XML data validation and Python tools for QE

Davide Brunato

brunato@sissa.it
SISSA

Summary

- A Python library for XML data validation and decoding
- Python post-processing tools for QE

XML

- A markup language defined with SGML (1997)
- XML is a way of structuring data, XML is a data format
- XML is also a language that you use to create other languages
- An XML file (document) is a tree
- XML libraries implementations use different API standards:
  - W3C Document Object Model (**DOM**)
  - Simple API for XML (**SAX**)
  - **Element Tree**

W3C XML Schema

- Widely used standard for XML data files
- Two versions:
  - 1.0 (2001: 1st edition – 2004: 2nd edition)
  - 1.1 (2012)
- Backward compatibility from 1.0 to 1.1:
  - An XSD 1.0 schema is also a valid 1.1 schema
- Pro:
  - Popularity, flexibility, powerful, XML based
- Cons:
  - More complex than other XML schema standards

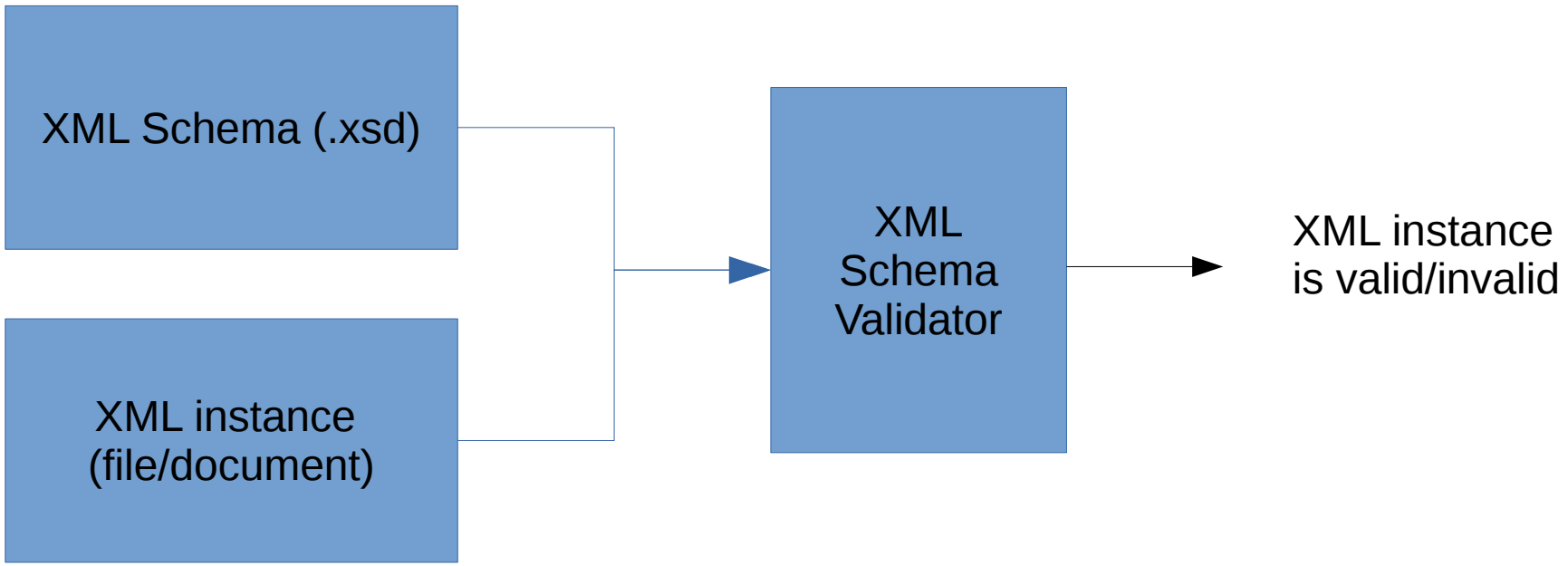
XML libraries for Python

- Standard libraries:
  - **xml.dom** and **xml.minidom**
  - **xml.sax**
  - **xml.etree**
- **lxml** library:
  - Pythonic bindings for C libraries *libxml2* and *libxslt*
  - Provides extended Element Tree API
  - **XML validation** (*not available with std. libraries ...*):

```
>>> root = etree.fromstring("<a>no int</a>", parser)
Traceback (most recent call last):
lxml.etree.XMLSyntaxError: Element 'a': 'no int' is not a valid value of
the atomic type 'xs:integer'...
```

