RULES FEST

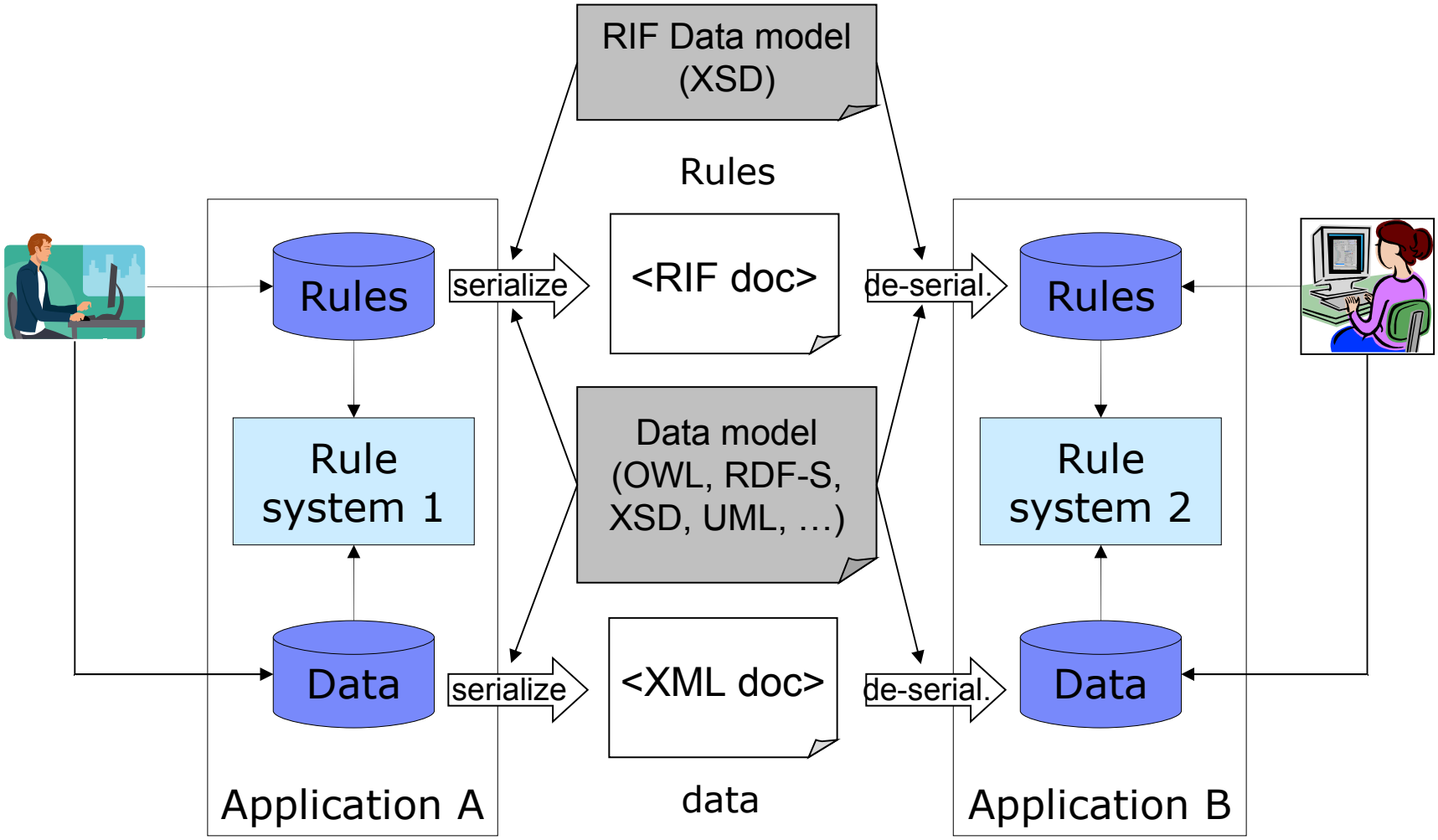
Choosing Data For Rule Interchange

Christian de Sainte Marie

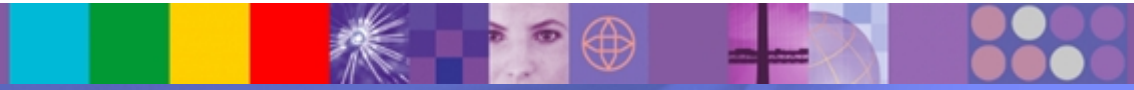
Rule and decision management standards

IBM ILOG

Rule interchange (rules as data)



A brief introduction to W3C RIF

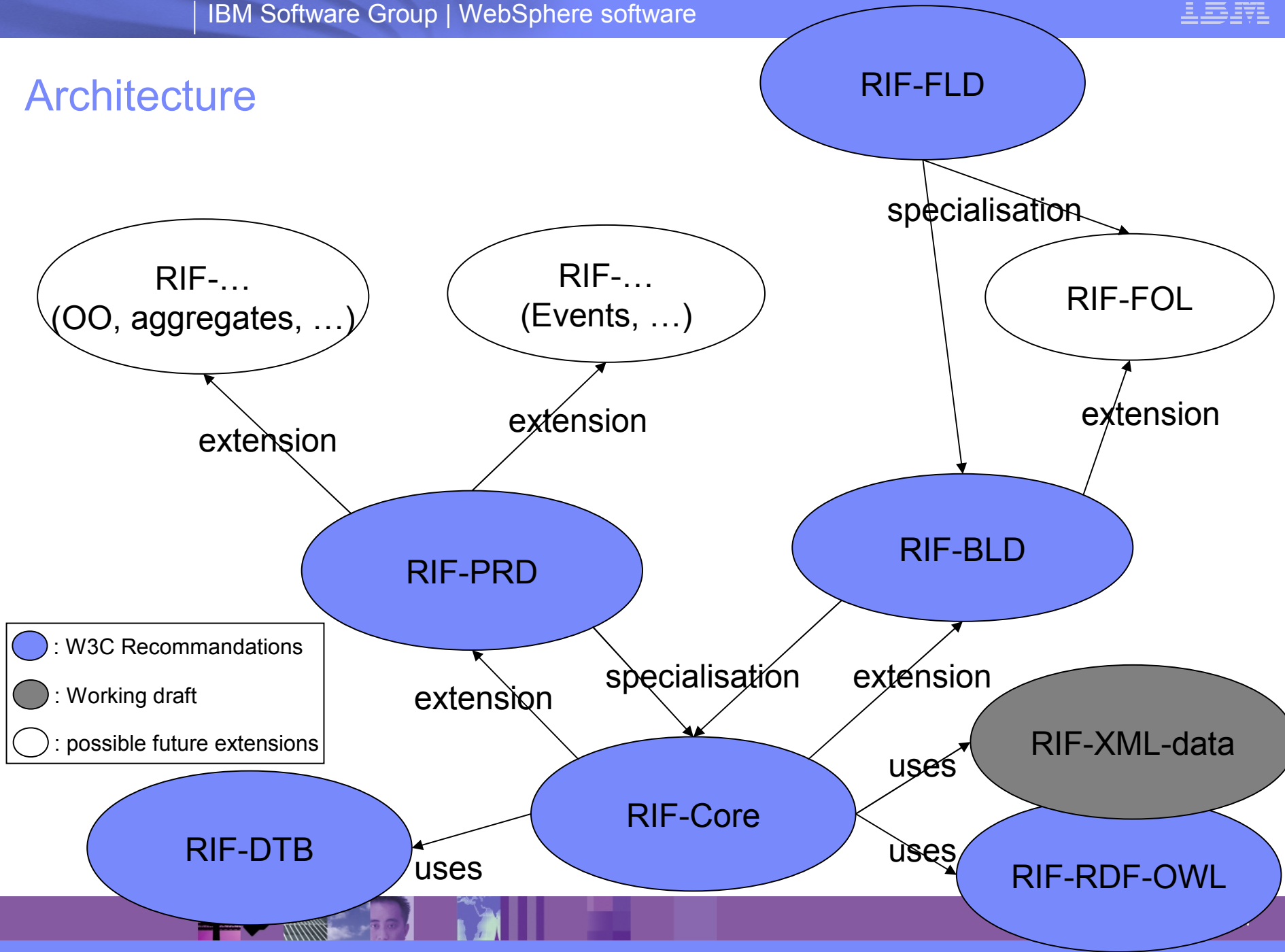


WebSphere software

W3C rule interchange format (RIF)

