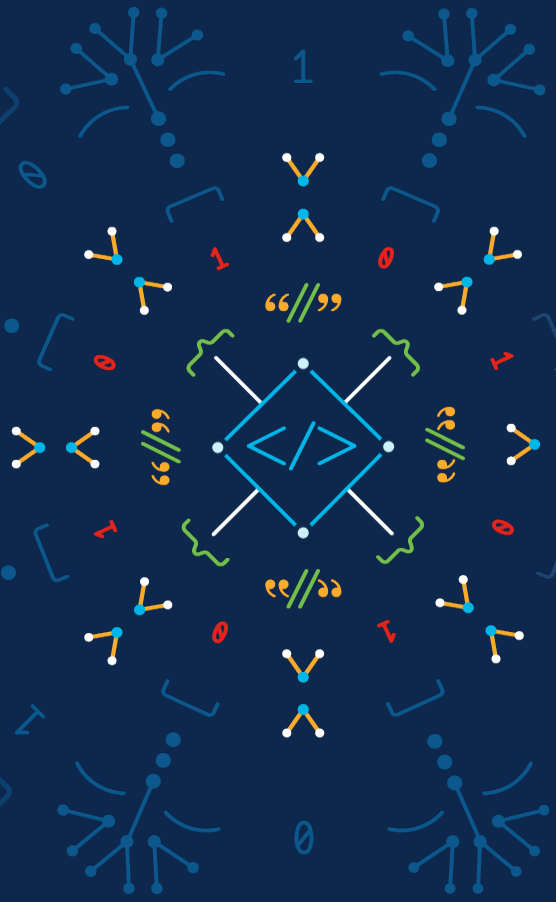
Developer Days
Automation

cisco
The bridge to possible



Enhanced Compliance Reporting

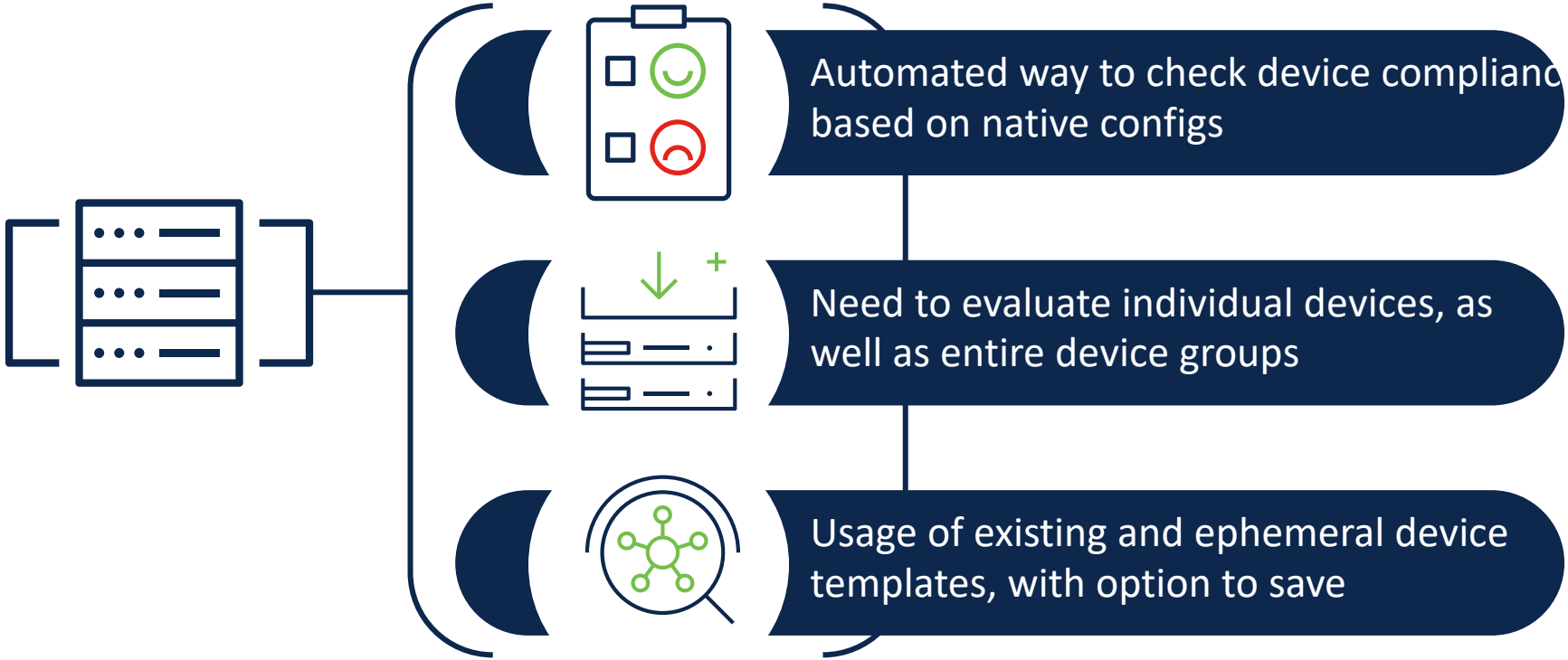
A Development Journey

Alfonso Sandoval, Cesar Alves
Software Consulting Engineers, CX Lisbon
May 11th, 2022

Agenda

1. Our customer User Story
2. Limitations of NSO compliance reporting
3. Enhanced Compliance Reporting PoC
4. First approach - Technical Debt
5. Second approach - Clean code refactoring
6. Demo