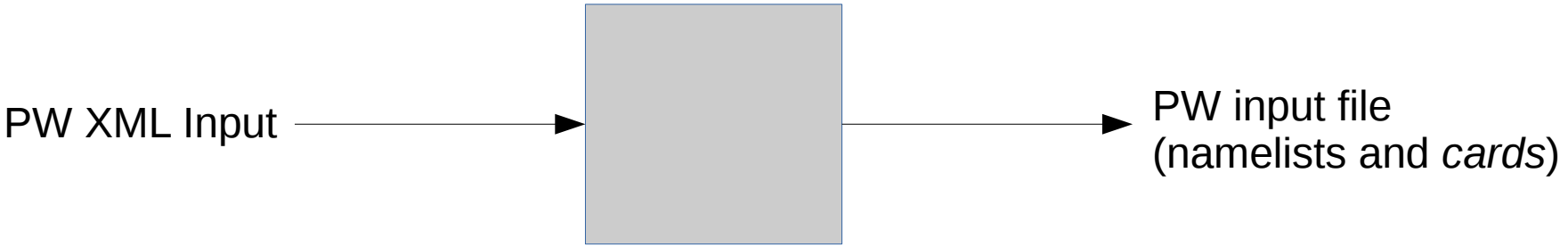
XML data validation

- The need of a *more formal* XML data for QE input and output
- Verify the product of simulation tasks



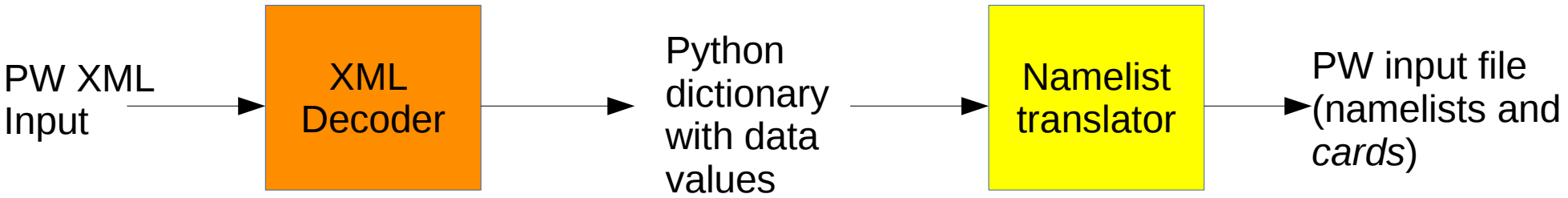
PW input and output

- Converting a XML input into Fortran's *namelist* input
  - Take over a prototype written by Giovanni Borghi for a first version of the PW schema
  - The prototype uses the lxml library for XML parsing and validation



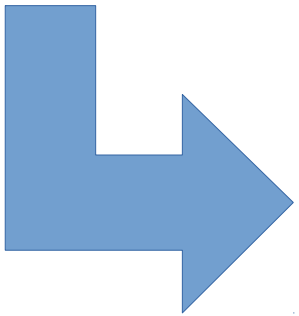
PW input translation

- The problem was decomposed in two parts:
  - Decoding the XML data input into a Python dictionary with values of specific type (str, int, float, bool)
  - Translate the XML input equivalent dictionary to a PW input file (namelists and QE *cards*)