- A standard XML serialization for rules
  - with standard semantics
  - allowing rules written for one application to be published, shared, and re-used in other applications and other rule engines (between rule languages with compatible semantics)
  - A rule is (just another) data item
- A set of dialects and an extensible framework, encouraging interoperability
  - A RIF-Core dialect with the expressive power of Datalog
  - Two extensions: Basic Logic Dialect and Production Rule Dialect
  - An extensive set of built-in data types, functions and operators
  - Combination with RDF, OWL and XML/XSD data sources and data models
- A semantic Web standard
- A W3C recommendation
  - An open standard, endorsed and supported by an well-established industry consortium
  - and a well-defined patent policy

Architecture



W3C RIF WG published documents

- W3C Recommendations
  - rdf:plainLiteral (common with OWL WG)
  - RIF-Core: RIF Core dialect
  - RIF-BLD: RIF basic logic dialect
  - RIF-FLD: RIF framework for logic dialects
  - RIF-PRD: RIF production rule dialect
  - RIF-DTB: RIF data types and builtins
  - RIF-RDF-OWL: RIF RDF and OWL compatibility
- REC-track specification
  - RIF-XML-data: RIF combination with XML data and schema
- Working group notes
  - RIF-UCR: RIF use cases and requirements
  - RIF-Tests: RIF Test Cases
  - RIF-Overview: Overview of RIF documentation
  - RIF-OWL2-RL: RIF implementation of OWL2 RL
  - RIFinRDF:

Web site: http://www.w3.org/2005/rules/wiki/RIF_Working_Group

W3C RIF Core dialect (RIF-Core)

- Expressiveness ≈ Datalog
  - Integrity constraints on relational data bases
  - Simple rules
    - No negative conditions
    - Can only assert relations and assign values to properties of objects
- Minimal Herbrant model semantics

The status of a customer who has purchased for over $1000 in the past three months, must be: Gold

```
Forall ?c
If Exists ?v1 ?v2 ?v3
        (And (QuarterMonthlyPurchase(?c ?v1 ?v2 ?v3)
                func:numeric-greater-than(fun:numeric-add(?v1 ?v2 ?v3) 1000)
Then CustomerStatus(?c "Gold")
```

W3C RIF Basic Logic Dialect (RIF-BLD)

- Extends RIF-Core
  - logic functions
  - subclass formulas
  - any atomic statement in the conclusion, including equality, membership, subclass
- Expressiveness: simple logic programs
  - No negation
- Standard first-order model-theoretic semantics with function and equality

The spouse of one's spouse is oneself

Forall ?x ?y ?z,

If ?x [spouse -> ?y] and ?y [spouse -> ?z] then ?x = ?z

W3C Production Rule Dialect (RIF-PRD)

- Extends RIF-Core
  - Negation
  - Assert, Retract, Modify fact
  - Execute function call
  - Add new object
- Expressiveness ≈ simple BRMS
- Standard RETE-style operational semantics

A customer who has purchased for over $1000 in the past three months, reaches status: gold

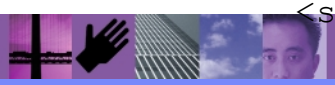
```
Forall ?c such that ?c # Customer
If Exists ?v1 ?v2 ?v3 (And (QuarterMonthlyPurchase(?c ?v1 ?v2 ?v3)
                           func:numeric-greater-than(func:numeric-add(?v1 ?v2 ?v3) 1000)
Then Modify(?c[status -> "Gold"  ])
```

If customer cart contains CD-player and no CD, then advertise CDs on sidebar, based on customer's musical preferences

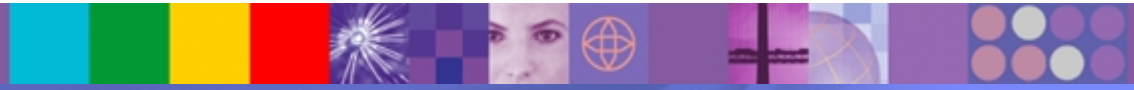
```
Forall ?c such that ?c # Customer
  Forall ?s ?iList such that And(?s # ShoppingCart ?c[shoppingCart -> ?s] ?s[items -> ?iList])
  If And(Exists ?cd-p (And ?cd-p # CD-Player pred:list-contains(?cd-p ?iList))
         Not(Exists ?cd (And ?cd # CD pred:list-contains(?cd ?iList))))
  Then Do((?pref ?c[preferences ->?pref])(?pref-music ?pref[musical -> ?pref-music))
          Execute(act:print(my:prepareAdvertisement(?pref-music))))
```

