

XML Business Rules Engines

Christian Nentwich
Technical Director
Systemwire Ltd

christian@systemwire.com

Agenda

The What and Why of XML Rules Engines

Types of Engines

This Talk: Focus on Validation of XML

Typical Requirements and Features

Demonstration (time permitting)

What is an XML Rules Engine?

Rules engines have to operate over a data model

- Java rule engines work with Java objects
- XML rule engines work directly with XML data

We need a way to uniquely identify data items and refer to them in rules

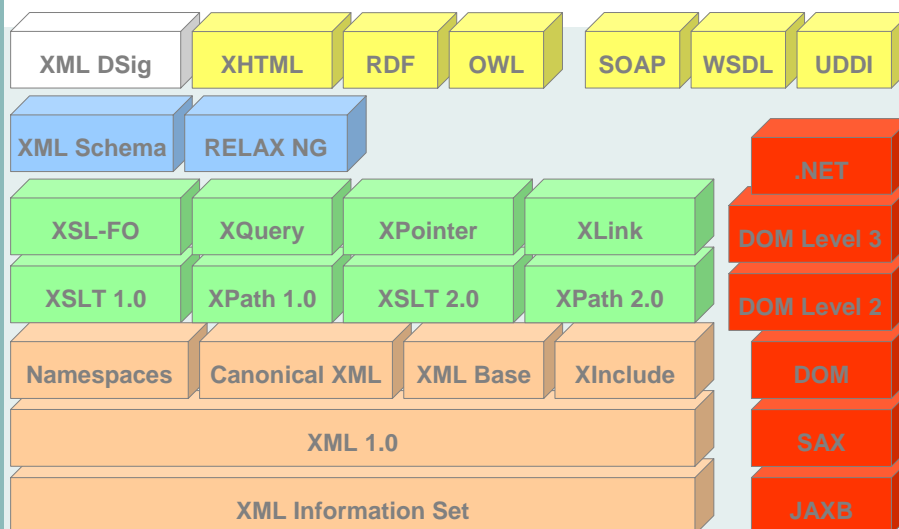
- Java has object references
- XML has XPath, XML Query, XPointer, XLink

XML rule engines process XML documents directly

- XML in, XML out

XML rule engines should fit into an XML architecture

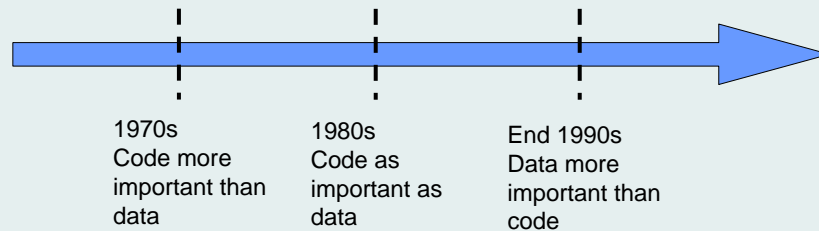
The Comfort of a Strong Family



Why XML Rules Engines?

XML has become a key standard in data-rich architectures

Architectures are now more data-focussed: it makes sense to apply business rules where the core business focus is



Why XML Rules Engines?

A lot of our data is now available in XML

- Can be manipulated in its native format before going into systems
- “Unintended use” opportunities

Performance considerations

Types of Engines

XML like most data can be

- Transformed
- Routed in a workflow
- Triggered according to process definitions

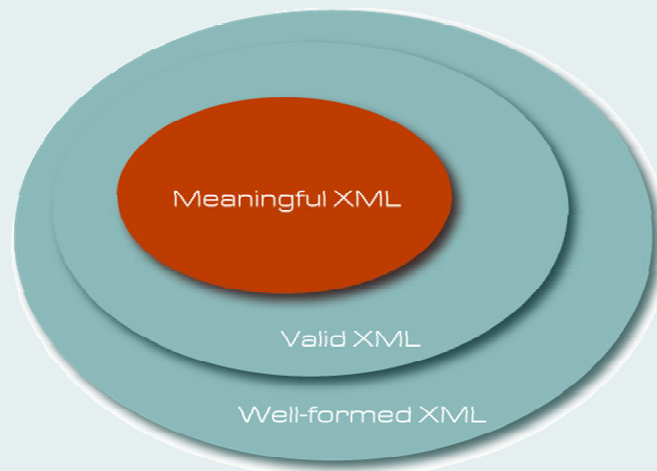
Rules engines can handle all of these scenarios

XML has a strong *validation* requirement

We will look at rule based validation of XML as an example

- This is where our experience lies
- Most of the functional requirements will apply to all types of XML rules engines

XML Validity



XML Validity Examples

Not well-formed:

```
<mydocument>
  <firstDate>2005-11-20</firstDate>
  <lastDate>2005-11-25</firstDate>
</mydocument>
```

Not valid:

```
<mydocument>
  <firstDate>2005-11-20</firstDate>
  <lastDate>2005-11-32</lastDate>
</mydocument>
```

Not meaningful:

```
<mydocument>
  <firstDate>2005-11-25</firstDate>
  <lastDate>2005-11-20</lastDate>
</mydocument>
```

XML Validation Rules

Specify the meaning of complex XML standards

- Support model owners in removing ambiguity
- Support implementers in understanding complex models

Are aimed at improving data quality through operational validation

Can assist implementers of XML systems in getting systems to a higher level of quality by checking for standards compliance

Rule Language Requirements

XML rule engines need a language that can access XML directly

- Need to be able to uniquely identify items in data model
- XPath a common language to use

Expressiveness is very important with complex standards

- Boolean logic can fall short
- If-then-else is not enough for validation

Ideally, should be able to optimise through rewriting

Engine Requirements



Performance!