Our customer User Story



Limitations of NSO compliance reporting

Diff calculation and report rendering

Only supports explicit device groups

All nodes need to be pre-configured

Rendered report

Device template

Device Group

Compliance report

Quick demo

Render a compliance report in NSO, top to bottom

Our solution:
Enhanced Compliance Report utility

1

Device Groups

\+

1+ Devices/ All Devices

Target Device input

2

or

Device Template

Native config + NED

Match source input

1

Device Groups

\+

1+ Devices/ All Devices

Target Device input

2

or

Device Template

Native config + NED

Match source input

3 When native config and NED are provided

Create a dummy device with the NED

Dry-run of the native configs loading in the dummy device

Embed the resulting XML in a template payload

Create a dummy device template with this payload

Store the device template if it is marked to be persistent

1

Device Groups

\+

1+ Devices/ All Devices

Target Device input

2

or

Device Template

Native config + NED

Match source input

3 When native config and NED are provided

4 Assembly of dummy Device Group

Extract all the devices and merge them into a new dummy Device Group

Merge the individual devices into a new Dummy Device Group

1

Device Groups

1+ Devices/ All Devices

Target Device input

2

or

Device Template

Native config + NED

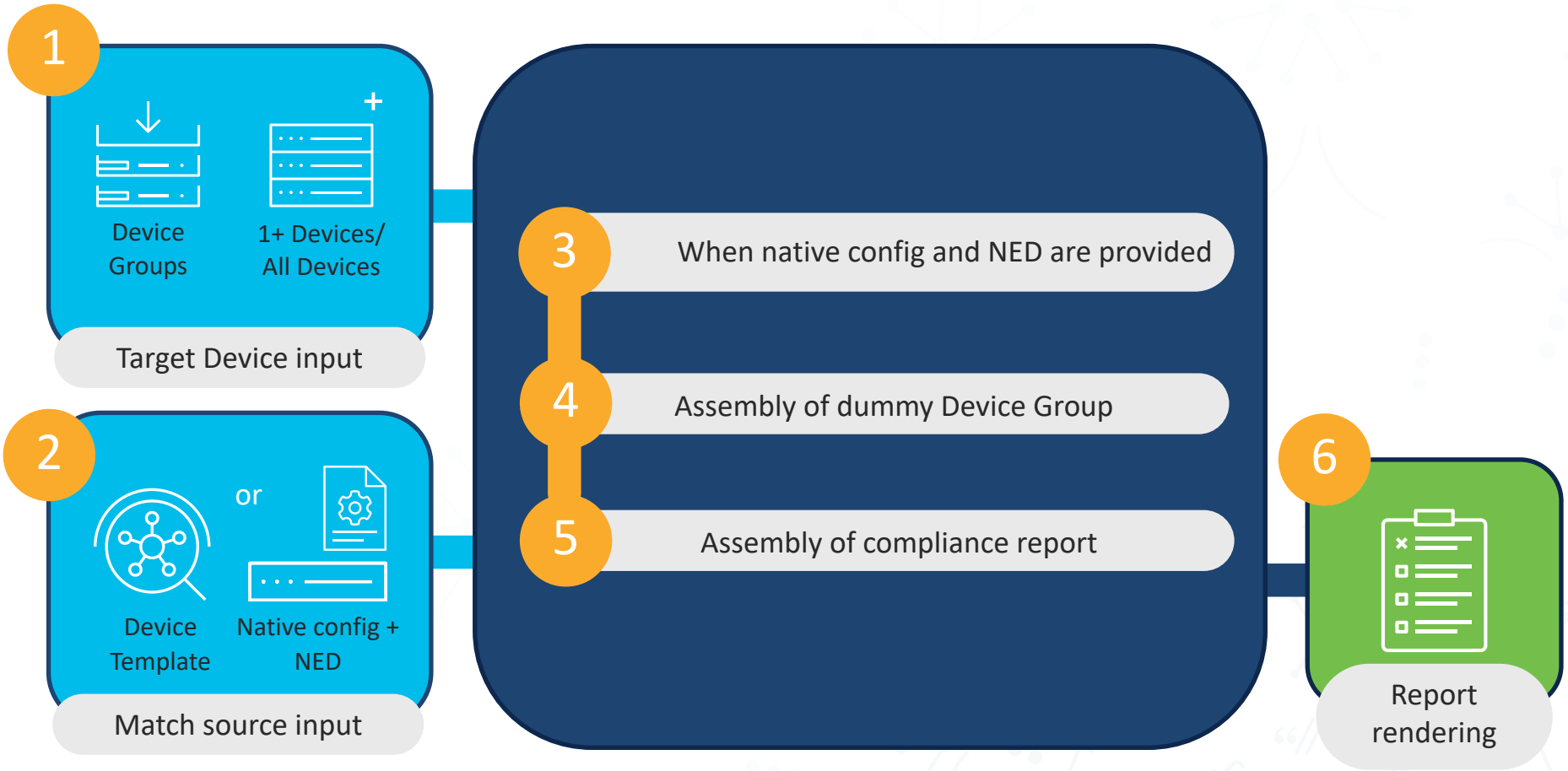
Match source input

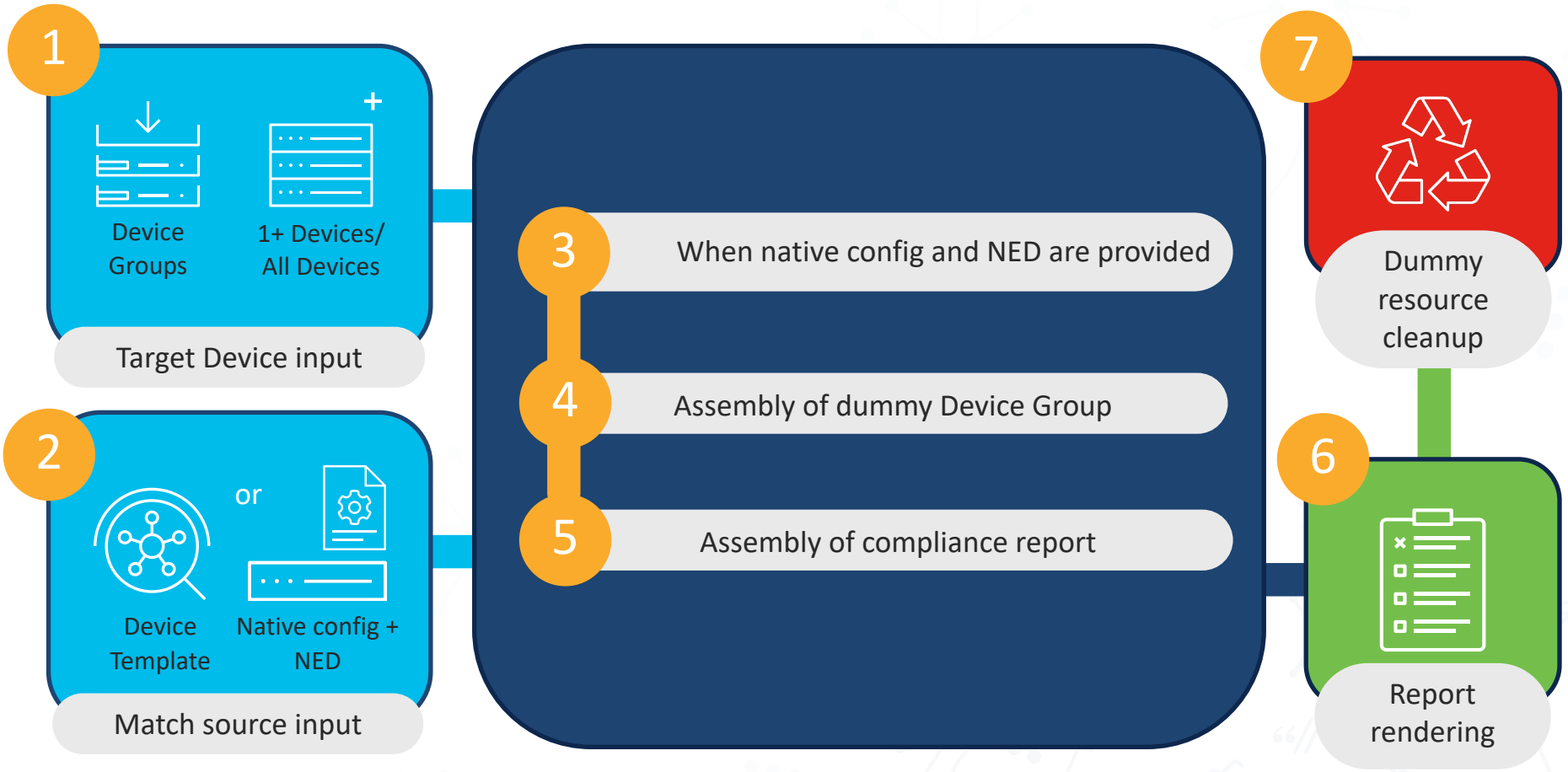
3 When native config and NED are provided

