Message Validation with Semantic Reasoning Tools

Peter Denno

National Institute of Standards and Technology

NIST

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

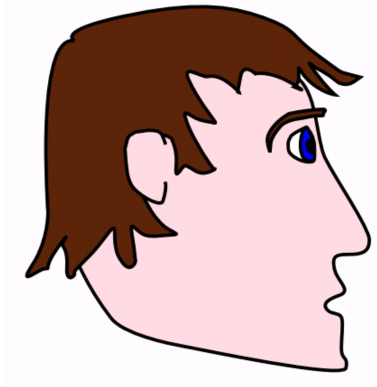
Overview

- Consider the concepts of communication and its validation
- Consider approaches to validation
- Ask where the future might lead

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Part 1 : Communication in Engineered Systems

Q: Do these guys really know what they are talking about ?



Adam

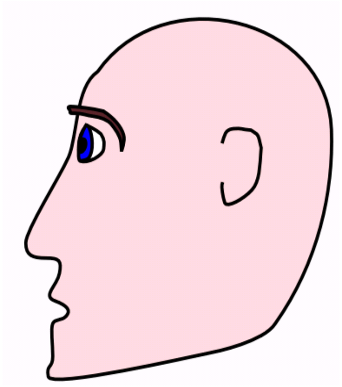
“I want to buy a foobar” →

← “I sell foobars. The base model is $1000”

“I need one with forged widgets” →

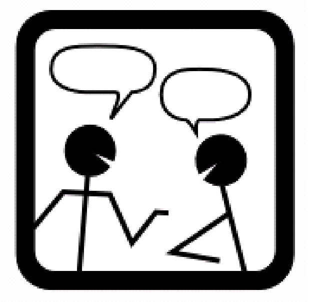
← “FOB, that is $1234”

“I’ll use Carl as my freight forwarder” →



Bart

Carl

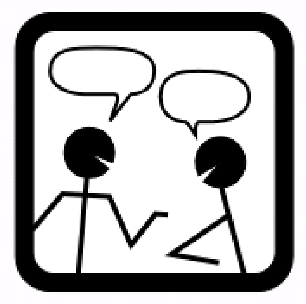


Answer:

If Adam gets his foobar,
and Bart and Carl get paid, yes

(or at least, who then cares?).

What matters?



- Behaviors conform to obligations between parties
- Behaviors conform to regulations and laws
- Behaviors are consistent with physical constraints
- + Behaviors are cost-effective

= Best Practice

A definition:

Validation = assessing whether or not “what matters” is being achieved.

What can be validated?

- Ultimately, we want to know about “what matters”
- But between our needs and their fulfillment (Adam getting his foobar), lies…
  - expressing our needs in shared concepts
  - expressing those concepts in IT engineering detail

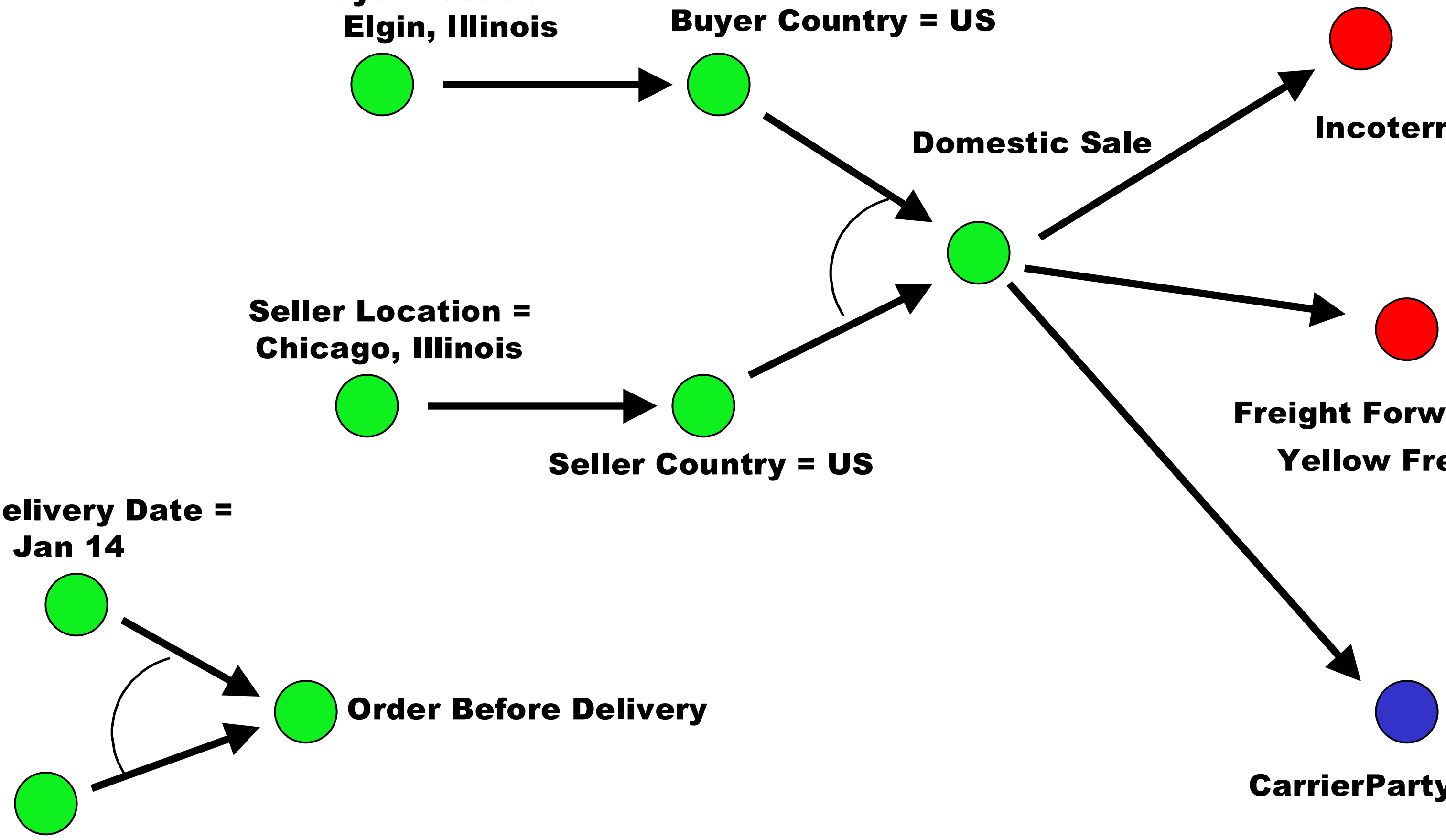
Syntax and Semantics

- Expressing our needs in shared concepts
  - “Message semantics” plays a role
  - Bounded only by the details of the circumstance

