recommended
total_active_users = 105
```

Clean code

Naming convention

Avoid noise words

Noise word are the words that do not offer any addition information about the variable. They are redundant and should be removed

Some popular noise words are:
- The, Info, Data, Variable, Object, Manager, ...

Clean code

Naming convention

Using pronounceable names

If you cannot pronounce a name, you can't discuss it without sounding silly

Bad:

```
const yyyymmdstr = moment().format("YYYY/MM/DD");
```

Good:

```
const currentDate = moment().format("YYYY/MM/DD");
```

Clean code

Function rules

Bosch Global Software Technologies
alt_future

Should	Should not
Keep them small ○ Functions should be small, really small. They should rarely be 20 lines longs for python (70 lines for java)	**Don’t use flag arguments** ○ A flag argument is a Boolean argument that is passed to a function. Two different actions are taken depending on the value of this argument
Make sure they just do one thing ○ If you follow this rule, it is guaranteed that they will be small	**Don’t repeat yourself** ○ Code repetition may be the root of all evil in software. Duplicate code means you need to change things in multiple places where there is a change in logic and it is very error prone
Fewer arguments ○ Functions should have **two** or fewer arguments, the fewer the better. Avoid three or more arguments where possible	

Clean code

Function rules

Make sure they just do one thing

```
# Not recommended
def getTotalScore(board):
    print("Total score is: ", board.totalScore)
    return board.totalScore

# Recommended
def showResult(arg):
    print("Total score is: ", arg)

def getTotalScore(board):
    return board.totalScore
```

Clean code

Function rules

Don’t use flag arguments

```
text = "Python is a simple and elegant programming language."

# Not recommended
def transform_text(text, uppercase):
    if uppercase:
        return text.upper()
    else:
        return text.lower()

uppercase_text = transform_text(text, True)
lowercase_text = transform_text(text, False)

# Recommended
def transform_to_uppercase(text):
    return text.upper()

def transform_to_lowercase(text):
    return text.lower()

uppercase_text = transform_to_uppercase(text)
lowercase_text = transform_to_lowercase(text)
```

Clean code

Comments rules

Should	Should not
Always try to explain yourself	Don’t leave code in comment ○ Because others who see the code will be afraid to delete. Because they don’t know if it is there for a reason. That commented out code will stay there for a long time.

Clean code

Comment rules

Don’t leave code in comment

```
}
// case 3:  Related have
//  Note: At the present, this widget have not been able to check case Related Att & Mandatory att have link type yet.
else {
    // the block for Mandatory Attribute Checking - vlc1hc comment because, at the present not support
    // attrToCheck.forEach(function(element){
    //  if(element.match(patternAtt) != null){
    //      var attToSearch = convertAttrToSearch(element);
    //      RM.Data.getLinkedArtifacts(artifact, attToSearch, function(linksResult){
    //          numSub++;
    //          if (linksResult.code == RM.OperationResult.OPERATION_OK){
    //              var inlink = linksResult.data.artifactLinks;
    //              var outlink = linksResult.data.externalLinks;
    //              if(inlink.length == 0 && outlink.length == 0){
    //                  missingAttributesTempt.push(element);
    //              }
    //              processLinkAttWithoutRecall(original, numSub);
    //          }
    //      });
    //  }
    // });

    // the block for Related Attribute Checking
    if(valNumRow > 0 && relatedAtt.length > 0 && relatedRulesMap.size > 0){
        var index = [];
        for(var i = 1; i <= valNumRow; i++){
            index.push(i);

            var numLinkType =0;
            relatedAtt.forEach(function(element){
                if(relatedRulesMap.get(element+i) != undefined
                    && element.match(patternAtt) != null){
                    numLinkType++;
                }
            });
```

Clean code

Bosch Global Software Technologies
alt_future

```python
if __name__ == '__main__':
    INP_SRC, OUT_SRC = init_argument()