4 Assembly of dummy Device Group

5 Assembly of compliance report

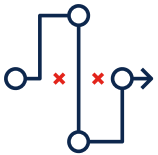
Create a dummy Compliance Report with all the nodes previously created


1
Device Groups
+
1+ Devices/ All Devices
Target Device input
2
or
Device Template
Native config + NED
Match source input
3
When native config and NED are provided
4
Assembly of dummy Device Group
5
Assembly of compliance report
6
Report rendering

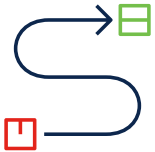

1
Device Groups
+
1+ Devices/ All Devices
Target Device input
2
or
Device Template
Native config + NED
Match source input
3
When native config and NED are provided
4
Assembly of dummy Device Group
5
Assembly of compliance report
7
Dummy resource cleanup
6
Report rendering

First approach - Technical Debt

-”Implied cost of additional rework caused by choosing a limited approach rather than a more complex one (that would take longer)”

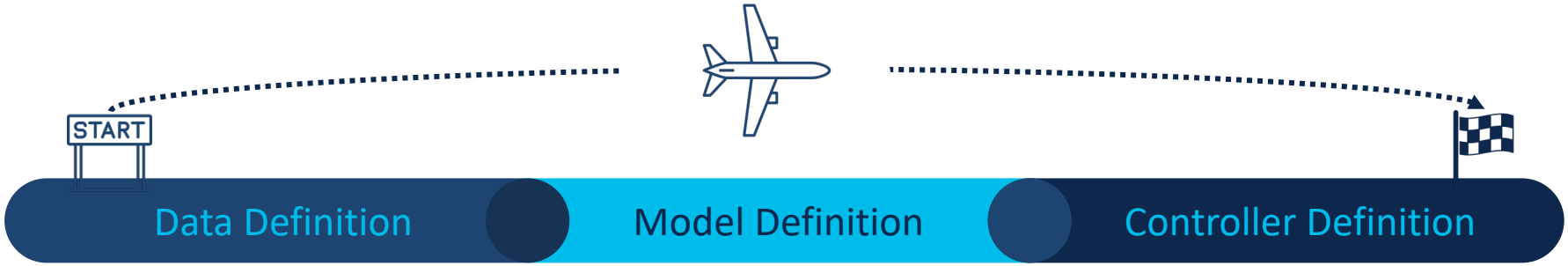


Tightly coupled elements within the code:
Single changes would affect the rest of the functions



Debugging and addition of new features under tight deadlines became a difficult task

Second approach - Clean code refactoring



Which attributes would the nodes (devices, templates, etc) have if they were entirely defined with code?