- Expressing those concepts in IT engineering detail
  - “Message syntax” plays a role
  - Bounded by encoded rules of “structural well-formedness”

Part 2 : Approaches to Validation

Propagating Constraints



Buyer Location = Elgin, Illinois

Buyer Country = US

Domestic Sale

Incoterm = FOB

Seller Location = Chicago, Illinois

Seller Country = US

Freight Forwarder = Yellow Freight

Req Delivery Date = Jan 14

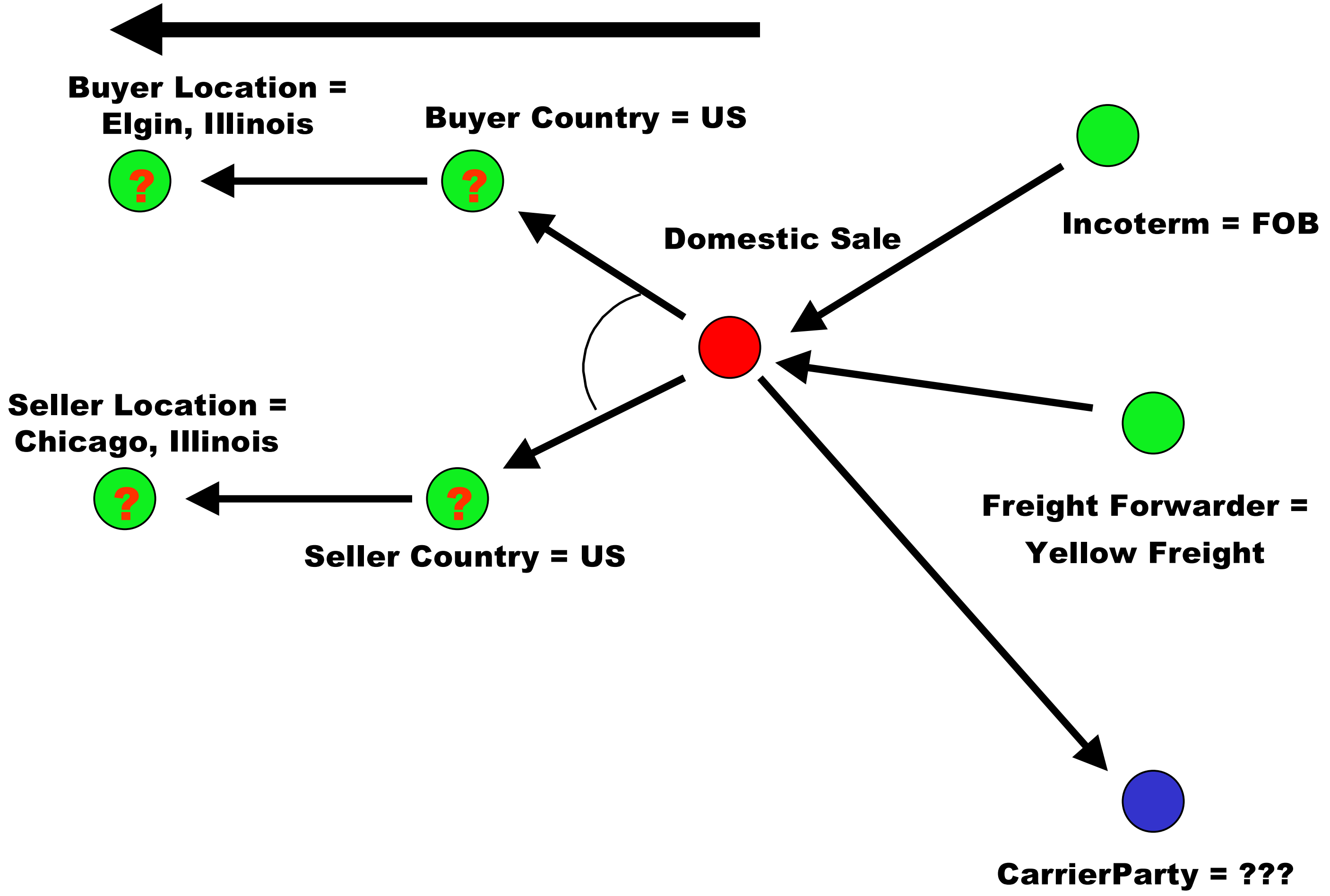
Order Before Delivery

CarrierParty = ???

Order Issue Date = Jan 2

Think like a prosecutor, not a programmer.

Run it ‘backward’



