



Defense Information Systems Agency
Joint Interoperability Test Command

Ontologies and Information Exchange: A Pragmatic Framework

Team NGIT

Bernard P. Zeigler, Ph. D

Arizona Center for Integrative Modeling and Simulation
(ACIMS)

Doohwan Kim, Ph. D.

RTSync.com



Overview

- Basic Concepts to be discussed
- Information Exchange Framework
- The need for ontology and pragmatics
- Importance of rules and constraints
- Strengths/Limitations of current environments

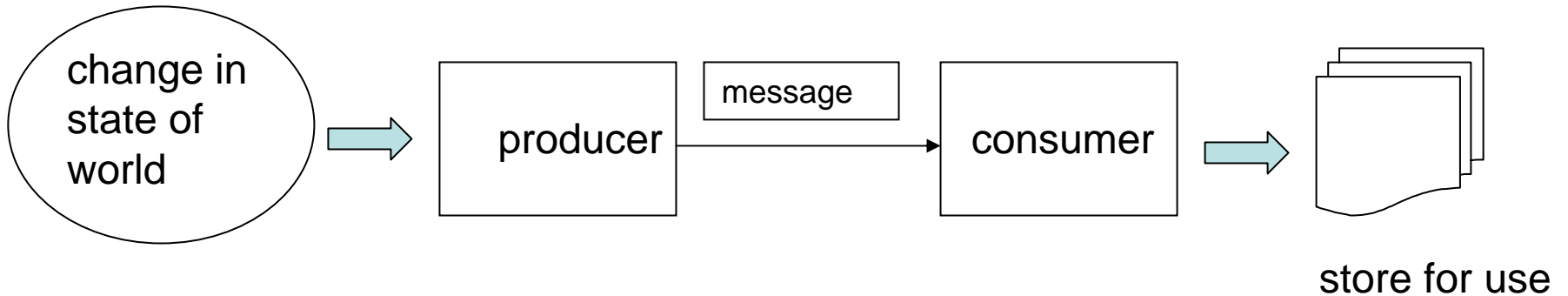
Basic Concepts

- Ontology – data model, delineates the set of valid world state descriptions for the domain of interest (semantics)
- Pragmatic Frame – intended purpose/application context behind data model (pragmatics)
- Formation rules – govern compositions of atomic elements that are well-formed (syntax)