Which actions (methods) would each node be able to perform in NSO?

How would these nodes interact with each other in order to render a compliance report?

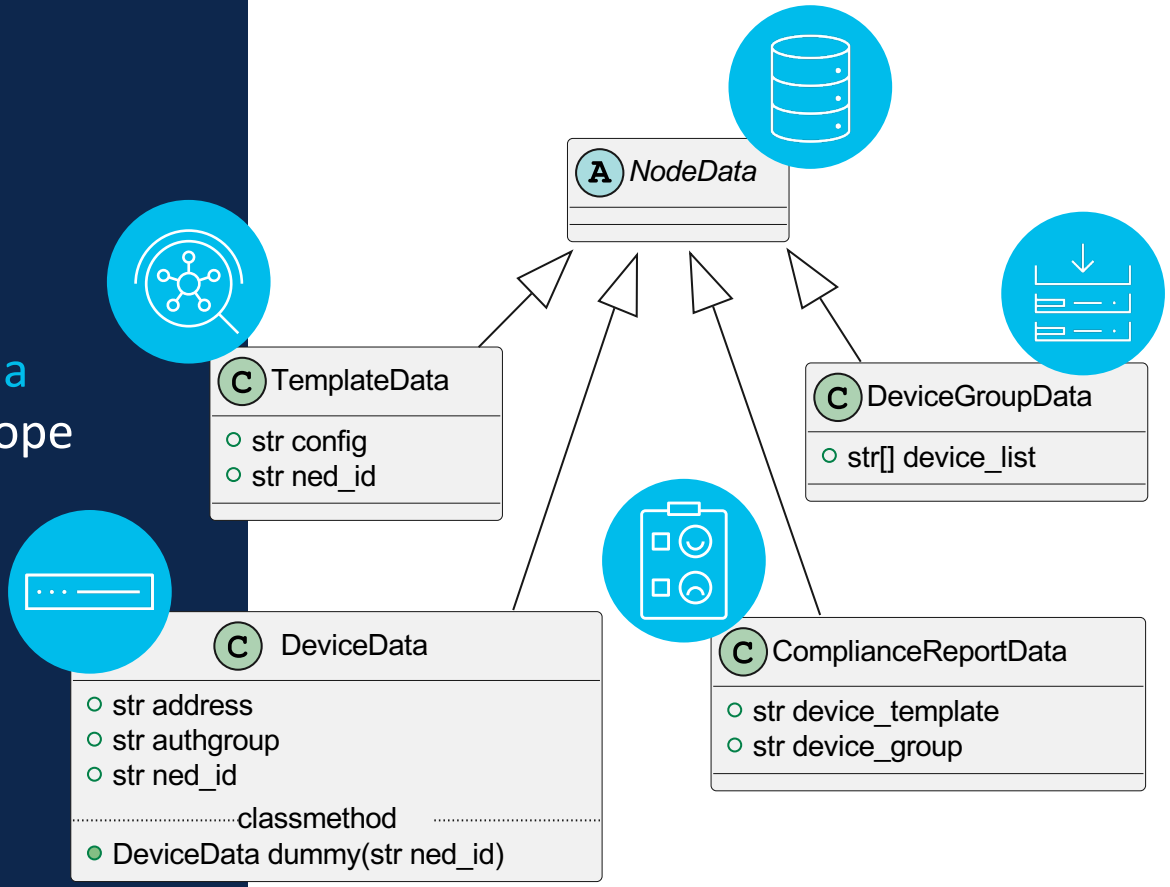
Data Definition

- Minimum attributes to describe each element as a node in NSO under the scope of our action package

Example:

Device

= IP address + Authentication group + ned ID



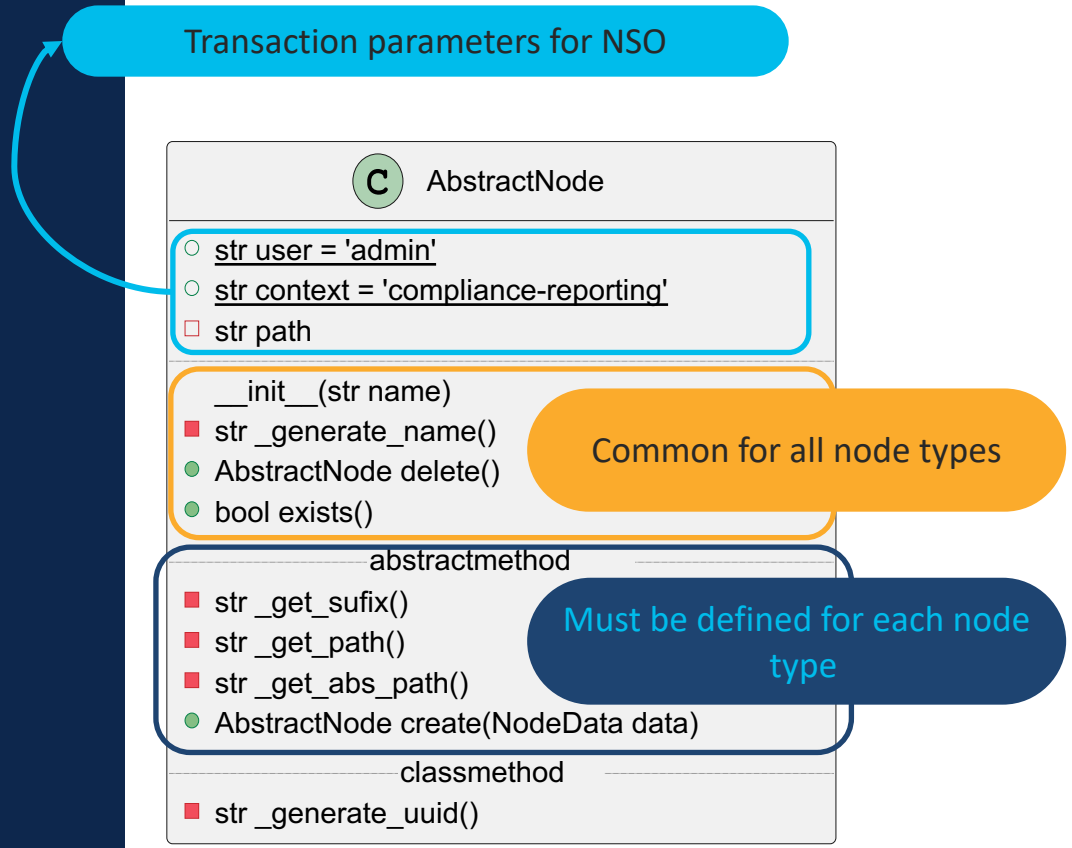
```
from abc import ABC
from dataclasses import dataclass

class NodeData(ABC):
    """
    Dataclass abstract node
    """
```

```
@dataclass
class DeviceData(NodeData):
    address: str
    authgroup: str
    ned_id: str
```

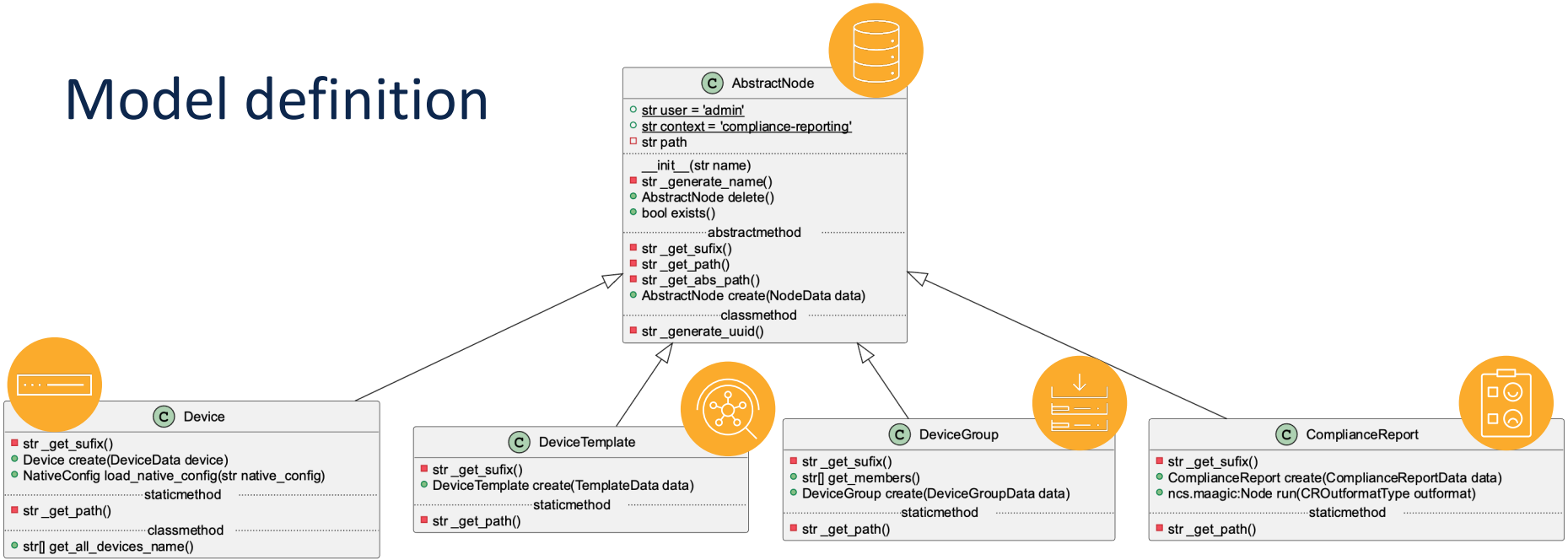
```
@dataclass
class ComplianceReportData(NodeData):
    device_template: str
    device_group: str
```

