(application specific) data (models)

- framing
  - issues
  - requirements
- what sort of models do we need to worry about?
- RDF/OWL
- XML/XSD

Framing the issue

- Many RIF applications will involve rulesets that process external data. How will they:
  - identify the data set?
  - identify the data model (schema, entailment regime) associated with the data set?
  - access the data?
  - access the data model?

  Not all of these may be required
- Some applications may transfer the data as RIF and no external data support is required
- Use cases say this is not enough

Requirements (to be confirmed)

- RIF will support rulesets which access data in non-RIF formats
- These include at least:
  - XML constrained by an XML Schema
  - RDF, augmented by RDFS/OWL-full ontology
  - *object data model* (come back to this one)
- Note: some rulesets may access more than one such model simultaneously (XML embedded in RDF)

Non-requirements (to be confirmed)

[May be side-effects but not explicit goals]

- exchange of rulesets between applications that use different data models
- data model interchange

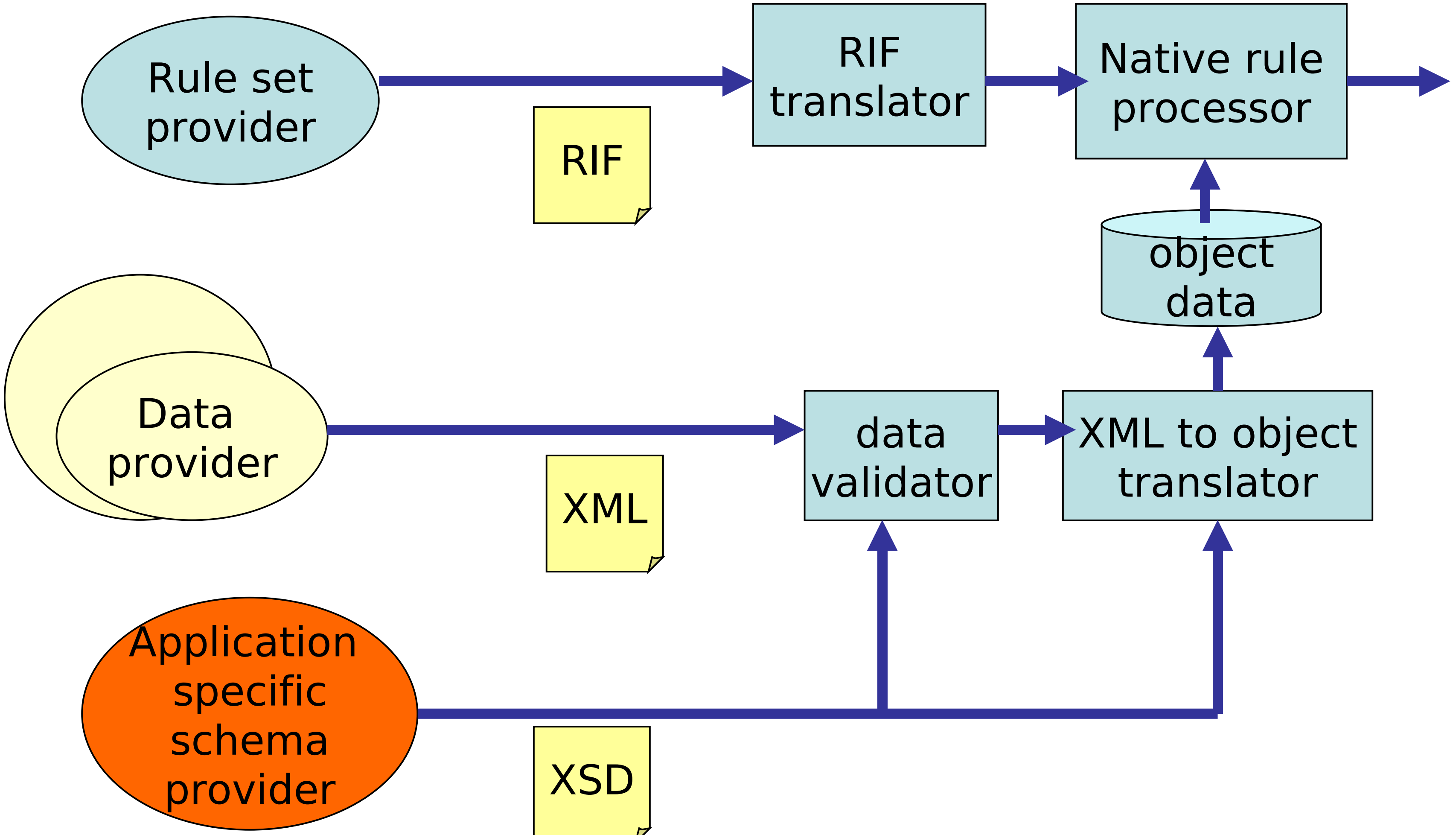
Object data models?