Buyer Location = Elgin, Illinois

Buyer Country = US

Domestic Sale

Incoterm = FOB

Seller Location = Chicago, Illinois

Seller Country = US

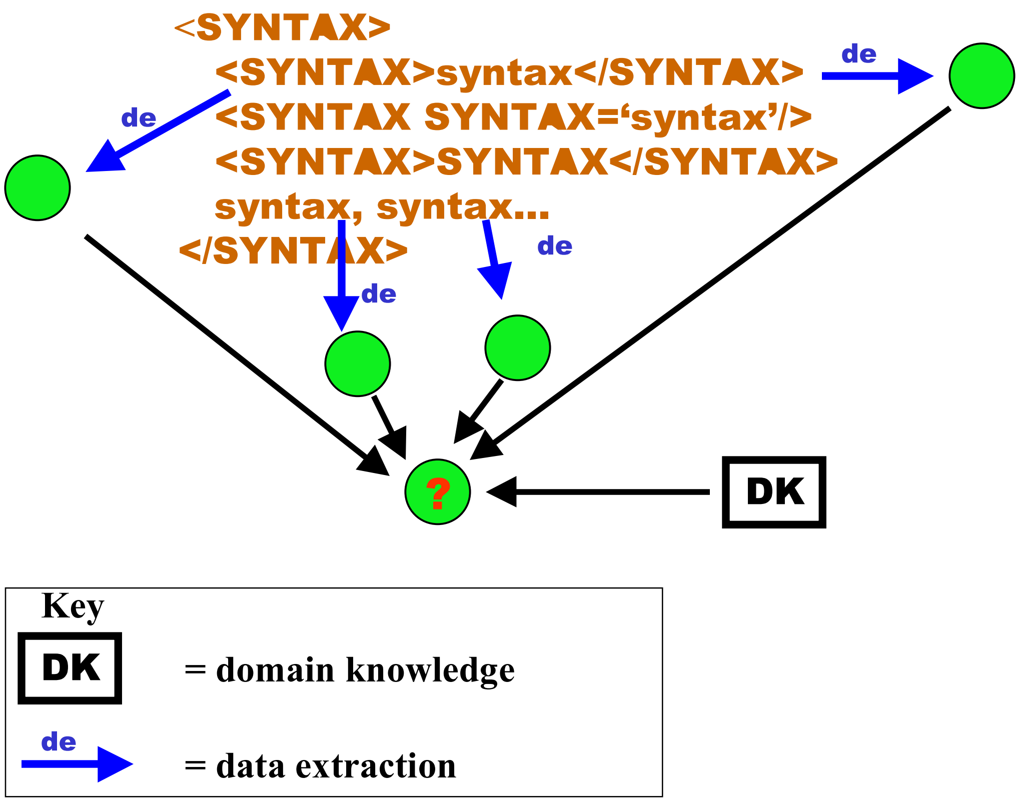
Freight Forwarder = Yellow Freight

CarrierParty = ???

Think like a prosecutor, not a programmer.

One way to accomplish this…

Schematron, et al.



```
<SYNTAX>
  <SYNTAX>syntax</SYNTAX>
  <SYNTAX SYNTAX='syntax'/>
  <SYNTAX>SYNTAX</SYNTAX>
  syntax, syntax…
</SYNTAX>
```

Key

DK = domain knowledge

de → = data extraction

Pro:

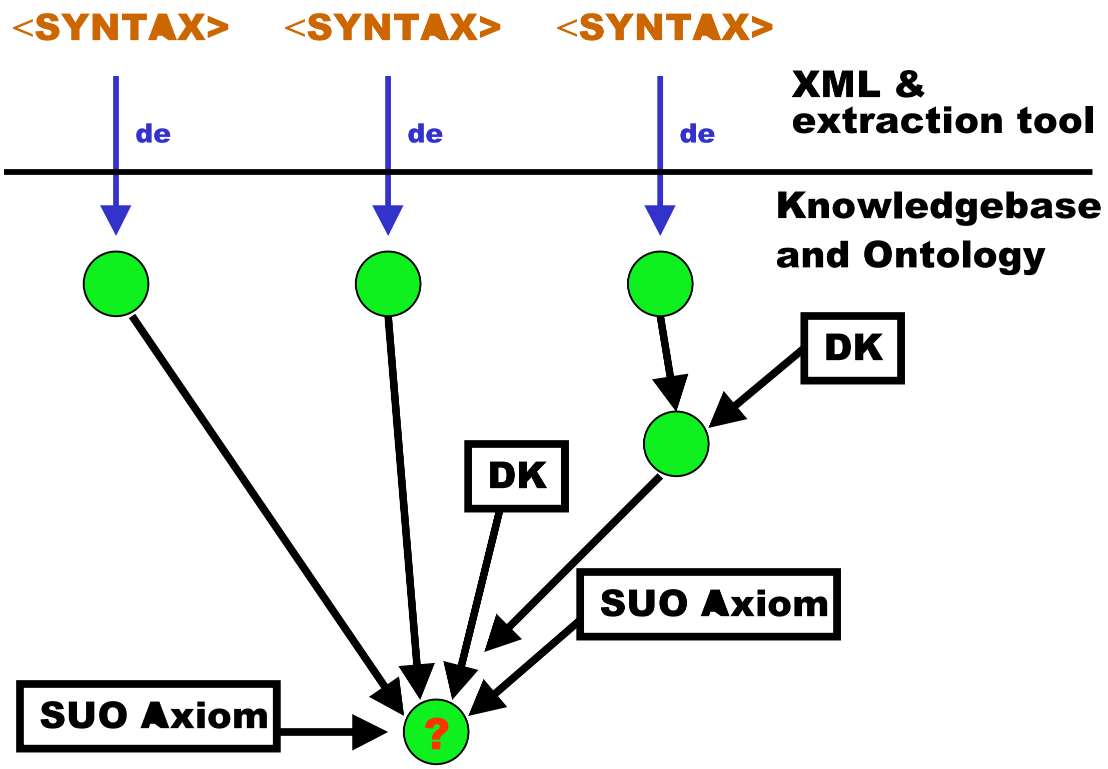
- Unit testing
- Elementary, proven tools

Con:

- New syntax? Throw it away.
- Repeats syntax pattern to set up context
- Domain Knowledge (DK) embedded in concerns about syntax
- Single message validation

...another way

Separate Data Extraction from Domain Knowledge



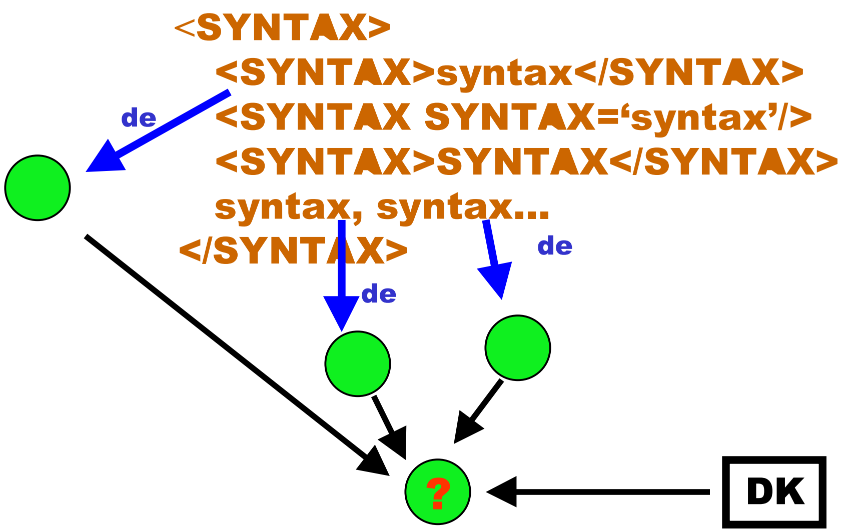
Pro:

- Uncoupled from syntax
- “Unanticipated validation”
- Multi-message validation

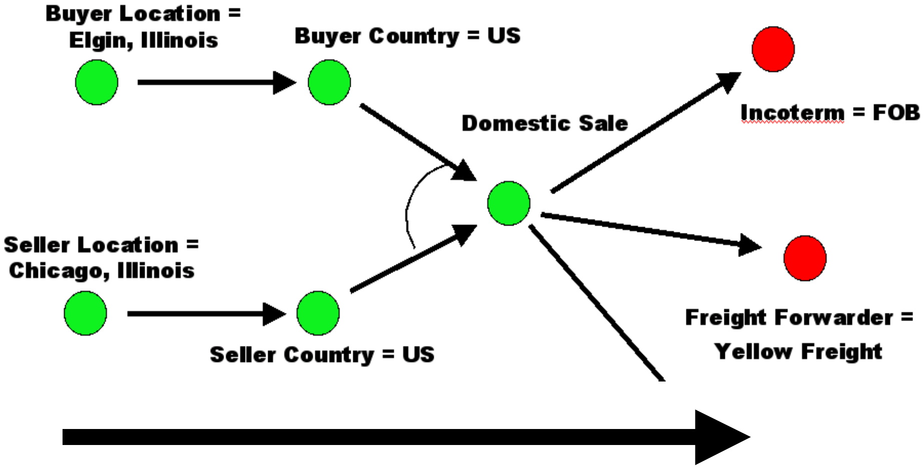
Con:

- No unit testing
- High-tech, unproven tools
- It requires an “ontologist”
- Proof --> meaningful report

Syntax-centric approach forces the network to run in one direction

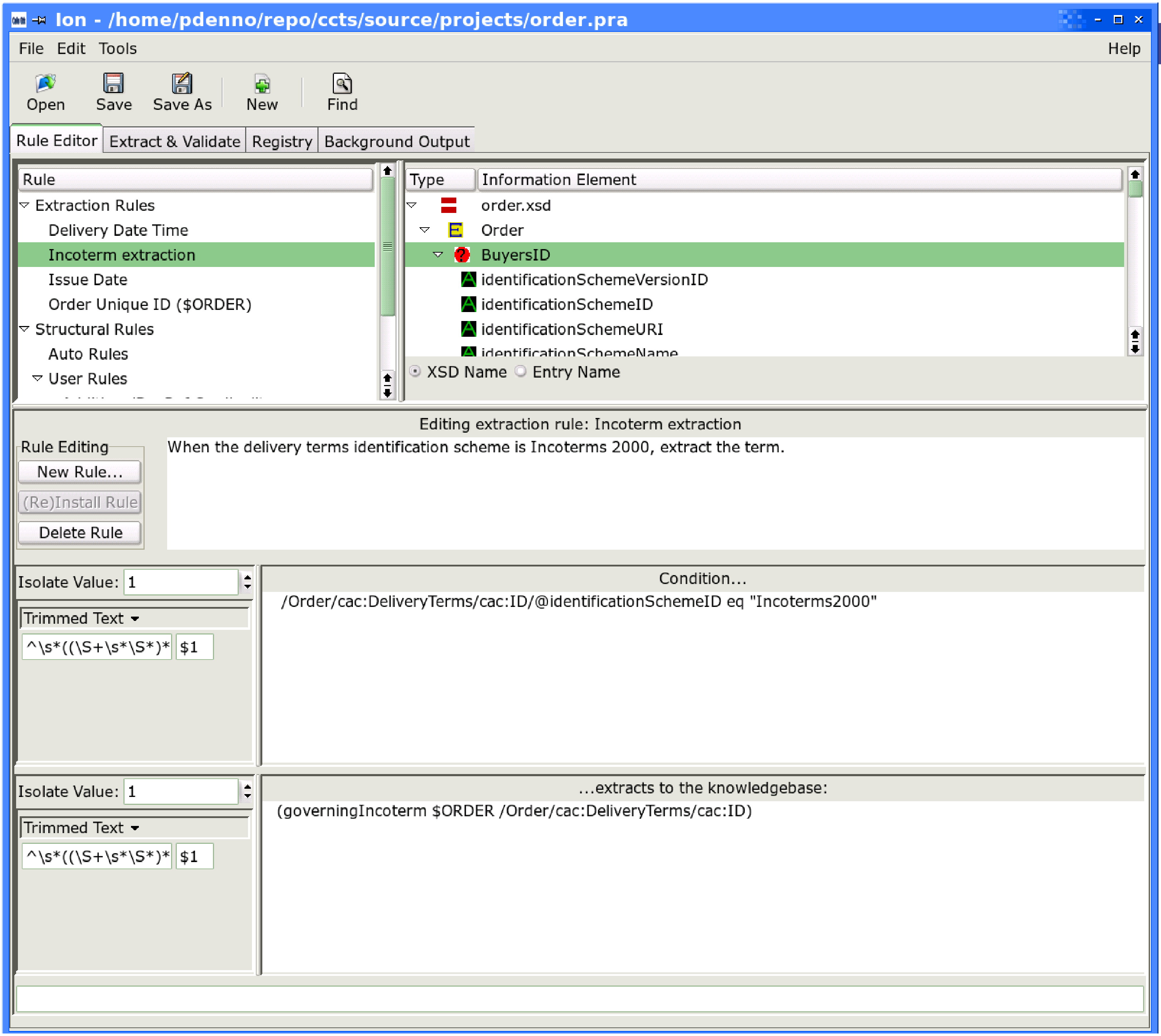


```
<SYNTAX>
  <SYNTAX>syntax</SYNTAX>
  <SYNTAX SYNTAX='syntax'/>
  <SYNTAX>SYNTAX</SYNTAX>
  syntax, syntax...
</SYNTAX>
```



Buyer Location = Elgin, Illinois
Buyer Country = US
Domestic Sale
Incoterm = FOB
Seller Location = Chicago, Illinois
Seller Country = US
Freight Forwarder = Yellow Freight

* More "procedural" means easier reporting of validation exceptions, but less circumspect, more repetitive



Ion - /home/pdenno/repo/ccts/source/projects/order.pra

Rule
Extraction Rules
Delivery Date Time
Incoterm extraction
Issue Date
Order Unique ID ($ORDER)
Structural Rules
Auto Rules
User Rules

Type	Information Element
🟥	order.xsd
E	Order
?	BuyersID
A	identificationSchemeVersionID
A	identificationSchemeID
A	identificationSchemeURI
A	identificationSchemeName

● XSD Name ○ Entry Name

Editing extraction rule: Incoterm extraction

When the delivery terms identification scheme is Incoterms 2000, extract the term.