XML syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<Document xmlns="http://www.w3.org/2007/rif#">
  <payload>
    <Group>
      <id><Const type="http://www.w3.org/2001/XMLSchema#string">purchaseRuleSet</Const></i
      <sentence>
        <Forall>
          <id><Const type="http://www.w3.org/2001/XMLSchema#string">purchaseRule</Const></
          <declare><Var>customer</Var></declare>
          <declare><Var>purchaseQTD</Var></declare>
          <pattern>
            <Member>
              <instance><Var>customer</Var></instance>
              <class><Const type="http://www.w3.org/2007/rif#iri">http://example.com/custo
            </Member>
          </pattern>
          <formula>
            <Implies>
              <if>
                <Exists>
                  <declare><Var>purchaseQTD</Var></declare>
                  <formula>
                    <And>
                      <formula>
                        <Frame>
                          <object><Var>customer</Var></object>
                          <slot ordered="yes">
                            <Const type="http://www.w3.org/2007/rif#iri">purchaserID</Cons
                            <Var>purchaseYTD</Var>
```

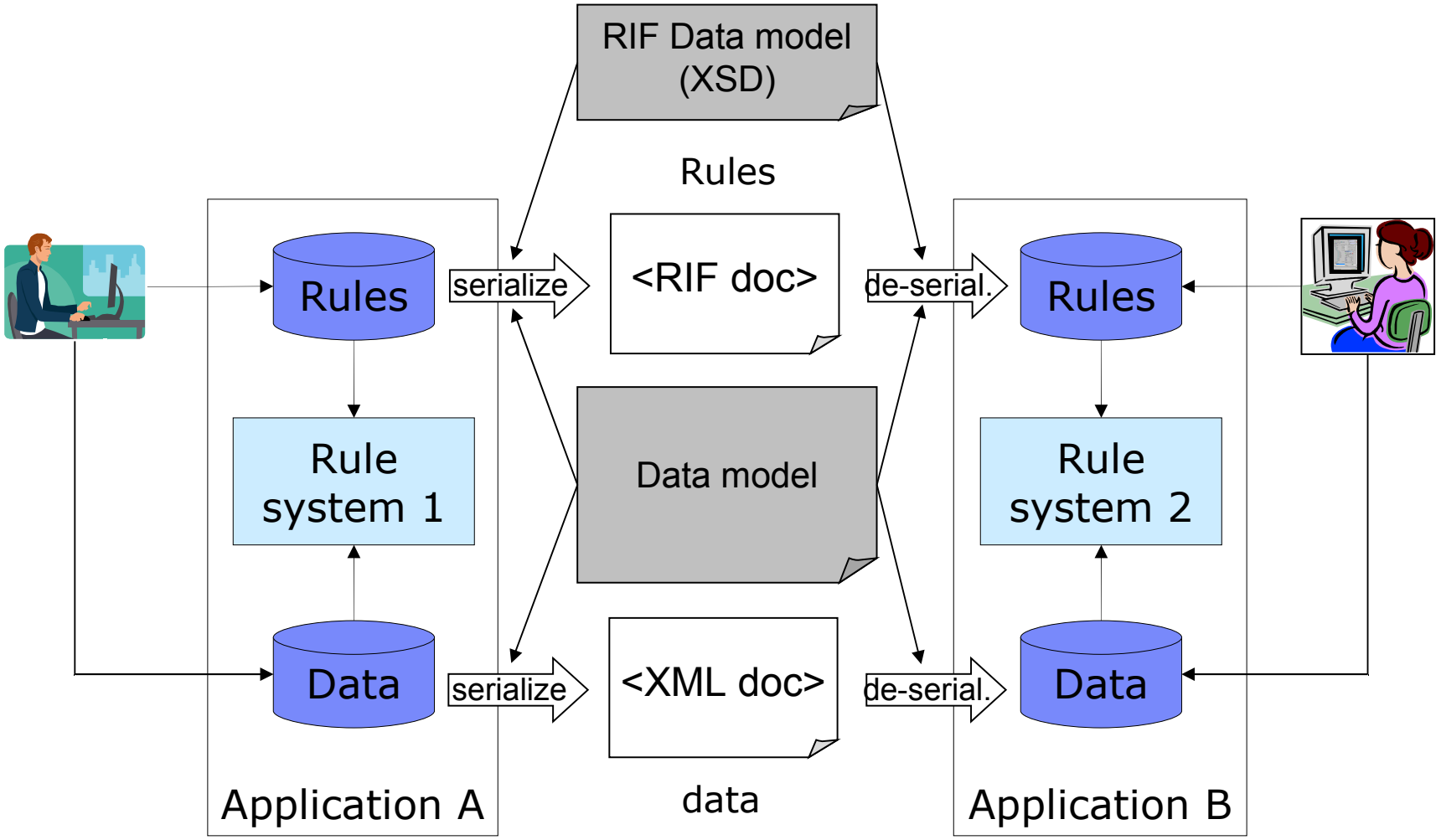


What data for interchanged rules?

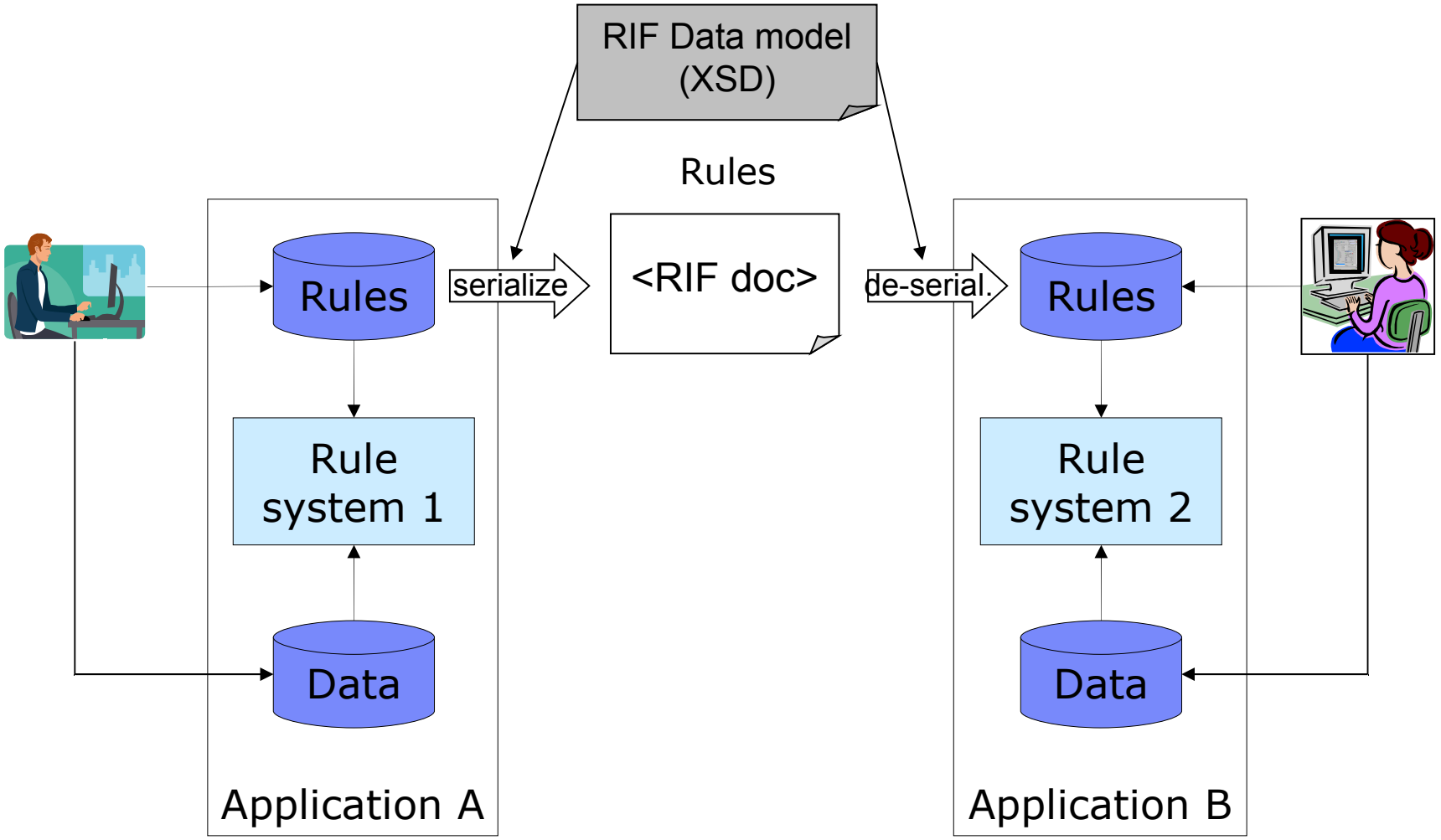


WebSphere software

Where is the data model?



Interchange stand-alone rules



Interchange the rules alone

- The data model is in the rules
  - The rule language is the data modeling language
- Typically: logic programming

The spouse of one's spouse is oneself

Forall ?x ?y ?z,
If ?x [spouse -> ?y] and ?y [spouse -> ?z] then ?x = ?z

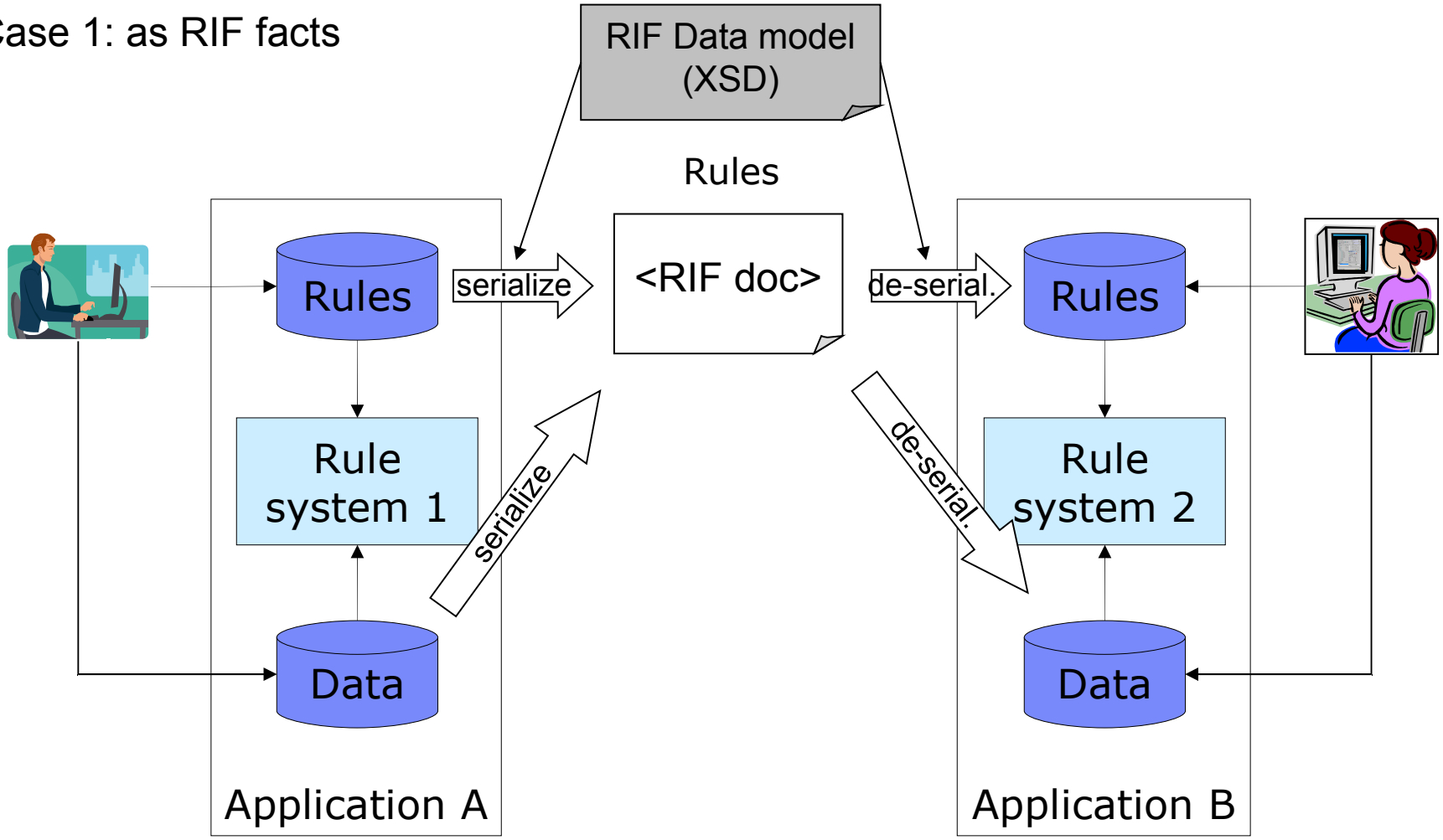
To append something to a list, append it to the tail of the list (and the result of appending any list to an empty list is the appended list)

append([],List,List).
append([H|Tail],X,[H|NewTail]) :- append(Tail,X,NewTail).

⇒ RIF-BLD (Basic Logic Dialect)

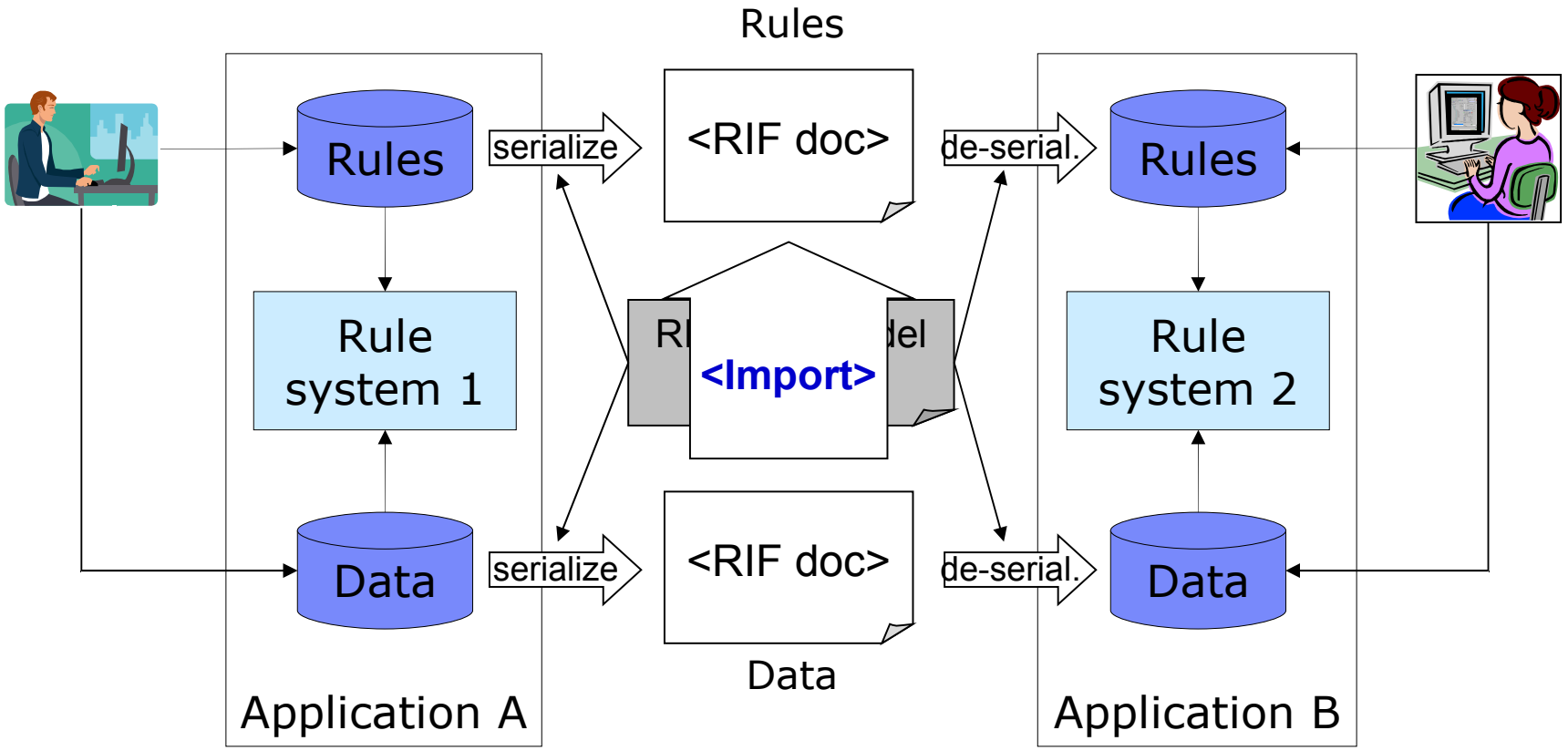
Interchange the data with the rules

Case 1: as RIF facts



Interchange the data with the rules

Case 1: as RIF facts



The Import directive