- Primary use case mentioned so far is:
  - data model defined in XML Schema
  - data is exchanged via XML
  - schema mapped to object model (JAXB etc)
  - rules access the corresponding object model
- Is that it for this phase?
  - do we need support for MOF + XMI, ODM ..?
  - my best guess is “no”

RDF/OWL

- know how RDF data (and thus RDFS, OWL) can be accessed from RIF
- some open issues
  - just access as data (MK, Hassan) or explicit support for RDF semantics (Jos)
  - if support semantics need metadata to identify entailment regime
    solution to this one has already been proposed
- basically under control
  so let’s focus on XML Schema

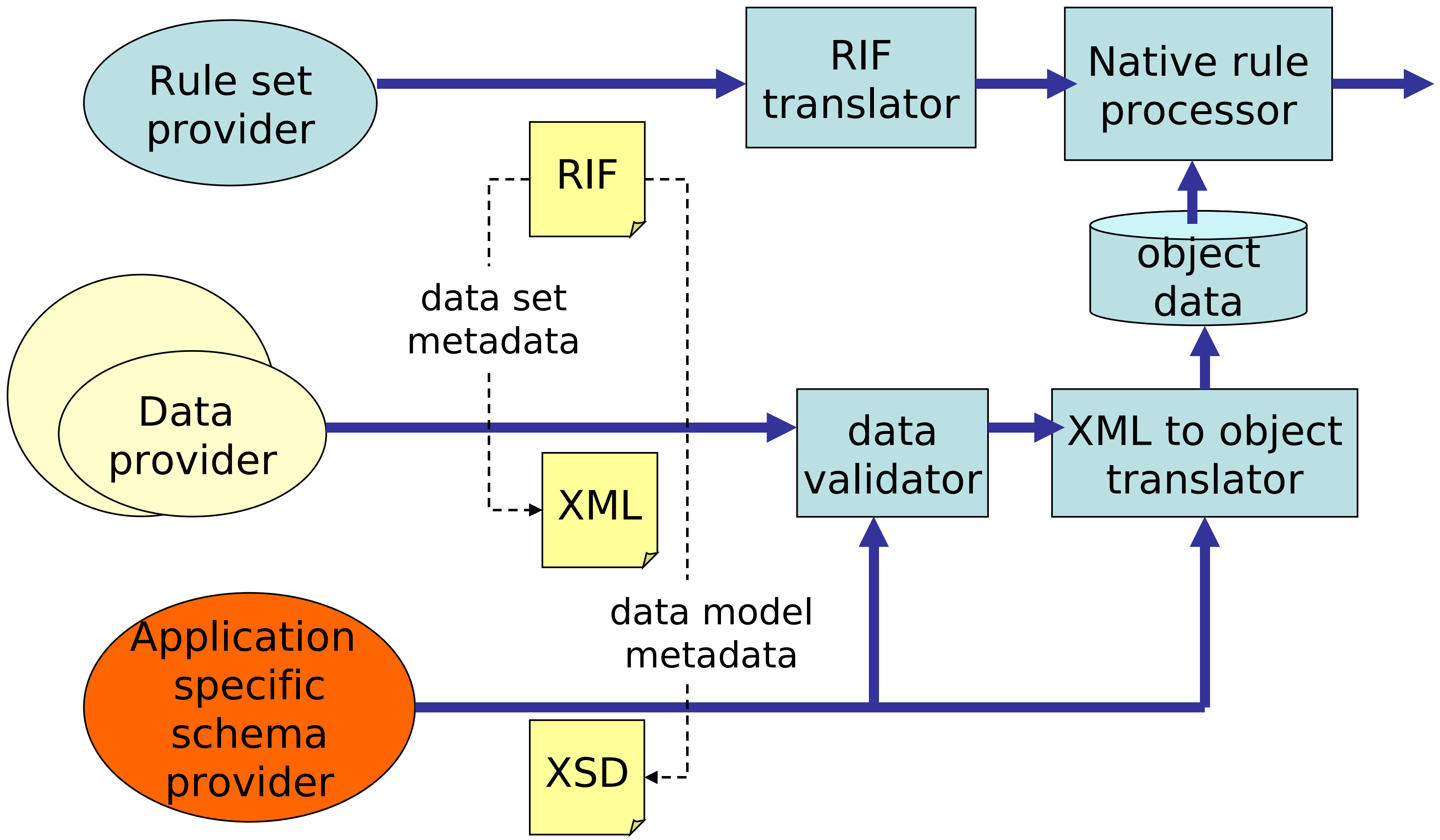
Typical XML processing model



So how to answer the original questions?

- data set identification
  - ruleset metadata using URIs to label fixed datasets
- data model identification
  - ruleset metadata (actually annotation on the data set metadata) using URIs to identify schema and data model class
- access the data
  - see later
- access the data schema
  - treat as data if necessary
  - not clear this is a requirement in the first place

Typical XML processing model



Example metadata (Turtle syntax)

```
[] a rif:RuleSet;
  rif:requiresDataSet [
     rdfs:label "order";
     rdfs:comment "The order to be processed" ;
     rif:dataModel <http://example.com/orderDataModel> ] .

<http://example.com/orderDataModel>
  a rif:XMLSchemaDataModel ;
  rdfs:comment "The id for an order schema agreed by
                 consortium";
  rif:schema <http://example.com/orderDataModel.xs> .
```

- Actual abstract, presentation and XML syntax for metadata yet to be defined

Accessing the data according to an external application specific schema

- proposal: single generic XML -> RIF mapping
  based on existing XML to object map (JAXB)
  - XML instance data mapping to a set of nested frames
  - frame type derived from schema complex type
  - slot name derived from schema element/attribute name
  - slot value is obvious mapping of primitive types or nested frames
  - frame id is URI (if element has an xml:id) or a gensymed constant

Deriving slot and type URIs

- if schema element (complex type, element, attribute) has **`sawsdl:modelReference`** annotation then use that
- otherwise form URI by concatenating schema URI with type/attribute/element name plus disambiguation for overlapping name spaces
  [details for how to do this exist (Gloze)]
- otherwise, if no schema, use rif:local names based on element/attribute name assuming striping?

Alternatives

- translate data (and data model) to RIF at provider side
  - consumer has an easy life
  - provider can translate how they want to
  - no commonality across users of related schemas
  - need data in native form anyway
- single metamodel (MOF, KM3)
  - define RIF mapping for that once
  - then each schema has to be mapped to this common metamodel

Alternatives

- translate data model itself to RIF as well
- ...