Isolate Value: 1

Trimmed Text

^\s*((\S+\s*\S*)* $1

Condition...

```
/Order/cac:DeliveryTerms/cac:ID/@identificationSchemeID eq "Incoterms2000"
```

Isolate Value: 1

Trimmed Text

^\s*((\S+\s*\S*)* $1

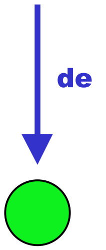
...extracts to the knowledgebase:

```
(governingIncoterm $ORDER /Order/cac:DeliveryTerms/cac:ID)
```

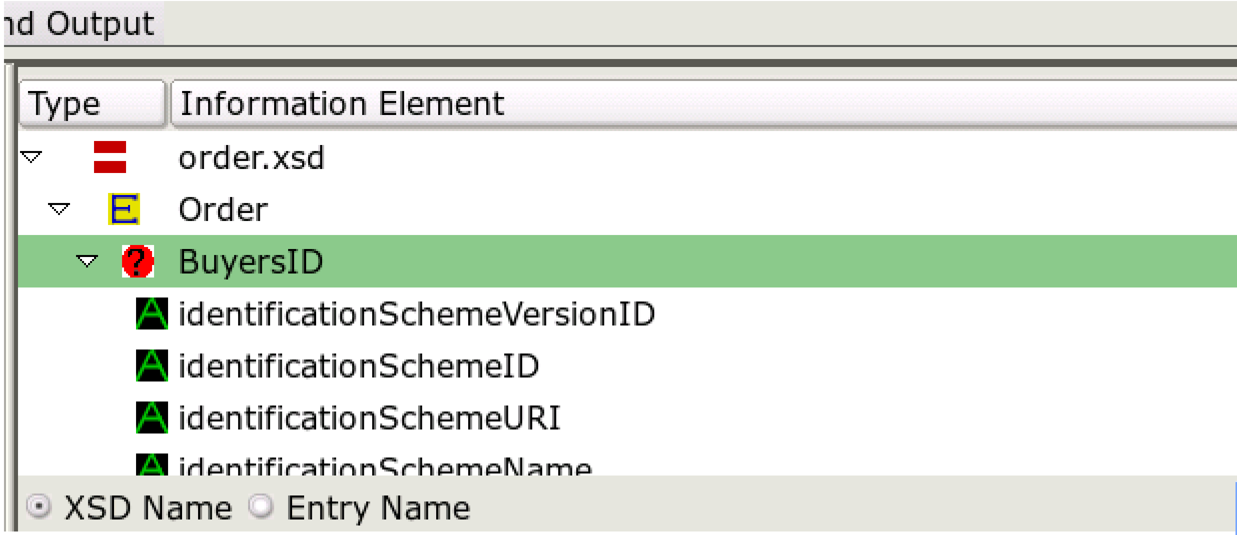
The Validation Tool

data extraction rules

<SYNTAX>



The Validation Tool



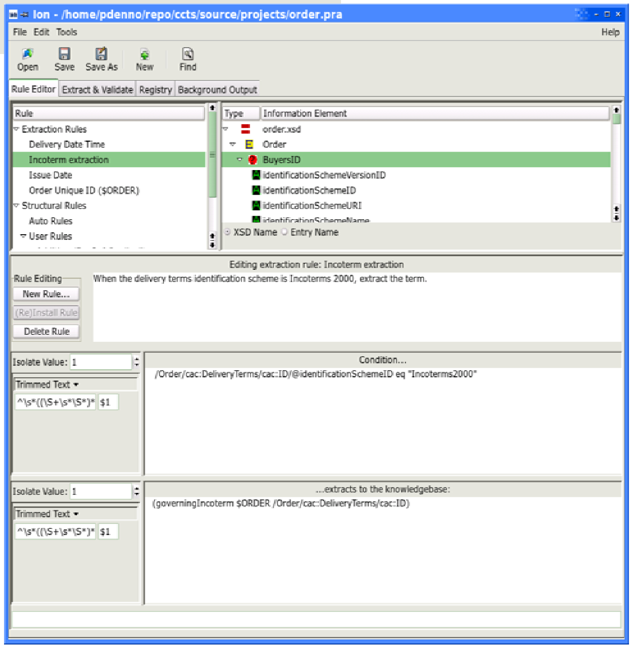
Type	Information Element
▽ 🟥	order.xsd
▽ E	Order
▽ ?	BuyersID
A	identificationSchemeVersionID
A	identificationSchemeID
A	identificationSchemeURI
A	identificationSchemeName

◉ XSD Name ○ Entry Name

data extraction rules
<SYNTAX>
de

The tree depicted is the XSD, arranged as an instance document might appear, if all the elements that might occur did occur.

You can address these by Xpath.
You can get the annotation provided in the UBL XSD.



Rule
▽ Extraction Rules
Delivery Date Time
Incoterm extraction
Issue Date
Order Unique ID ($ORDER)
▽ Structural Rules
Auto Rules
▽ User Rules

Type	Information Element
▽ 🟥	order.xsd
▽ E	Order
▽ ?	BuyersID
A	identificationSchemeVersionID
A	identificationSchemeID
A	identificationSchemeURI
A	identificationSchemeName

◉ XSD Name ○ Entry Name

Editing extraction rule: Incoterm extraction

When the delivery terms identification scheme is Incoterms 2000, extract the term.

