volib
A Python implementation of the VO Data Modeling Language

Omar Laurino
SAO

Scope

Scope

- Provide VO developers with a framework for dealing with complex data models (client and server side) and create reusable software

Scope

- Provide VO developers with a framework for dealing with complex data models (client and server side) and create reusable software
- Provide VO users with friendly, extensible I/O libraries while shielding them from the technical details of the data models specs

Scope

- Provide VO developers with a framework for dealing with complex data models (client and server side) and create reusable software
- Provide VO users with friendly, extensible I/O libraries while shielding them from the technical details of the data models specs
- Stress on a clean, simple API. “Focus on the science, not on the specifications”

Scope

- Provide VO developers with a framework for dealing with complex data models (client and server side) and create reusable software
- Provide VO users with friendly, extensible I/O libraries while shielding them from the technical details of the data models specs
- Stress on a clean, simple API. “Focus on the science, not on the specifications”
- Code generation, if needed

Specifications

- **VO-DML** WD: Consistent language for describing data models
  - Machine-readability
  - Model reuse
  - Model extensibility
  - Portable identifiers
  - Backward compatibility with older serializations
- **Mapping Data Models to VOTable** WD: define a standard strategy for serializing any data model instance to VOTable

Features

- **VO-DML** WD: Consistent language for describing data models
  - Model reuse —> Reusable libraries
  - Model extensibility —> Inheritance/Polymorphism
  - Machine-readability —> Model-agnostic. Code generation.
  - Portable identifiers —> Context and Resolvers
  - Versioning
- **Mapping Data Models to VOTable** WD: define a standard strategy for serializing any data model instance to VOTable
  - VOTable I/O

Representing Data Models

```
class SkyCoordinate(DataType):
    vodml_id = 'source.stc.SkyCoordinate'

    longitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.longitude',
                          doc="""The longitude part of this position in units of degrees.""")

    latitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.latitude',
                          doc="""The latitude part of this position in units of degrees.""")

    error = Attribute(SkyError,
                          'source.stc.SkyCoordinate.error',
                          doc="""None""")

    frame = Reference(SkyCoordinateFrame,
                          'source.stc.SkyCoordinate.frame',
                          doc="""TODO : Missing description : please, update your UML model asap.""")
```

Representing Data Models

```
class SkyCoordinate(DataType):
    vodml_id = 'source.stc.SkyCoordinate'

    longitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.longitude',
                          doc="""The longitude part of this position in units of degrees.""")

    latitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.latitude',
                          doc="""The latitude part of this position in units of degrees.""")

    error = Attribute(SkyError,
                          'source.stc.SkyCoordinate.error',
                          doc="""None""")

    frame = Reference(SkyCoordinateFrame,
                          'source.stc.SkyCoordinate.frame',
                          doc="""TODO : Missing description : please, update your UML model asap.""")
```

VO-DML Types

Representing Data Models

```
class SkyCoordinate(DataType):
    vodml_id = 'source.stc.SkyCoordinate'

    longitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.longitude',
                          doc="""The longitude part of this position in units of degrees.""")

    latitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.latitude',
                          doc="""The latitude part of this position in units of degrees.""")

    error = Attribute(SkyError,
                          'source.stc.SkyCoordinate.error',
                          doc="""None""")

    frame = Reference(SkyCoordinateFrame,
                          'source.stc.SkyCoordinate.frame',
                          doc="""TODO : Missing description : please, update your UML model asap."""
```

VO-DML Types

Types from Imported Models

Representing Data Models

```
class SkyCoordinate(DataType):
    vodml_id = 'source.stc.SkyCoordinate'

    longitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.longitude',
                          doc="""The longitude part of this position in units of degrees.""")

    latitude = Attribute(ivoa_1_0.RealQuantity,
                         'source.stc.SkyCoordinate.latitude',
                         doc="""The latitude part of this position in units of degrees.""")

    error = Attribute(SkyError,
                      'source.stc.SkyCoordinate.error',
                      doc="""None""")

    frame = Reference(SkyCoordinateFrame,
                      'source.stc.SkyCoordinate.frame',
                      doc="""TODO : Missing description : please, update your UML model asap."""
```