XML decoding issue

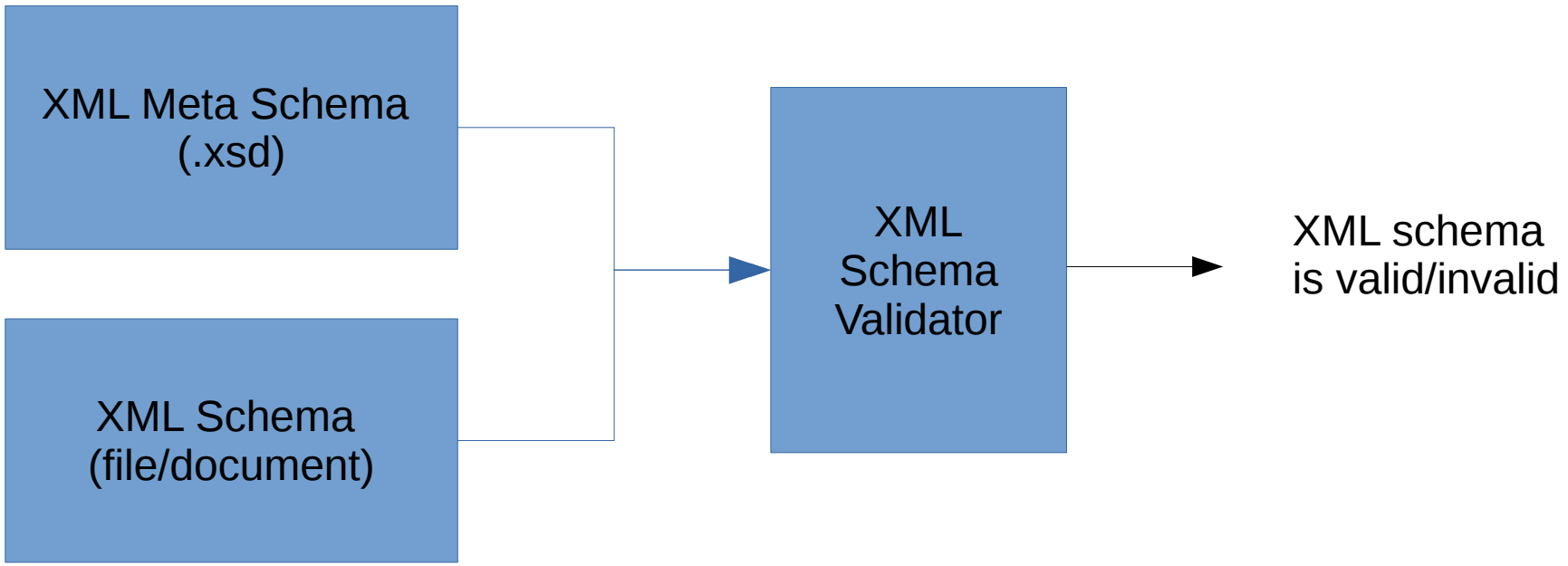
- Require a parse of the XSD schema a structure of XSD **types**, **elements** and **attributes**
- The lxml Element Tree API is almost useless:
  - `lxml.XMLSchema` class not include API for access to XSD declarations
- Also other available Python packages are useless



Create a new Python library for decoding … encoding … *validating* XML

XML meta-schema validation

- Use the same validators to verify the XSD schema with a XSD metaschema
  - An idea taken from *jsonschema* Python package



Python ingredients used for XSD

- OOP for XSD declarations
- Closures
- Generators
- Factories + keyword arguments
- Decorators

xmlschema package

- On GitHub since last september:
  - https://github.com/brunato/xmlschema
- License: MIT
- Status:
  - Beta, first release previewed for the end of january (regex)
  - Now cover XSD 1.0, next stable version will cover also 1.1
- Package:
  - Available into the Python Package Index (*tar* and *wheel*)
- Compatibility:
  - Python 2.7 and Python 3.4+
- About ~2400 lines of code
- Unit testing with 100+ XSD schemas:

Example: convert to dict

```
>>> import xmlschema
>>> xd = xmlschema.XMLSchema('collection.xsd')
>>> xd.to_dict('collection.xml')
{'{http://example.com/ns/collection}collection': {'object':
{'id': 'b0836217462', 'position': 1, 'available': True,
'estimation': Decimal('10000.00'), 'author': {'id': 'CMS',
'name': 'Pierre-Auguste Renoir', 'dead': '1919-12-03',
'qualification': 'painter', 'born': '1841-02-25'}, 'title':
'The Umbrellas'}, '{http://www.w3.org/2001/XMLSchema-
instance}schemaLocation':
'/home/brunato/Development/projects/xmlschema/xmlschema/tes
ts/examples/collection/collection.xsd'}}
```

Python Tools for QE

- Experimental **postqe** package (august 2016):
  - Developed using with f2py (Numpy) and Wx (graphical library)
  - Use also xmlschema for XML data decoding
- Collected requirements:
  - Usable as a library with API
  - Interface for QE tools (pp, bands, dos …)
- Proposal:
  - Usable both as a library and from command line
  - Split the graphical part(s) in dependent packages (eg. postqe-Wx)
  - Command line usage with only one command and many subcommands:

```
# postqe pp args
# postqe bands args
```

    - Easy to do this with Python 2.7+ *argparse* library
  - Try to use Python C interface instead of f2py

Conclusions

- XML and XSD require a systematic approach:
  - Better if treated with a dedicated small library
  - To support multiple schemas and schema changes
  - Experiment a schema-based conversion to other formats (JSON, YAML …)
- postqe
  - Support also Python 2.7 or only Python 3?
  - Start with less, improve adding sub commands