Isolate Value: 1

Trimmed Text ▾

`^\s*((\S+\s*\S*)*` `$1`

Condition...

```
/Order/cac:DeliveryTerms/cac:ID/@identificationSchemeID eq "Incoterms2000"
```

Isolate Value: 1

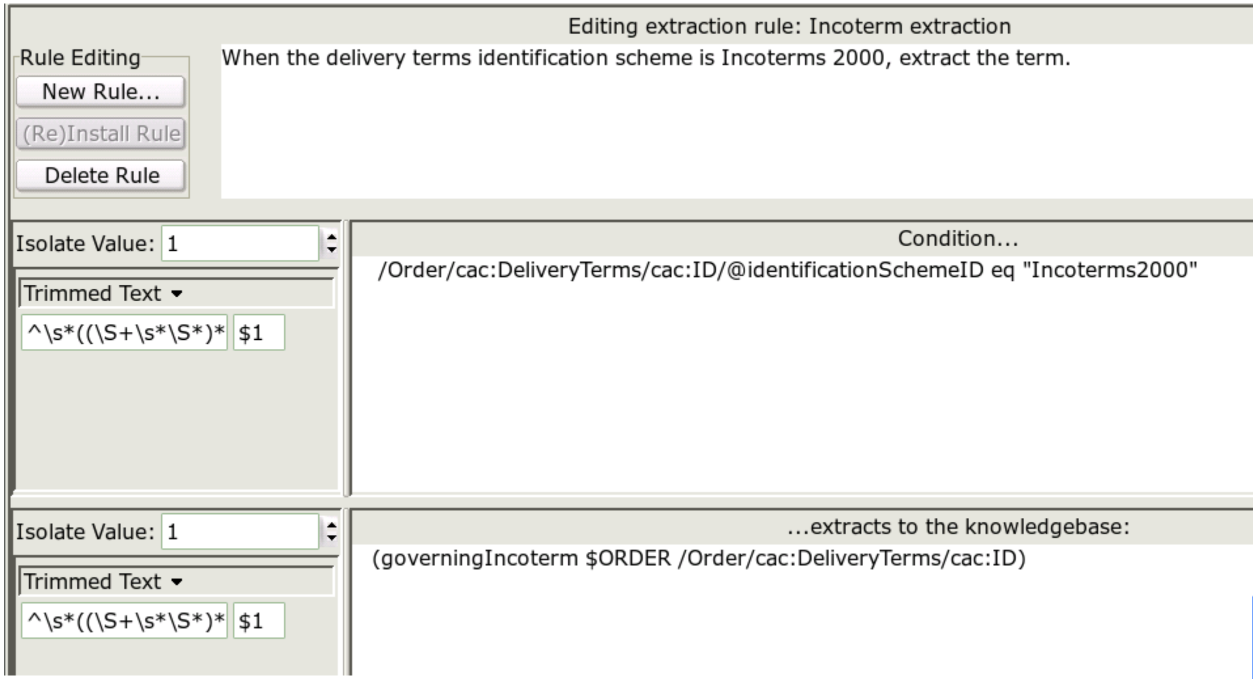
Trimmed Text ▾

`^\s*((\S+\s*\S*)*` `$1`

...extracts to the knowledgebase:

```
(governingIncoterm $ORDER /Order/cac:DeliveryTerms/cac:ID)
```

The Validation Tool



Editing extraction rule: Incoterm extraction

Rule Editing

- New Rule...
- (Re)Install Rule
- Delete Rule

When the delivery terms identification scheme is Incoterms 2000, extract the term.

Isolate Value: 1

Trimmed Text ▾

^\s*((\S+\s*\S*)* $1

Condition...

```
/Order/cac:DeliveryTerms/cac:ID/@identificationSchemeID eq "Incoterms2000"
```

Isolate Value: 1

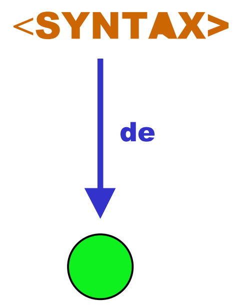
Trimmed Text ▾

^\s*((\S+\s*\S*)* $1

...extracts to the knowledgebase:

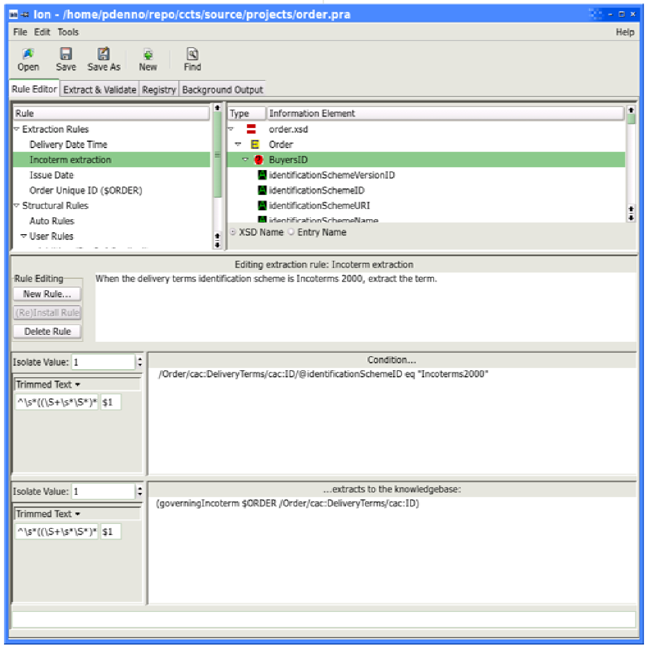
```
(governingIncoterm $ORDER /Order/cac:DeliveryTerms/cac:ID)
```

data extraction rules



One Rule:

- **Top : Documentation**
- **Middle : Condition is XPath (in this case, id scheme = “Incoterms”)**
- **Bottom : what is extracted to the knowledgebase (uses XPath too)**



Rule

- Extraction Rules
  - Delivery Date Time
  - Incoterm extraction
  - Issue Date
  - Order Unique ID ($ORDER)
- Structural Rules
  - Auto Rules
  - User Rules

Type	Information Element
	order.xsd
E	Order
	BuyersID
	identificationSchemeVersionID
	identificationSchemeID
	identificationSchemeURI
	identificationSchemeName

○ XSD Name ○ Entry Name

Editing extraction rule: Incoterm extraction

Rule Editing

- New Rule...
- (Re)Install Rule
- Delete Rule

When the delivery terms identification scheme is Incoterms 2000, extract the term.

Isolate Value: 1

Trimmed Text ▾

^\s*((\S+\s*\S*)* $1

Condition...