VO-DML Types

Types from Imported Models

Types from Same Model

Representing Data Models

```
class SkyCoordinate(DataType):
    vodml_id = 'source.stc.SkyCoordinate'

    longitude = Attribute(ivoa_1_0.RealQuantity,
                          'source.stc.SkyCoordinate.longitude',
                          doc="""The longitude part of this position in units of degrees.""")

    latitude = Attribute(ivoa_1_0.RealQuantity,
                         'source.stc.SkyCoordinate.latitude',
                         doc="""The latitude part of this position in units of degrees.""")

    error = Attribute(SkyError,
                      'source.stc.SkyCoordinate.error',
                      doc="""None""")

    frame = Reference(SkyCoordinateFrame,
                      'source.stc.SkyCoordinate.frame',
                      doc="""TODO : Missing description : please, update your UML model asap.""")
```

VO-DML Types

Types from Imported Models

Types from Same Model

Declarative approach allows users/developers/providers to easily create new classes and serialize them as compliant VOTable and VO-DML/XML

Inheritance/Polymorphism

```
class SkyError(DataType):
    vodml_id = 'source.stc.SkyError'

class CircleError(SkyError):
    vodml_id = 'source.stc.CircleError'

    radius = Attribute(ivoa_1_0.real,
                       'source.stc.CircleError.radius',
                       doc="""TODO : Missing description : please, update your UML model asap

class GenericEllipse(SkyError):
    vodml_id = 'source.stc.GenericEllipse'

    major = Attribute(ivoa_1_0.real,
                      'GenericEllipse.major',
                      doc="""major axis of error ellipse""")

    minor = Attribute(ivoa_1_0.real,
                      'source.stc.GenericEllipse.minor',
                      doc="""TODO : Missing description : please, update your UML model asap
```

Inheritance/Polymorphism

```
class SkyError(DataType):
    vodml_id = 'source.stc.SkyError'

class CircleError(SkyError):
    vodml_id = 'source.stc.CircleError'

    radius = Attribute(ivoa_1_0.real,
                       'source.stc.CircleError.radius',
                       doc="""TODO : Missing description : please, update your UML model asap

class GenericEllipse(SkyError):
    vodml_id = 'source.stc.GenericEllipse'

    major = Attribute(ivoa_1_0.real,
                      'GenericEllipse.major',
                      doc="""major axis of error ellipse""")

    minor = Attribute(ivoa_1_0.real,
                      'source.stc.GenericEllipse.minor',
                      doc="""TODO : Missing description : please, update your UML model asap
```

Parent Class

Inheritance/Polymorphism

```
class SkyError(DataType):
    vodml_id = 'source.stc.SkyError'

class CircleError(SkyError):
    vodml_id = 'source.stc.CircleError'

    radius = Attribute(ivoa_1_0.real,
                       'source.stc.CircleError.radius',
                       doc="""TODO : Missing description : please, update your UML model asap

class GenericEllipse(SkyError):
    vodml_id = 'source.stc.GenericEllipse'

    major = Attribute(ivoa_1_0.real,
                      'GenericEllipse.major',
                      doc="""major axis of error ellipse""")

    minor = Attribute(ivoa_1_0.real,
                      'source.stc.GenericEllipse.minor',
                      doc="""TODO : Missing description : please, update your UML model asap
```

Parent Class

Extensions

Inheritance/Polymorphism

Inheritance/Polymorphism

```
>>> from reference.ref_1_0.source.stc import CircleError, GenericEllipse
>>> error = CircleError()
>>> coord.error = error
>>> s.position = coord
>>> s.position.error.radius = 1
>>> print s.position.error.radius
1.0

>>> coord.error = GenericEllipse()
>>> coord.error.major = 0.1
>>> coord.error.minor = 0.1
>>> coord.error.pa = 20
>>> s.position.error
<reference.ref_1_0.source.stc.GenericEllipse object at 0x101228610>
>>> s.position.error.pa
20.0
```

Any children of SkyError can be used as coordinate errors

Inheritance/Polymorphism

```
>>> from reference.ref_1_0.source.stc import CircleError, GenericEllipse
>>> error = CircleError()
>>> coord.error = error
>>> s.position = coord
>>> s.position.error.radius = 1
```

```
>>> coord.error = Source()
ERROR:volib.model:Cannot cast value <reference.ref_1_0.source.Source object at 0x101228550> to datatype
```

```
>>> coord.error = GenericEllipse()
>>> coord.error.major = 0.1
>>> coord.error.minor = 0.1
>>> coord.error.pa = 20
>>> s.position.error
<reference.ref_1_0.source.stc.GenericEllipse object at 0x101228610>
>>> s.position.error.pa
20.0
```