    # Reading an XML file specified by URL, parsing it into a Python dictionary using the `xmltodict` library
    data_dict = xmltodict.parse(open(INP_SRC).read())

    json_data = json.dumps({
        "Module Name": find_name_module(data_dict),
        "Module Type": find_type_module(data_dict),
        "List Artifact Info": find_list_artifact_info(data_dict)
    }, indent=4)

    with open(OUT_SRC, "w") as json_file:
        json_file.write(json_data)
```

Clean code

Bosch Global Software Technologies
alt_future

```
artifact_info = {}

type_ref = spec_object['TYPE']['SPEC-OBJECT-
type_object = find_type_spec(spec_type, type_ref)

artifact_info.update(
    {"Attribute Type": type_object.get(NAME_TAG)})

for key in spec_object['VALUES'].keys():
    spec_attrs = type_object['SPEC-ATTRIBUTES']
    spec_obj_values = spec_object['VALUES']

    for info in mapping_attr_definition(spec_attrs, spec_obj_values,  attr_key=key):
        artifact_info.update(info)

return dict(sorted(artifact_info.items()))
```

Using noise words

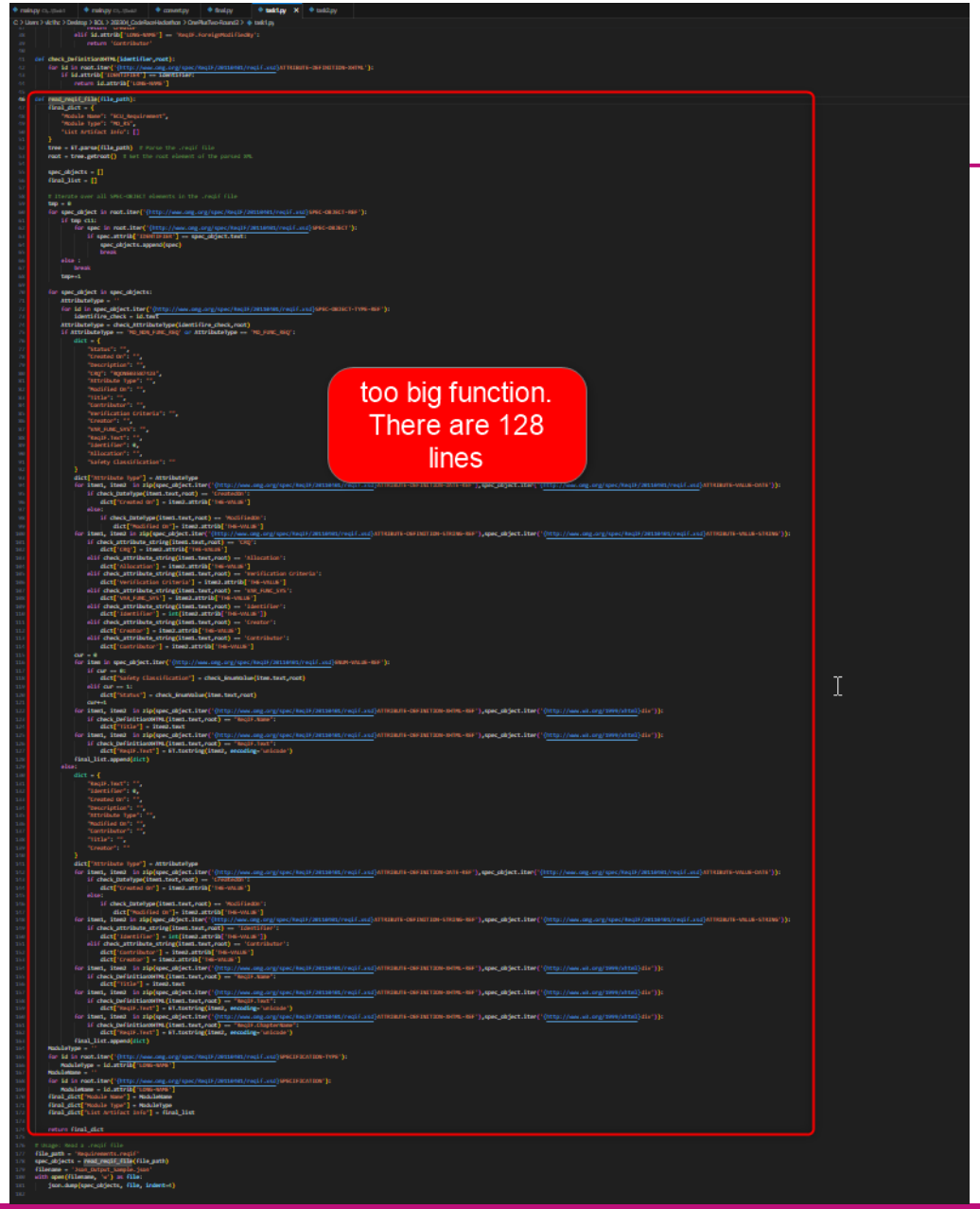
Clean code

Bosch Global Software Technologies
alt_future