/Order/cac:DeliveryTerms/cac:ID/@identificationSchemeID eq "Incoterms2000"

Isolate Value: 1

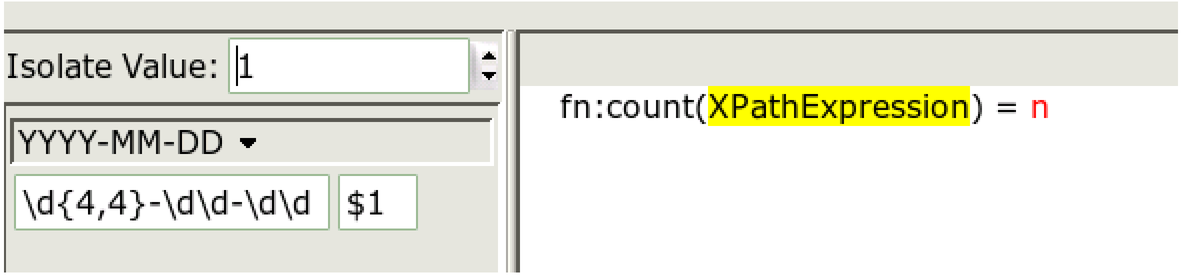
Trimmed Text ▾

^\s*((\S+\s*\S*)* $1

...extracts to the knowledgebase:

(governingIncoterm $ORDER /Order/cac:DeliveryTerms/cac:ID)

The Validation Tool



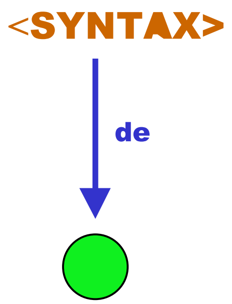
Isolate Value: 1

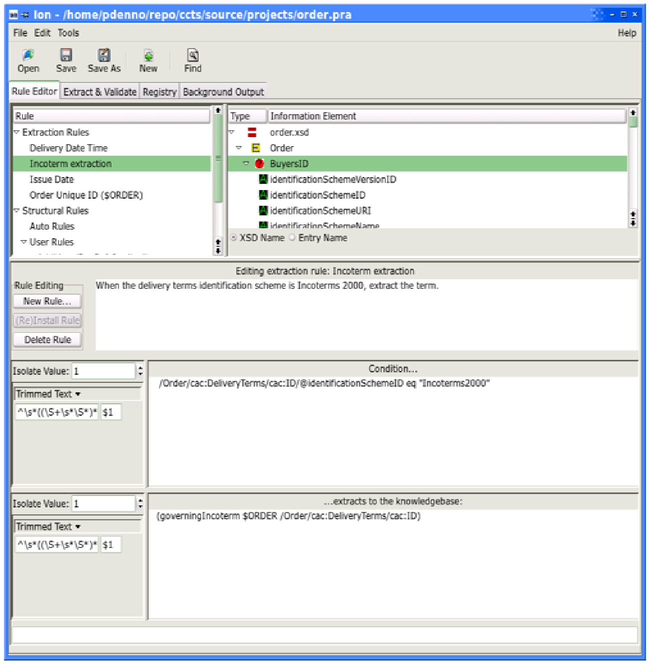
YYYY-MM-DD

\d{4,4}-\d\d-\d\d $1

fn:count(XPathExpression) = n

data extraction rules





File Edit Tools Help
Open Save Save As New Find
Rule Editor Extract & Validate Registry Background Output
Rule Editing
New Rule...
(Re)Install Rule
Delete Rule

Ion - /home/pdenno/repo/ccts/source/projects/order.pra

Rule
Extraction Rules
Delivery Date Time
Incoterm extraction
Issue Date
Order Unique ID ($ORDER)
Structural Rules
Auto Rules
User Rules

Type	Information Element
	order.xsd
E	Order
	BuyersID
A	identificationSchemeVersionID
A	identificationSchemeID
A	identificationSchemeURI
A	identificationSchemeName

● XSD Name ○ Entry Name

Editing extraction rule: Incoterm extraction

When the delivery terms identification scheme is Incoterms 2000, extract the term.

Isolate Value: 1

Trimmed Text

^\s*((\S+\s*\S*)* $1

Condition...

/Order/cac:DeliveryTerms/cac:ID/@identificationSchemeID eq "Incoterms2000"

Isolate Value: 1

Trimmed Text

^\s*((\S+\s*\S*)* $1

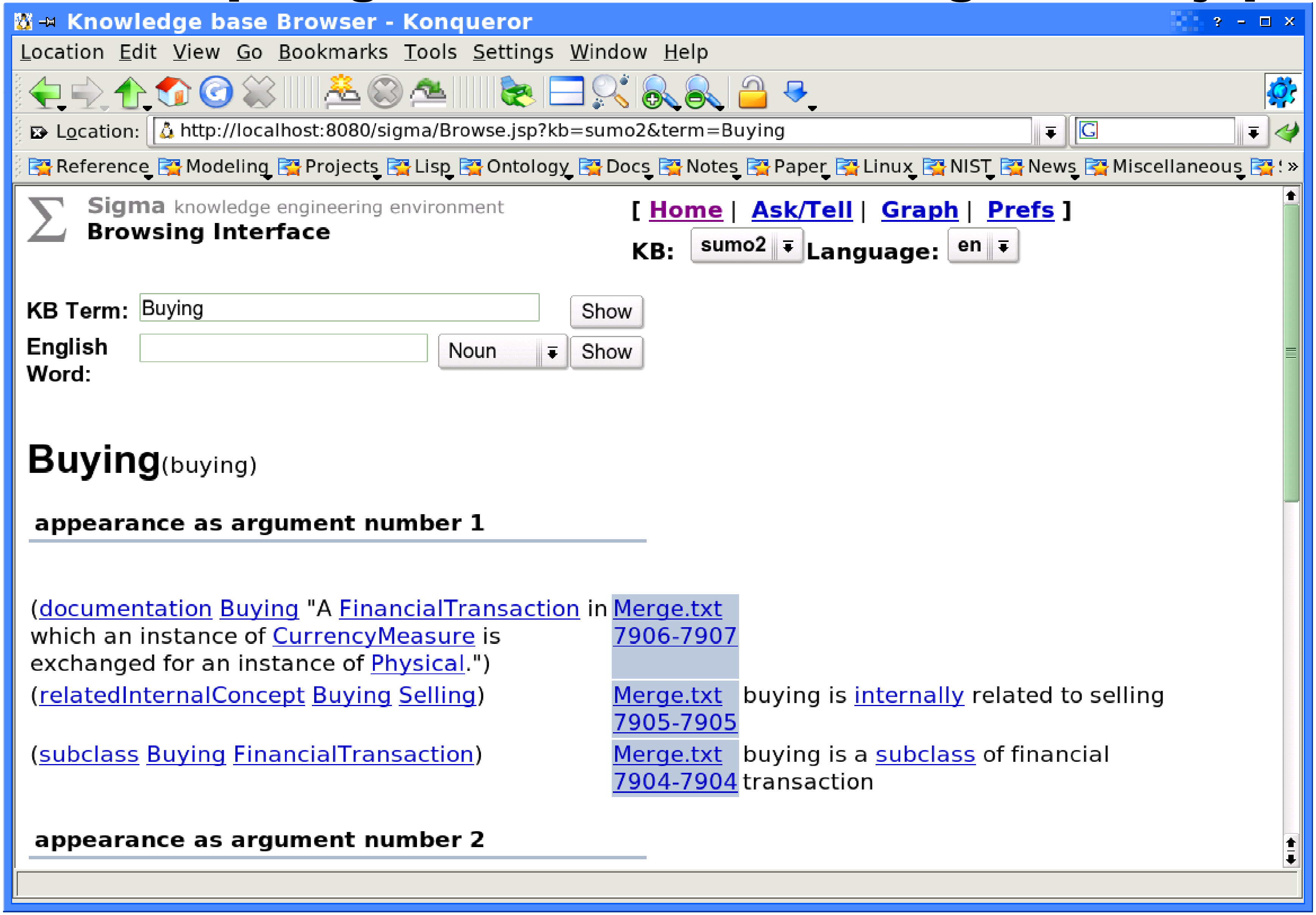
...extracts to the knowledgebase:

(governingIncoterm $ORDER /Order/cac:DeliveryTerms/cac:ID)

Extraction part of rule :

- **Left : How extracted data is to be interpreted.**
- **Right : A pattern before it is edited.**

Sigma KEE and Vampire used in development http://sigma2.cim3.net:8080/sigma/KBs.jsp


Knowledge base Browser - Konqueror
Location Edit View Go Bookmarks Tools Settings Window Help
Reference Modeling Projects Lisp Ontology Docs Notes Paper Linux NIST News Miscellaneous
[Home | Ask/Tell | Graph | Prefs]
Show
Show

Location: http://localhost:8080/sigma/Browse.jsp?kb=sumo2&term=Buying

Σ **Sigma** knowledge engineering environment
Browsing Interface

KB: sumo2 **Language:** en

KB Term: Buying

English Word: Noun

Buying(buying)

appearance as argument number 1

(documentation Buying "A FinancialTransaction in which an instance of CurrencyMeasure is exchanged for an instance of Physical.")	Merge.txt 7906-7907	
(relatedInternalConcept Buying Selling)	Merge.txt 7905-7905	buying is internally related to selling
(subclass Buying FinancialTransaction)	Merge.txt 7904-7904	buying is a subclass of financial transaction

appearance as argument number 2

Part 3 :
The Future of
Message Validation ?

Future tool development effort

- Message Assembly
- Incoterms 2000
- UN Rec 16 LOCODES
- BIEs to aid data extraction
- Multi-message
- OAGIS

INCOTERMS FOR AMERICANS®

(Fully Revised For Incoterms 2000)

Simplifies and answers questions about Incoterms® 2000 for U.S. foreign traders.

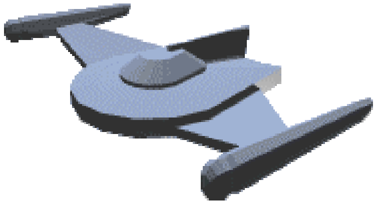
by Frank Reynolds

from INTERNATIONAL PROJECTS, INC.

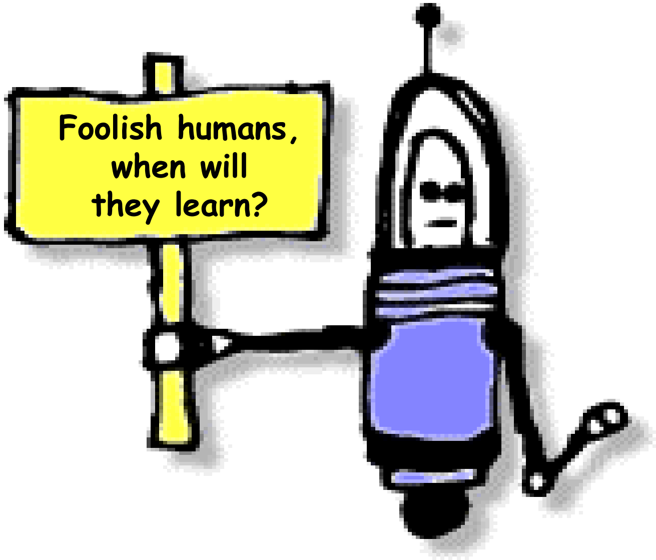
Copyright 1999 by
International Projects, Inc.
Toledo, OH USA
ALL RIGHTS RESERVED
ISBN: 1-886457-06-9

The future, more generally

- Multi-message validation
- Normative codes (Incoterms, LOCODES, etc) integrated with an upper ontology



- Registry content integrated with an upper ontology
- Search engine registry lookup



Conclusions

- **Validation requires reasoning across a broad range of concerns.**
  - **“What matters,” not just XML Schema !**

- **Semantic reasoning tools, upper ontology, knowledge from normative codes, address that range of concerns.**

- **More info on the UBL tool: peter.denno@nist.gov**

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Backup Slides

About Context:

- Registry Viewpoint:

  "A context names a
  category of usage
  of a term."

- Ontology Viewpoint:
  "'Context' is the word we use to refer to the constraints on interpretation before we know them."

About Distinctions:

- Registry viewpoint:

  “Partition vocabularies where distinctions seem likely.”

- Ontology viewpoint:

  “Show me the distinction, and I’ll encode it.”

More definitions:

- System =a collection of parts that work jointly toward some goal.
- Systems integration = enabling the communication that allows the parts to work together
- Semantic integration = that part of system integration that concerns whether or not messages, correctly received and disassembled, in fact serve to direct the recipient to perform the desired behavior.