Any children of SkyError can be used as coordinate errors

Inheritance/Polymorphism

```
>>> from reference.ref_1_0.source.stc import CircleError, GenericEllipse
>>> error = CircleError()
>>> class MyEllipseError(GenericEllipse):
...     def area(self):
...         import math
...         return math.pi*self.major*self.minor
>>> myerror = MyEllipseError()
>>> myerror.minor = 1.5
>>> myerror.major = 3.1
>>> coord.error = myerror
>>> s.position.error.area()
14.608405839192539
>>> s.position.error.pa
20.0
```

Any children of SkyError can be used as coordinate errors

Inheritance/Polymorphism

```
>>> from reference.ref_1_0.source.stc import CircleError, GenericEllipse
>>> error = CircleError()
```

```
>>> class MyEllipseError(GenericEllipse):
...     def area(self):
...         import math
...         return math.pi*self.major*self.minor
>>> myerror = MyEllipseError()
>>> myerror.minor = 1.5
>>> myerror.major = 3.1
>>> coord.error = myerror
>>> s.position.error.area()
14.608405839192539
```

This custom class can be serialized just as a GenericEllipse

```
20.0
```

Any children of SkyError can be used as coordinate errors

Some useful properties

```
>>> from reference.ref_1_0.source import Source, SourceClassification
>>> s = Source()
>>> s.classification = SourceClassification.STAR
>>> s.classification
<volib.model.Enum object at 0x101220c10>
>>> print s.classification
star
```

Some useful properties

```
>>> from reference.ref_1_0.source import Source, SourceClassification
>>> s.classification = 'galaxy'
>>> s.classification
<volib.model.Enum object at 0x101220c50>
>>> print s.classification
galaxy

>>> s.classification = 'foo'
Traceback (most recent call last):
    ...
TypeError: Wrong value for Enum SourceClassification. Valid values: SourceClassification.PLANET or "planet",
    SourceClassification.AGN or "AGN", SourceClassification.STAR or "star",
    SourceClassification.GALAXY or "galaxy", SourceClassification.UNKNOWN or "unknown"
```

Some useful properties

```
>>> from reference.ref_1_0.source import Source, SourceClassification
```

```
>>> s.classification = 'galaxy'
>>> s.classification
```

```
>>> from reference.ref_1_0.source.stc import SkyCoordinate
>>> coord = SkyCoordinate()
>>> print coord.longitude
None
```

Once instantiated, the SkyCoordinate object have attributes with no values. However, if we try to assign a value to the attribute:

```
>>> coord.longitude = 5
>>> coord.longitude.value
5.0
```

TO DO

TO DO

- Updates from latest draft, abstract types

TO DO

- Updates from latest draft, abstract types
- VOTable (de-)serialization

TO DO

- Updates from latest draft, abstract types
- VOTable (de-)serialization
- VO-DML/XML generation from Python classes

TO DO

- Updates from latest draft, abstract types
- VOTable (de-)serialization
- VO-DML/XML generation from Python classes
- Implement VO-DML Quantities as Astropy Quantities (units conversions, quantity comparison/combinations)

TO DO

- Updates from latest draft, abstract types
- VOTable (de-)serialization
- VO-DML/XML generation from Python classes
- Implement VO-DML Quantities as Astropy Quantities (units conversions, quantity comparison/combinations)
- Data binding/Callbacks - Glue integration? (see C. Beaumont talk @ADASS about hackable interfaces)

Summary

Summary

- Interoperable VO “pickle”

Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:

Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter

Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter
  - decoupled from implementation

Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter
  - decoupled from implementation
  - concentrate on science, not VO specs.

Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter
  - decoupled from implementation
  - concentrate on science, not VO specs.
  - no need to be a Python guru to create your own interoperable classes/ models.