```
<Import>
        <location> xs:anyURI </location>
</Import>
```

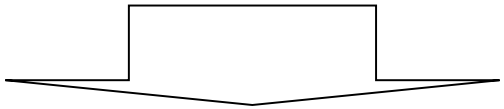
The location of a RIF document

- Import RIF documents (in RIF Core ⇒inherited by all other dialects)
- Semantically equivalent to merging the importing and imported document, with a caveat:
  - `rif:local` constants are local to a document
  - Two `rif:local` constants with the same literal that occur in two different documents are not equal

RIFdoc1.xml

```
<Document xmlns:…>
  <payload>
    <Group>
      <sentence>
        <Atom>
          <op>
            <Const type="&xs;string">
              human
            </Const>
          </op>
          <args ordered="yes">
            <Const type="&rif;local">
              aLocalName
            </Const>
          </args>
        </Atom>
      </sentence>
    </Group>
  </payload>
</Document>
```

```
Document(
  Import(RIFdoc1.)
  Group(
    Forall ?x (
      mortal(?x) :- And( human(?x)
                         ?x = _aLocalName
                       )
                  )
               )
)
```

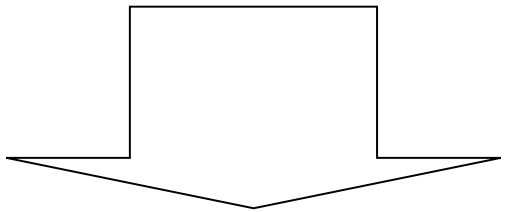


Does not entail mortal(*_aLocalName*)

RIFdoc1.xml

```
<Document xmlns:…>
  <payload>
    <Group>
      <sentence>
        <Atom>
          <op>
            <Const type="&rif;local">
              human
            </Const>
          </op>
          <args ordered="yes">
            <Const type="&xs;string">
              Socrates
            </Const>
          </args>
        </Atom>
      </sentence>
    </Group>
  </payload>
</Document>
```

```
Document(
  Import(RIFdoc1.)
  Group(
    Forall ?x ( mortal(?x) :- _human(?x) )
          )
)
```

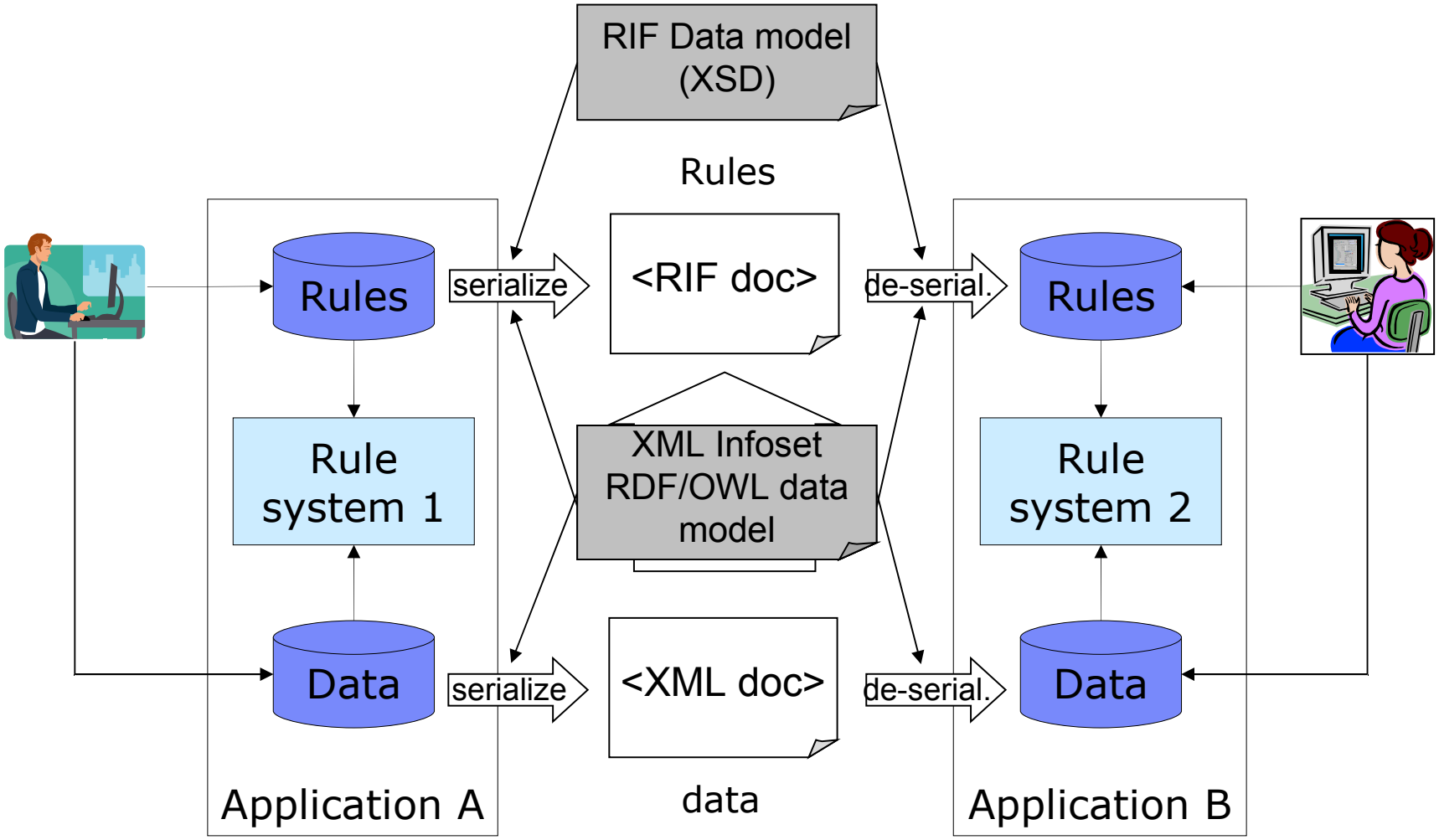


Does not entail
mortal(Socrates)

Interchange the data with the rules

- Special case: as RIF facts
- General case: the data model is in the data
  - XML data
  - RDF graph
  - OWL ontology

Interchange the data with the rules: XML, RDF, OWL data

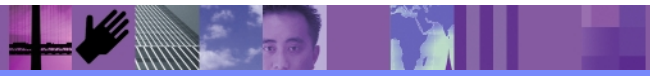


The Import directive

The location of an XML document

```
<Import>
     <location> xs:anyURI </location>
     <profile> rif-import-profile:xml-data </profile>
</Import>
```

- Import any XML document as data (proposed RIF extension)
- The data model is the XML Infoset (as used in XPath 2.0)
- Import profile must be
  - `http://www.w3.org/2007/rif-import-profile#xml-data`



Combining RIF rules with XML data

- Sub-element and attribute names mapped to slot names, using XPath 2.0 expressions as RIF `rif:xpath` constants
  - `o[expr^^rif:xpath->v]` must be true if
    - `o` represents an imported information item (a node in the instance of the data model),
    - *expr* is an XPath 2.0 expression,
    - and `v` is (derived from) the typed value of the sequence that is matched by the XPath 2.0 expression *expr* when `o` is the context node

Forall ?c such ?c [self::Customer^^rif:xpath -> ?c]

Forall ?a such that ?c [Address^^rif:xpath ->?a]

If Or(?c [@xml:lang^^rif:xpath -> "en"]

?a[City^^rif:xpath -> "London"])

Then Do(Assert(speakEnglish(?n)))

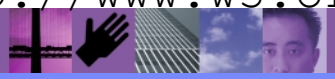
```
<Customer">
    <Name>John</Name>
    <Age>32</Age>
    <Address>
        <Street>Wall street</Street>
        <City>London</City>
    </Adress>
</Customer>
<Customer xml:lang=« en">
    <Name>John</Name>
</Customer>
```

The Import directive

The location of an RDF graph