```
def extractKeysValues():
    list={}
    global def_dictionary
    for i in spec_objects:
        obj={}
        b = reqif_bundle.get_spec_object_by_ref(i.identifier)
        obj["Attribute Type"]=reqif_bundle.get_spec_object_type_by_ref(b.spec_object_type).long_name#heading(i.spec_object_type)
        obj["Modified On"]=i.last_change
        obj["Description"]=i.description if i.description else ""
        obj["ReqIF.Text"]=""
        for key in i.attribute_map:
            def_ref = i.attribute_map.get(key).definition_ref
            obj[return_key(def_dictionary[def_ref])]=process_value(return_key(def_dictionary[def_ref]),i.attribute_map.get(key).value)

        obj.pop(None)
        list[i.identifier]=obj
    final = []
    for specification in reqif_bundle.core_content.req_if_content.specifications:
        for current_hierarchy in reqif_bundle.iterate_specification_hierarchy(specification):
            final.append(list[current_hierarchy.spec_object])
    return final
```

to short variable name

Clean code



Bosch
Global
Software
Technologies
alt_future

Clean code

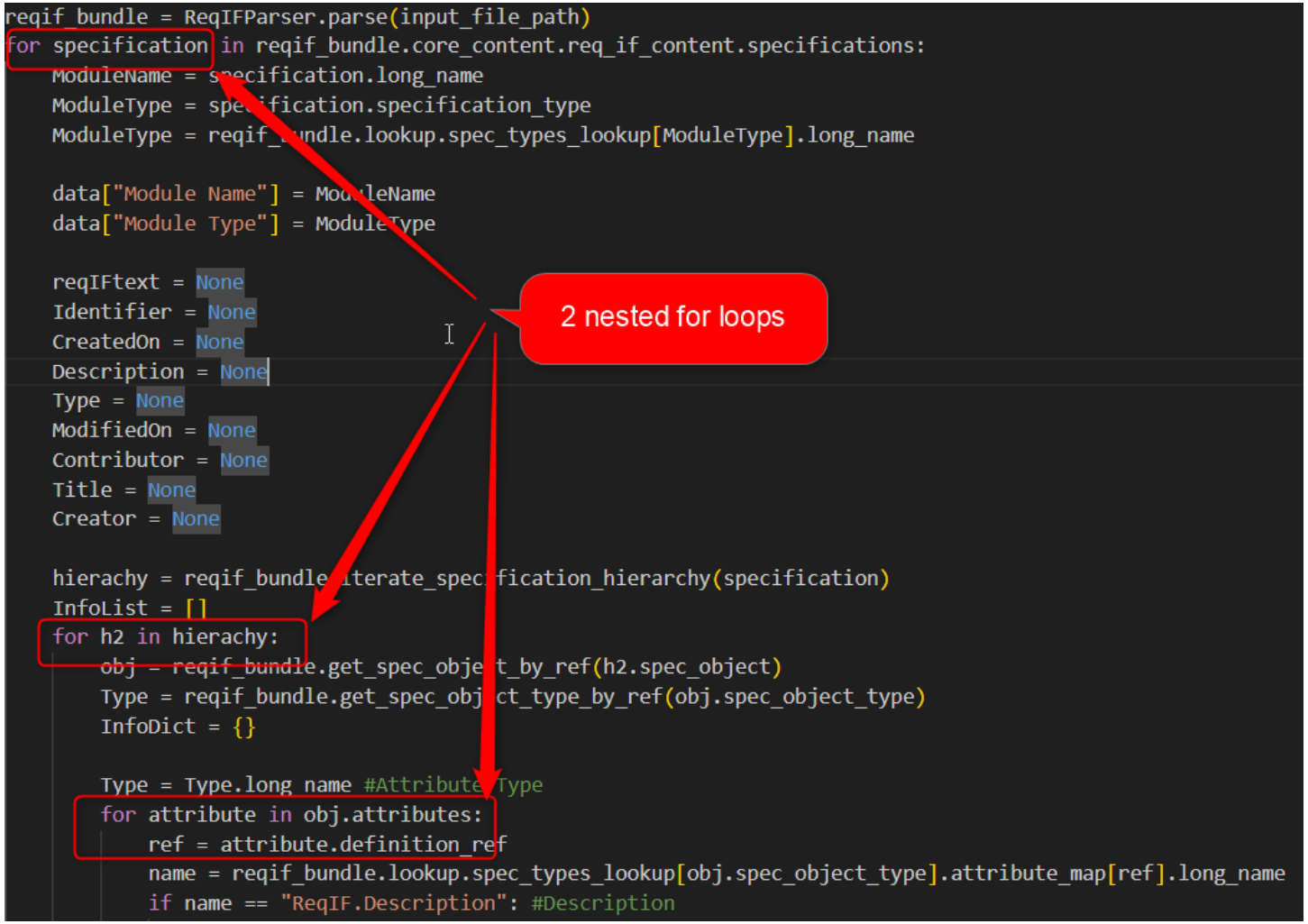
Bosch Global Software Technologies
alt_future