Model definition

- Minimum methods of an NSO node under the scope of our action package
- Macro-to-micro: This is an abstraction of the essential methods for creation, deletion, and checking of existence



Model definition



- All the children nodes implement their own methods for creation, get_suffix(), and get_path()
- Some have specific methods. For example, DeviceGroup must be able to return the devices which contains, and ComplianceReport must be able to render the report

```
class AbstractNode(ABC):
```

This is reused by all children

```
def exists(self) -> bool:

ncs.maagic.get_node(transaction,
self._get_abs_path())
```

```
def _get_abs_path(self):

return
f'{self._get_path()}{{{self.name}}}'
```

This must be implemented by every child

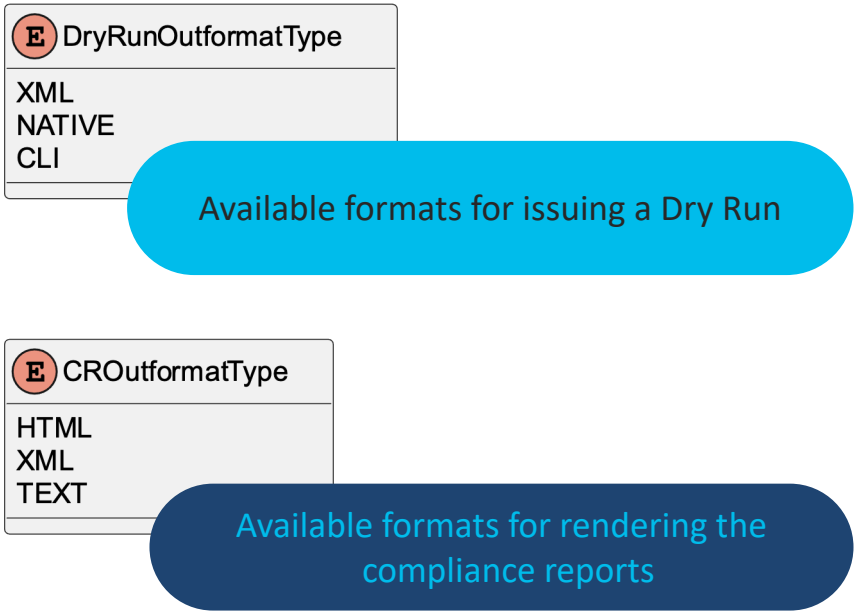
```
@abstractmethod

def _get_path():
```

```
class Device(AbstractNode):

@staticmethod

def _get_path():

  return '/devices/device’
```

. . .