```
<Import>
    <location> xs:anyURI </location>
    <profile> a RIF-RDF import profile </profile>
</Import>
```

- Import any RDF graph as data (W3C recommandation)
- Four interpretation and entailment profiles
  - http://www.w3.org/ns/entailment/Simple
  - http://www.w3.org/ns/entailment/RDF
  - http://www.w3.org/ns/entailment/RDF-S
  - http://www.w3.org/ns/entailment/D
- One generic profile (no specific notion of interpretation, entailment, model)
  - http://www.w3.org/2007/rif-import-profile#Generic



RIF combination with RDF

- Correspondence between RDF and RIF symbols
  - RDF IRIs are RIF constants in the `rif:iri` symbol space
  - RDF plain literals are RIF constants in the `rdf:plainLiteral` symbol space
  - RDF typed literals are RIF constants with symbol spaces
- Common RIF-RDF interpretation and common model
  - *s'[p'->o']* is true if the triple *s p o* is in the imported RDF graph
  - *a # b* is true iff *a rdf:type b* is in the imported RDF graph
  - *a ## b* is true if *a rdf:subClassOf b* is in the imported RDF graph
  - Condition on data types alignment and lists
  - Simple, RDF, RDFS, D-RDF interpretation iff respective vocabulary included and axioms satisfied

Ex:John ex:brotherOf ex:Jack
Ex:Jack ex:parentOf ex:Mary

Forall ?x ?y ?z (?x [ex:uncleOf -> ?z] :-

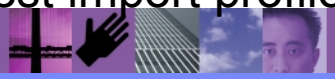
And(?x [ex:brotherOf -> ?y] ?y[ex:parentOf -> ?z]))

The Import directive

The location of an OWL ontology