```
def Rst_Write_Content_Formatted(File, Header, Footer, Tab, Content, Prefix1, Prefix2, Empty):
    File.write(Header)

    List_Line_Content = Content.split("\n")
    for Line in List_Line_Content:
        if Empty and Line.strip() == "":
            continue
        File.write(Tab)
        if len(List_Line_Content) > 1:
            File.write(Prefix1)
        Formatted_Line = Limit_Characters_Per_Line(Line, 68)
        for i in range(len(Formatted_Line)):
            if i > 0:
                File.write(Tab)
                if len(List_Line_Content) > 1:
                    File.write(Prefix2)
            File.write(Formatted_Line[i] + '\n')

    File.write(Footer)
```

too much arguments

Clean code

Bosch Global Software Technologies
alt_future



Clean code

```python
from bs4 import BeautifulSoup
from reqif.parser import ReqIFParser
import json
import sys


input_file_path = sys.argv[1]
data = {}

reqif_bundle = ReqIFParser.parse(input_file_path)
for specification in reqif_bundle.core_content.req_if_content.specifications:
    ModuleName = specification.long_name
    ModuleType = specification.specification_type
    ModuleType = reqif_bundle.lookup.spec_types_lookup[ModuleType].long_name

    data["Module Name"] = ModuleName
    data["Module Type"] = ModuleType

    reqIFtext = None
    Identifier = None
    CreatedOn = None
    Description = None
    Type = None
    ModifiedOn = None
    Contributor = None
    Title = None
    Creator = None

    hierachy = reqif_bundle.iterate_specification_hierarchy(specification)
    InfoList = []
    for h2 in hierachy:
        obj = reqif_bundle.get_spec_object_by_ref(h2.spec_object)
        Type = reqif_bundle.get_spec_object_type_by_ref(obj.spec_object_type)
        InfoDict = {}

        Type = Type.long_name #Attribute Type
        for attribute in obj.attributes:
```

THANK YOU