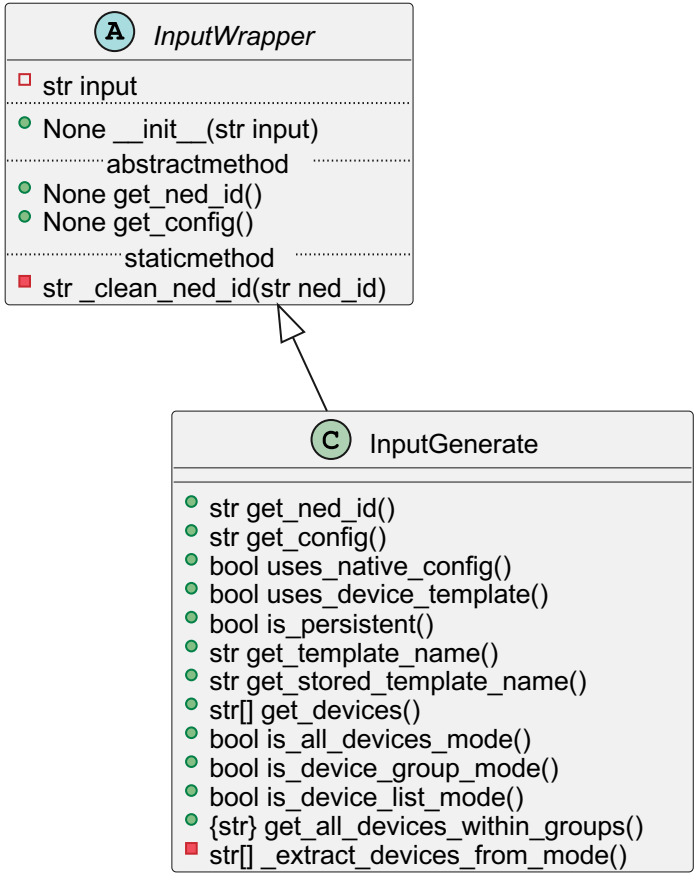
Model definition

- The usage of enumerations reduces the risk of typos or value inconsistency across the different models



Controller Definition

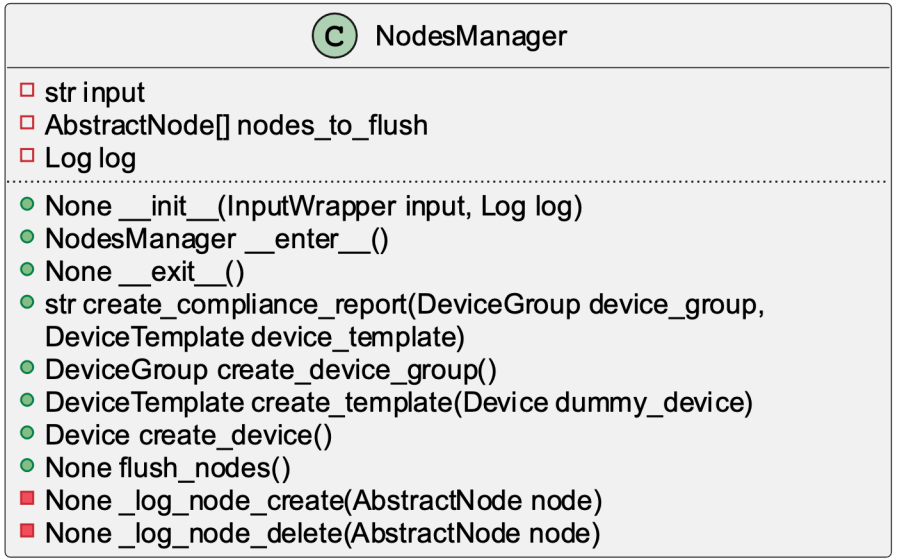
- Access to input data in a programmatic, standardized approach
- Instead of reading values directly from YANG, these controllers provide the data in a formatted and compliant way



Controller, give me all the devices from the YANG input please

Controller Definition

- Management of the node lifecycle through the execution of the action package
- Creation, execution and deletion of the different nodes



Controller, create a dummy device please

Controller, render the compliance report please

Controller, delete all dummy nodes please

```
class NodesManager():
```

. . .

Now this controller can be instantiated as a context manager

```
def __enter__(self):
 return self
```

When its lifespan is over, it will automatically flush all the dummy nodes

```
def __exit__(self, exc_type,
exc_val, exc_tb):
 self.flush_nodes()
```

main.py

```
class Generate(Action):

@Action.action

def cb_action(self, uinfo, name, kp,
input, output, trans):

with NodesManager(input, self.log)
as nodes_manager:

    if input.uses_native_config():

    . . .
```

Demo

Enhanced Compliance Reporting utility

Thank you!

Contact us: alfsando@cisco.com | cealves@cisco.com

The bridge to possible