```
<Import>
    <location> xs:anyURI </location>
    <profile> a RIF-OWL import profile </profile>
</Import>
```

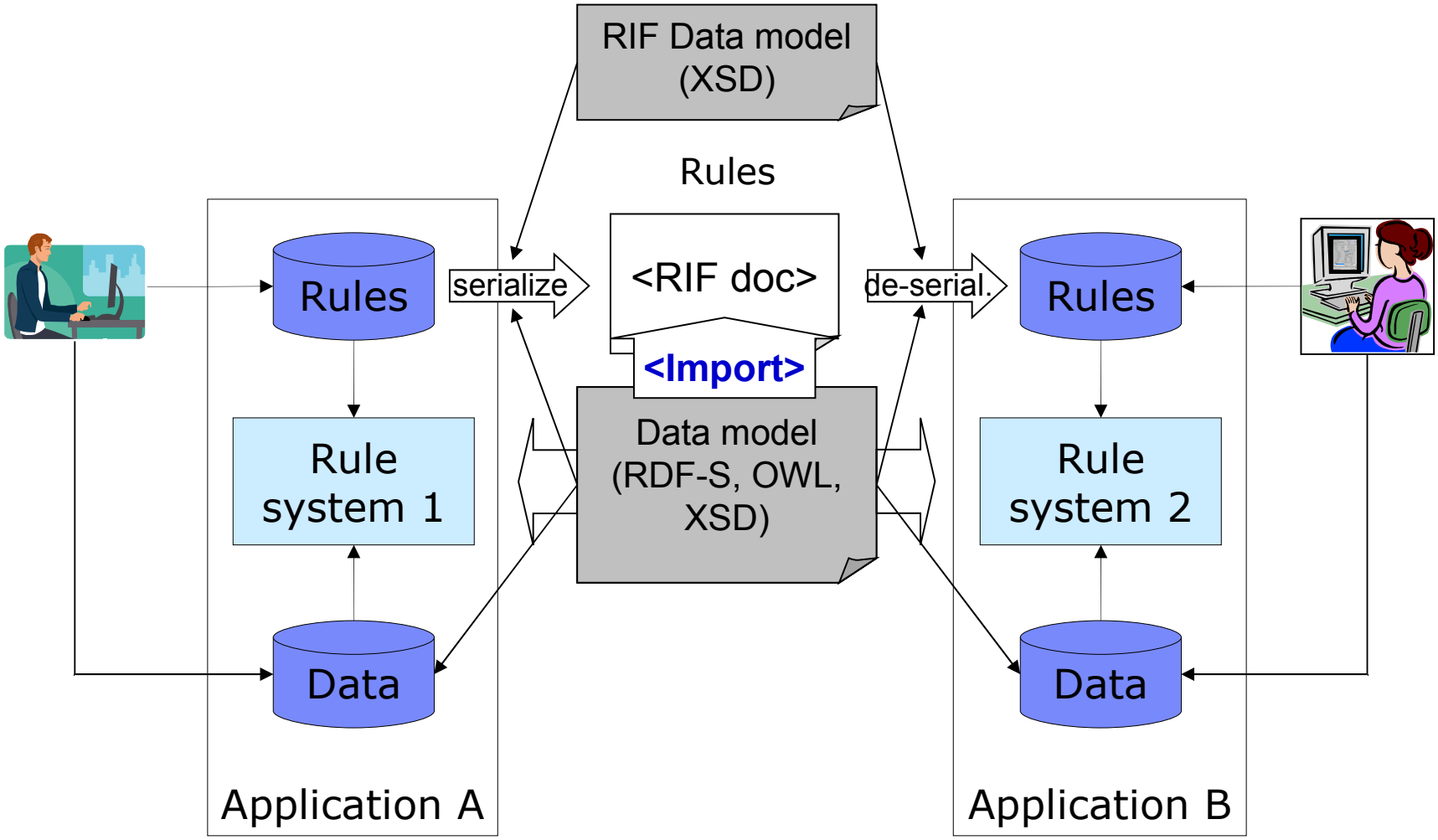
- Import any OWL ontology as data (W3C recommandation)
- Two interpretation and entailment profiles
  - `http://www.w3.org/ns/entailment/OWL-Direct`
  - `http://www.w3.org/ns/entailment/OWL-RDF-Based`
- RDF and OWL combination profiles are ordered
  - `Simple < RDF < RDF-S < D < OWL-RDF-Based`
  - `OWL-Direct < OWL-RDF-Based`
  - Highest import profile in a RIF document applies to all imported graphs



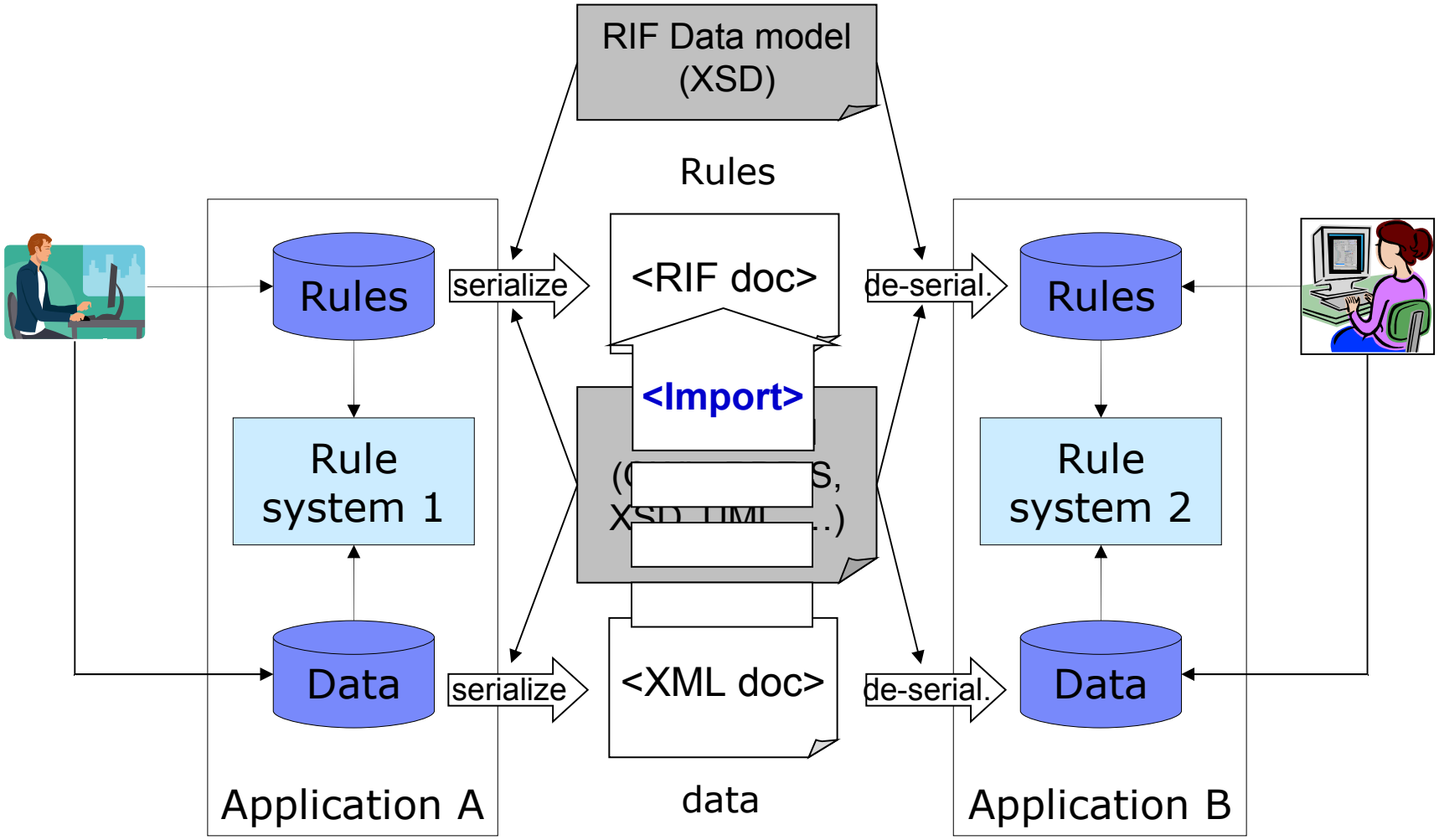
RIF combination with OWL

- OWL 2 Full compatibility is straightforward extension of RDF compatibility
- RIF combination with OWL 2 DL is defined only for RIF DL documents
  - RIF frame *o[p->v]* is a RIF DL frame iff *p* is a constant and *v* is a constant if *p* is *rdf:type*
  - RIF class membership formula *o # c* is a RIF DL formula iff *c* is a constant
  - RIF subclass formula *s ## C* is a RIF DL formula iff *s* and *C* are constants
  - A RIF variable is DL-safe if it does not occur in a DL frame such that *p* belongs to an imported ontology or *p* is *rdf:type* and *v* belongs to an imported ontology
- RIF combination with OWL 2 DL requires semantic extension of RIF frames
  - Frame *o[p->v]* is interpreted as relation *p(o, v)* if *p* is not *rdf:type*, and as *o* belonging to set *v* if *p* is *rdf:type*
- For the purpose of combination with RIF, OWL1 lite and OWL1 DL are seen as syntactic subsets of OWL2 DL

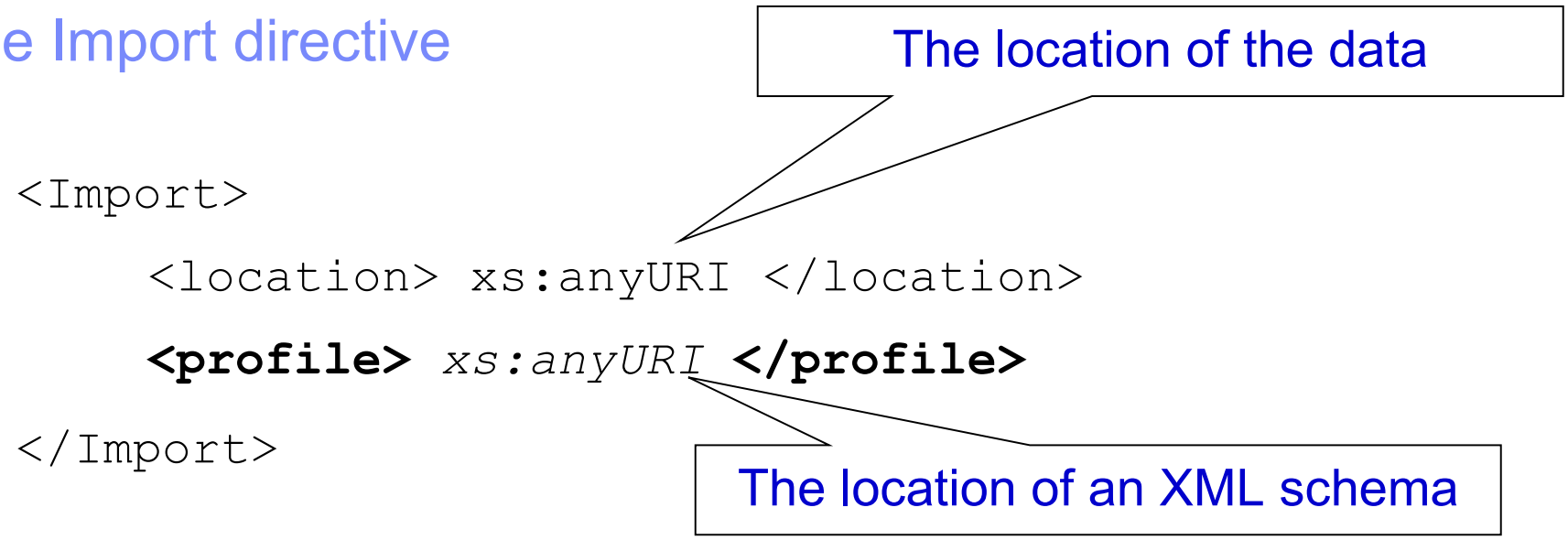
Interchange rules, use own data



Interchange rules, use own data and/or imported data



The Import directive



- Import XML Schema, with or without schema-valid XML document (proposed extension)
- The location of the data can be
  - The URI of an XML document that validate against the schema
  - `http://www.w3.org/2007/rif-import-profile#consumer-side-data`

RIF combination with XML Schema (and consumer-side data)

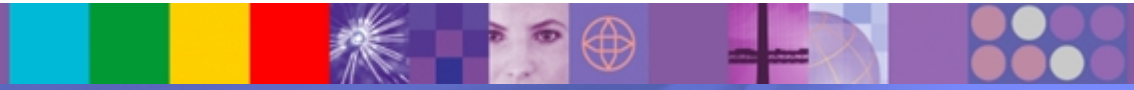
- Element names and schema types mapped to class names, using XML Schema component designator expressions as RIF `rif:xsc-cd` constants
  - If *expr* is an XML Schema component designator expression,
  - And if *expr* designates an imported schema definition that defines a complex type or an element with an anonymous complex type,
  - then `o # expr^^rif:xsc-cd` must be true if and only if
    - `o` represents an information item that is valid against the schema definition designated by *expr*
  - $Expr_1$`^^rif:xsc-cd ##` $expr_2$`^^rif:xsd-cd` must be true if
    - $expr_1$ and $expr_2$ are XML Schema component designator expressions,
    - $expr_1$ and $expr_2$ designate imported schema definitions that defines complex types,
    - and $expr_1$ is derived from $expr_2$