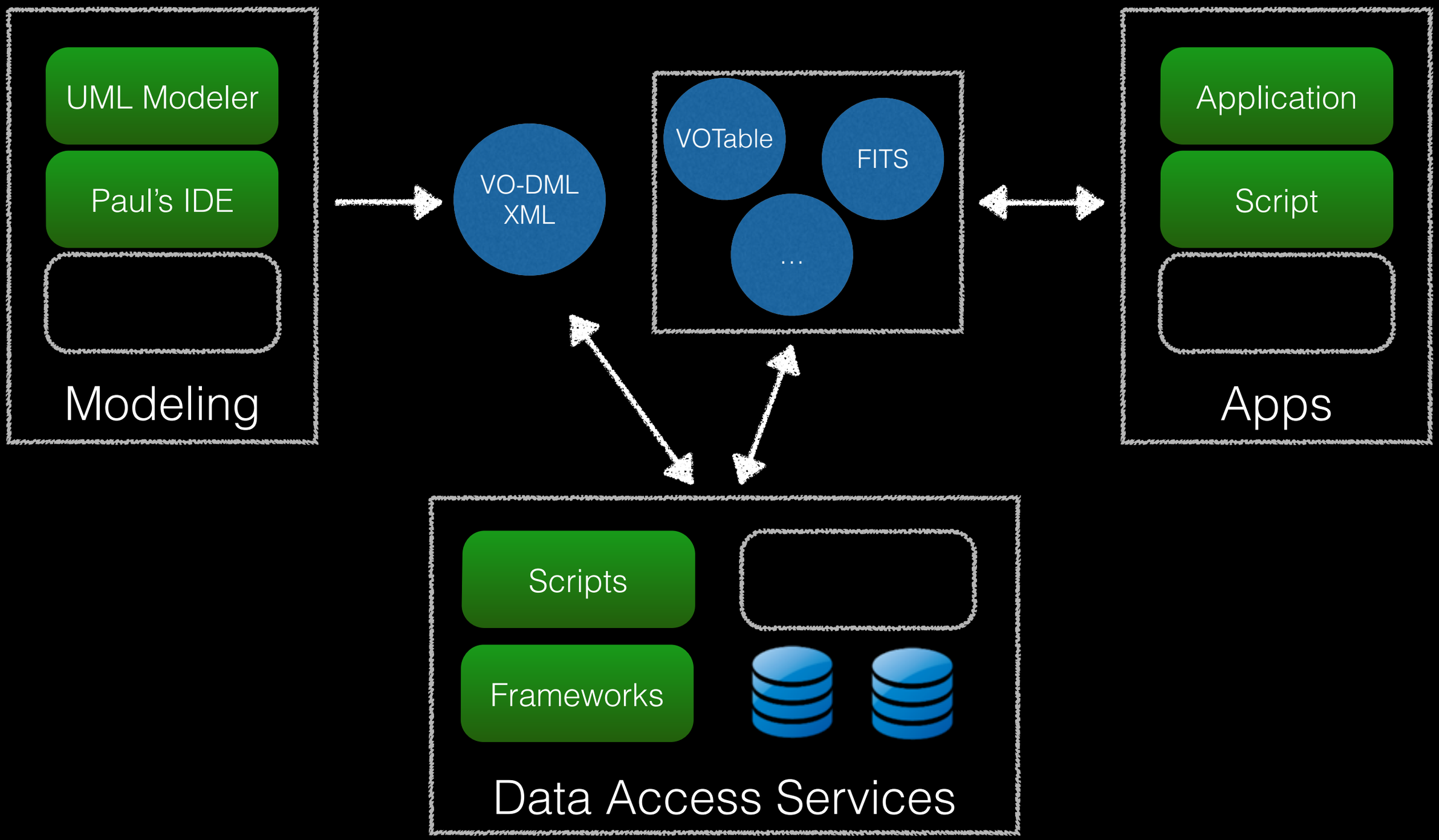
Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter
  - decoupled from implementation
  - concentrate on science, not VO specs.
  - no need to be a Python guru to create your own interoperable classes/ models.
- Model/Serialization separation of concerns:

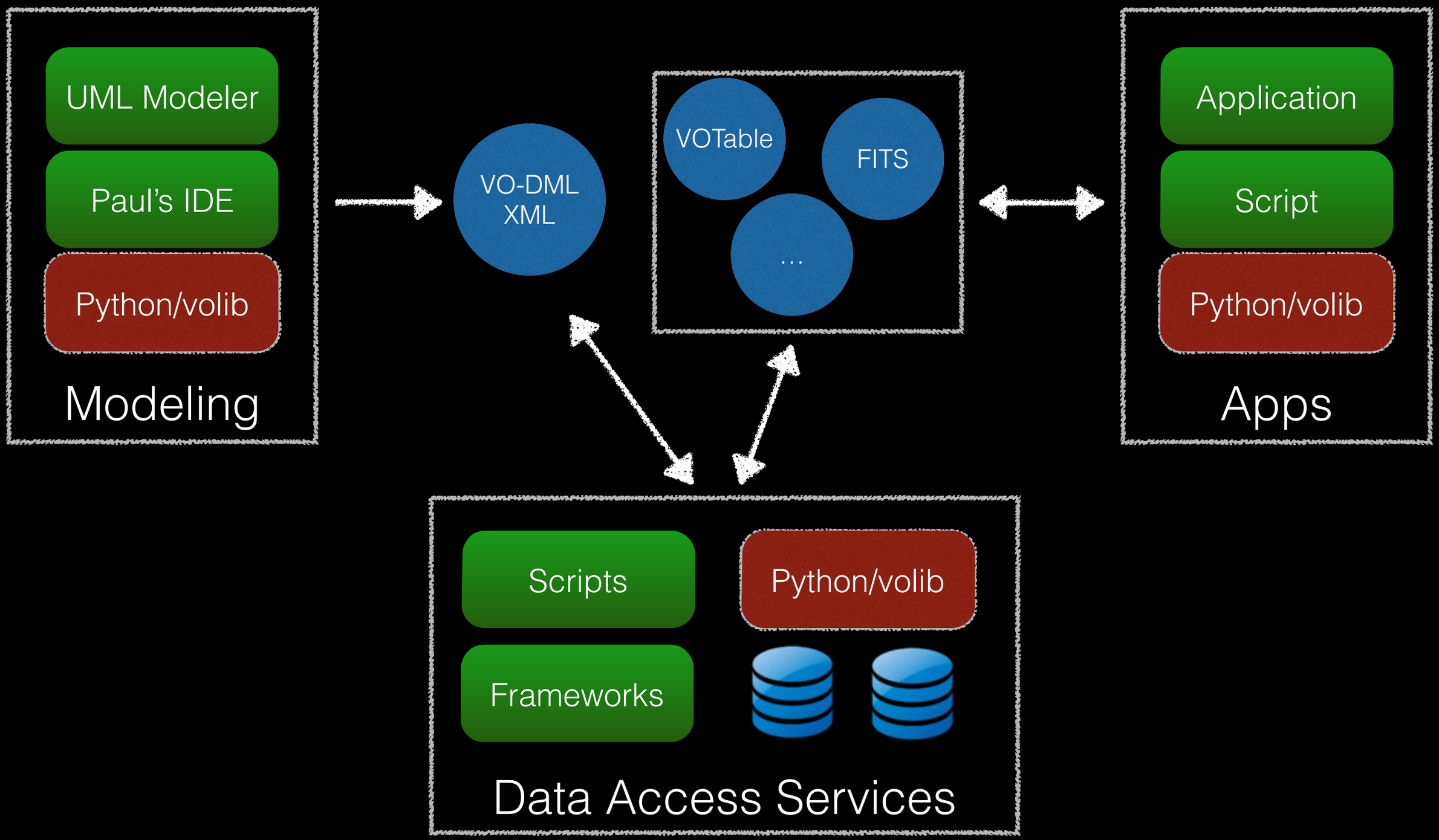
Summary

- Interoperable VO “pickle”
- Declarative language. Just like Django, SQLAlchemy (or Hibernate), etc:
  - no __init__, no clutter
  - decoupled from implementation
  - concentrate on science, not VO specs.
  - no need to be a Python guru to create your own interoperable classes/ models.
- Model/Serialization separation of concerns:
  - Drivers can provide alternative representation formats (e.g. HDF5, FITS)

The Big Picture



The Big Picture



Fork me on GitHub

Thank you!

And thanks to Markus Demleitner for his input.

More information on VO-DML, with examples
in DM2, Saturday @ 11
https://github.com/olaurino/volib