- XML documents can be huge: **without optimisation it's "game over"!**
- Rule rewriting
- Caching techniques

Ideally should read and write plain XML to fit into architecture

Typically needs access to external data sources for complete processing

- Example: validation of internal rules vs. external validation

Implementation Considerations

Internal use of tree model representation may be necessary for complex rule languages

- However: use of Document Object Model (DOM) blows up documents in memory by factor of 5

“Streaming” processing is possible for some languages, but extremely complex to implement

Internal Validation

Cross-field comparison is a very common requirement

- If fieldA has value A then fieldB has value B
- If fieldA exists then fieldB is equal to it

Additional constraints on top of XML schema also common

- Make optional elements mandatory depending on content
- Further restrict content of elements
- XML schema is quite rigid with respect to workflow state

Complex operations

- Rule languages must offer a scripting/call-out fallback for most real-world applications

External Data Access

XML documents frequently contain data items logically or physically sourced from other data sources

- System identifiers
- Business dates
- Reference items: currencies, country codes, ...

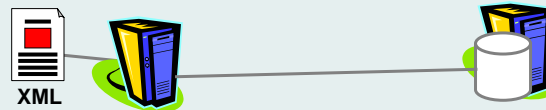
XML documents also come in collections rather than individually

External Data Access: Reference Data

Simple items of data often need to be validated against reference databases

- Such data can be static or dynamic
- Can be low or high volume

Different mechanism required depending on how dynamic/voluminous: caching, real-time retrieval, triggering

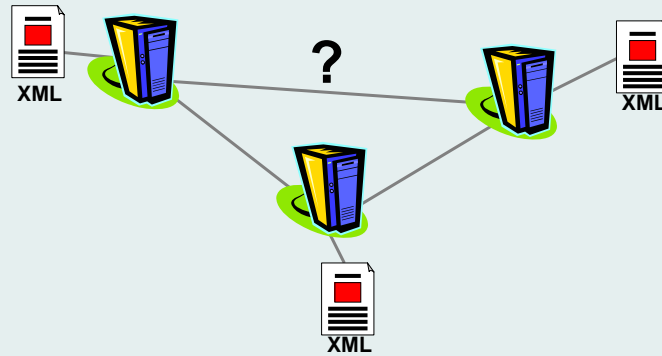


```
<mydocument>
  <trade>
    <currency>GBP</currency>
  </trade>
</mydocument>
```

```
currencies
|iso|name |
|GBP|British Pound |
|EUR|Euro |
|...|... |
```

External Data Access: Distributed Data

Data scattered across documents
Global data consistency required
Tough problem!



©2006 Systemwire Ltd

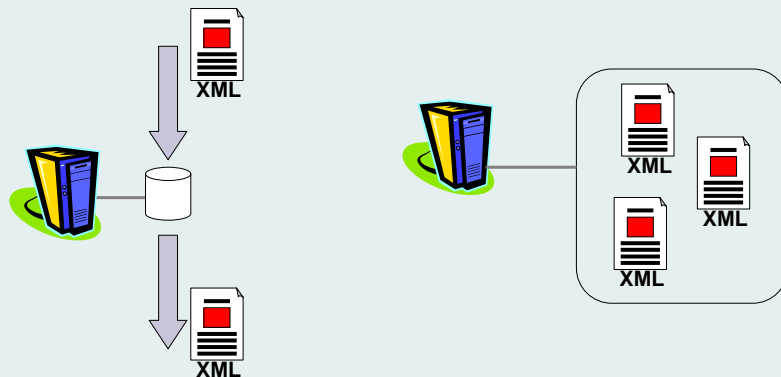
17

SYSTEMWIRE

Architectural Integration

XML documents are typically

- Passed around in real-time flows on queues or
- Processed in bulk



©2006 Systemwire Ltd

18

SYSTEMWIRE

XML Rules Engine Requirements Summary

Should consume and produce XML data natively

- And ideally would also be XML-configured...

Expressive language due to complexity of data models

- Scripting/call-out fall-backs necessary if language is declarative

Ability to access reference data sources in non-XML format

Memory and run-time performance optimisation is a must!

Two Case Study Examples

Standards roll-out

- Business rules specified for complex XML standard by dedicated standards group
- Rolled out to testing and development teams for manual and automated testing

Architectural Resilience

- Financial matching and reconciliation system strengthened with validation
- Reconciliation between two systems cannot proceed until they observe the same business rules: errors detected early

A Demonstration

If we got here in time!

Questions and Comments

<http://www.systemwire.com>

christian@systemwire.com