```
<xs:element name="Customer">
   <xs:complexType>
    <xs:sequence>
      <xs:element ref="Name"/>
      <xs:element ref="Age"/>
      <xs:element ref="Address"/>
    </xs:sequence>
   </xs:complexType>
   <xs:attribute ref="xml:lang"/>
 </xs:element>
```

```
Forall ?c such ?c #  /Customer^^rif:xsc-cd
   Forall ?a such that And( ?a # /Customer/Address^^rif:xsc-cd
                            ?c[Address^^rif:xpath ->?a] )
     If Or( ?c [@xml:lang^^rif:xpath -> "en"^^xs:language]
                ?a[City^^rif:xpath -> "London"] )
     Then Do( Assert( speakEnglish(?c) ) )
```

Object-oriented RIF

(using RIF with object-oriented rule languages)



WebSphere software

What object-oriented RIF?

- Object-oriented RIF must provide
  - Syntax to serialize rule constructs common to...
  - Representation of associated object model with features used in...
  - Semantics that preserves properties common to...

  ... most object-oriented rule languages

Requirements (wish-list)

- Syntax to represent, in RIF formulas,
  - Typing of object variables...
  - Access (get/set) to values of object properties...
  - Method calls...
  - Path expressions...

  ...with reference to associated object model
  - Non-existent objects, undefined values
- Syntax to bind object models to RIF documents
  - Discrete class hierarchies
  - Class signatures
    - Name, type and multiplicity of members
    - Signature of methods
  - Encapsulation
- Semantics to impose
  - Well-typing
    - every object's member must be covered by a signature
  - Enforcement of signature declaration
  - That all objects have a primary class
  - Structural inheritance

Already provided in RIF

- Syntax to represent, in RIF formulas,
  - Typing of object variables... → Class membership formulas
  - Access (get/set) to values of object properties... → Frame formulas
  - Method calls... → External functions as slot in frame formulas
  - Path expressions...

  ...with reference to associated object model
  - Non-existent objects, undefined values
- Syntax to bind object models to RIF documents
  - Discrete class hierarchies
  - Class signatures
    - Name, type and multiplicity of members
    - Signature of methods
  - Encapsulation
- Semantics to impose
  - Well-typing
    - every object's member must be covered by a signature
  - Enforcement of signature declaration
  - That all objects have a primary class
  - Structural inheritance → Semantics of class membership and subclass formulas

Class

Provided by XML schema

- Syntax to represent, in RIF formulas,
  - Typing of object variables... → Class membership formulas
  - Access (get/set) to values of object properties... → Frame formulas
  - Method calls... → External functions as slot in frame formulas
  - Path expressions...

  ...with reference to associated object model
  - Non-existent objects, undefined values
- Syntax to bind object models to RIF documents
  - Discrete class hierarchies → Type derivation mechanism
  - Class signatures
    - Name, type and multiplicity of members → Schema component definition
    - Signature of methods
  - Encapsulation
- Semantics to impose
  - Well-typing
    - every object's member must be covered by a signature → Schema validation
  - Enforcement of signature declaration → Schema validation
  - That all objects have a primary class → Schema validation
  - Structural inheritance → Semantics of class membership and subclass formulas

Class

Provided by RIF combination with XML data and schema

- Syntax to represent, in RIF formulas,
  - Typing of object variables... → Class membership formulas
  - Access (get/set) to values of object properties... → Frame formulas
  - Method calls... → External functions as slot in frame formulas
  - Path expressions... → XPath expressions as slot in frame formulas

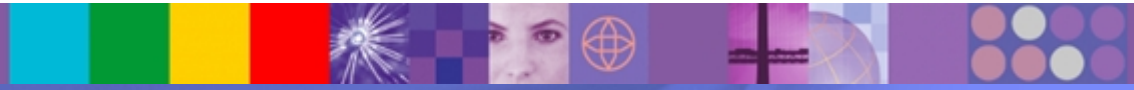
  ...with reference to associated object model → XPath and XSD-CD expressions
  - Non-existent objects, undefined values
- Syntax to bind object models to RIF documents
  - Discrete class hierarchies → Type derivation mechanism → Import and XPath and XSD-CD expressions
  - Class signatures
    - Name, type and multiplicity of members → Schema component definition
    - Signature of methods
  - Encapsulation
- Semantics to impose
  - Well-typing
    - every object's member must be covered by a signature → Semantics of `o#"expr"` and schema validation
  - Enforcement of signature declaration → Semantics of `o#"expr"` and schema validation
  - That all objects have a primary class → Semantics of `o#"expr"` and schema validation
  - Structural inheritance → Semantics of `#, ##, "expr`$_1$`"#"expr`$_2$`"` and Schema validation

How far fro mhere to object-oriented RIF?

- We are almost there
  - Combination with XML data and schema adds almost all the pieces that separate RIF from object-oriented RIF
- Remaining issues
  - Non-existent objects, missing values
    - Shall we extend RIF with a null value: `rif:null`?
  - Encapsulation
    - Private members are used to hide implementation details
    - Not exposed in rules
    - Object-oriented RIF must (only) support interchange of rules grounded in object models
    - Do we need encapsulation?
  - Methods
    - What is missing, really, is a standard way to share the semantics of functions
    - This is orthogonal to rule interchange (and object orientation)

Conclusion

Do we need RIF combinations with more data modelling formats?



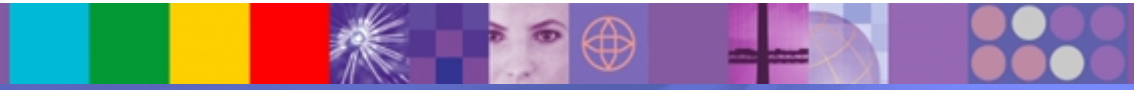
WebSphere software

Conclusion

- Rules is only a part of the knowledge
- Trying to capture all the knowledge as rules is problematic
  - Design, modeling
  - Interchange, sharing, re-use, re-deployment (on different platforms)
  - Maintenance, evolution
- Separating domain knowledge from application specific rules is good
  - Re-combining them is required
  - RIF+RDF/OWL, RIF+XSD
  - What else? SBVR?

Thank you!

Questions?



WebSphere software

Problems

- Mixes (conceptual) domain knowledge with (operational) business rules...
- Mixes domain knowledge with implementation dependent data model...
- Reduces wide ranging policies to application specific operational rules
- ...although these have
  - Different lifecycles
  - Different scope (re-use etc)
  - Different owners
- Increases risk (makes collaboration a quality risk)
  - Design
  - Interchange, sharing, re-use, re-deployment (on different platforms)
  - Maintenance, evolution

A project funded by
SEVENTH FRAMEWORK PROGRAMME