Information Exchange Framework

See Chapter 1:
p9

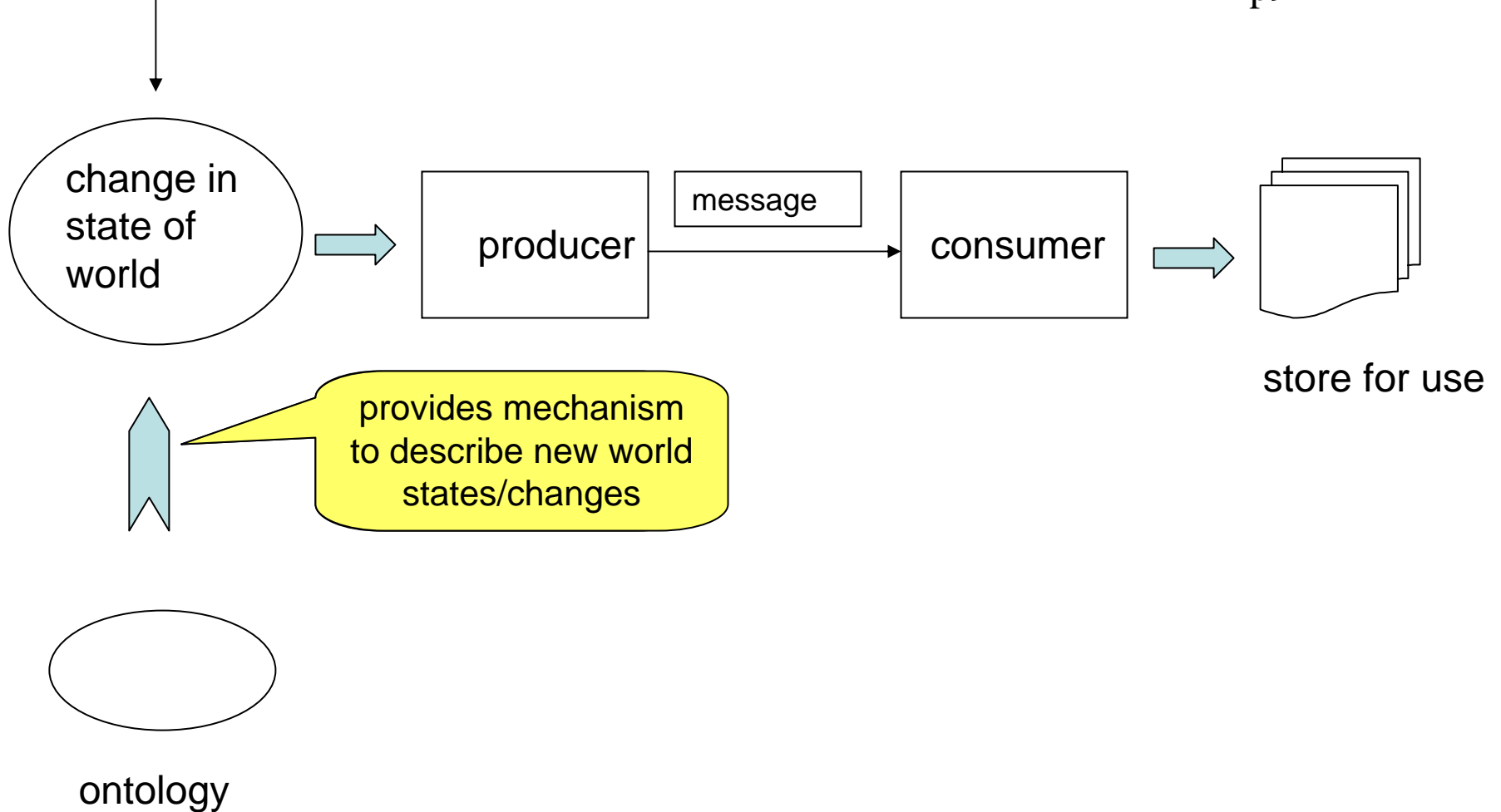
event occurrence



Information Exchange Framework

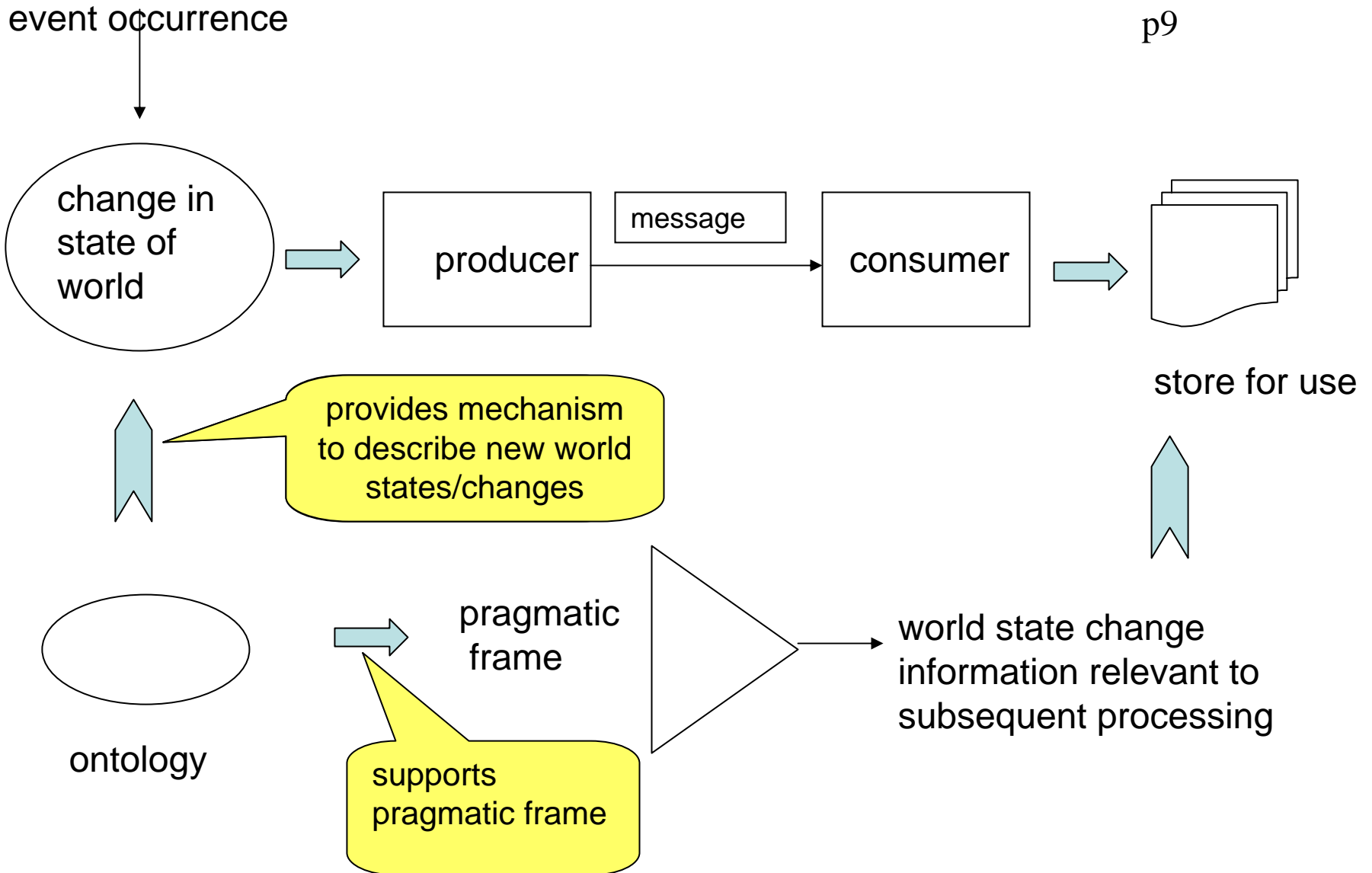
See Chapter 1:
p9

event occurrence



Information Exchange Framework

See Chapter 1:
p9



Example: Car Purchase Notices

See Chapter 1:
p10

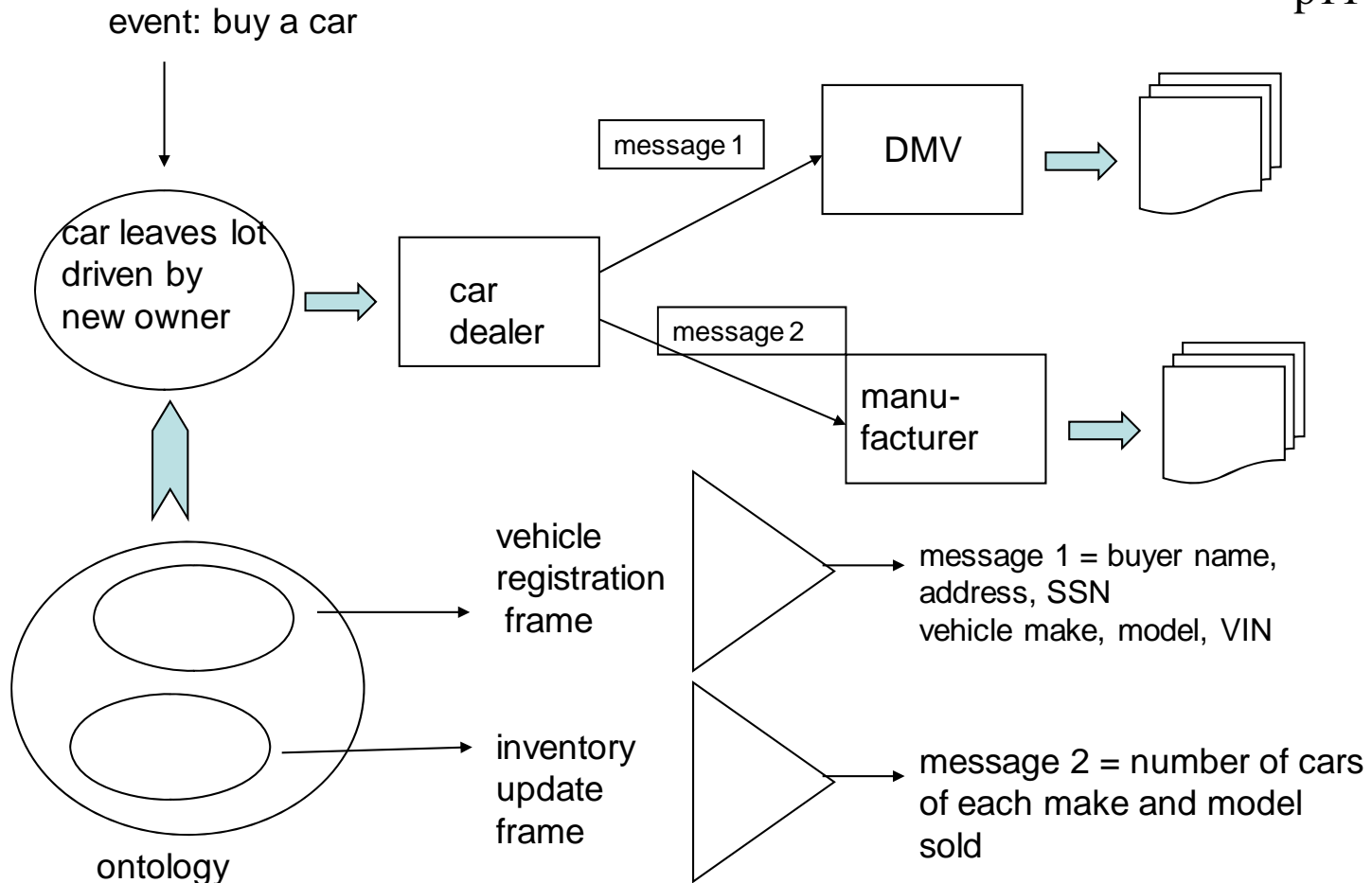
Event causing world state change	Producer	Consumer	Pragmatics: subsequent use	Message Contents
New car purchase—car ownership is transferred from dealer to buyer	Dealer	Department of Motor Vehicles	Register owner's vehicle and give out plates	Buyer and vehicle identification
Same as above	Same as above	Manufacturer Headquarters	Assess inventory levels and production schedule	Number of cars of each make and model that were sold during the month.

The purchase of the car caused a world state change.

The pragmatics determine the ontology underlying the data to be sent

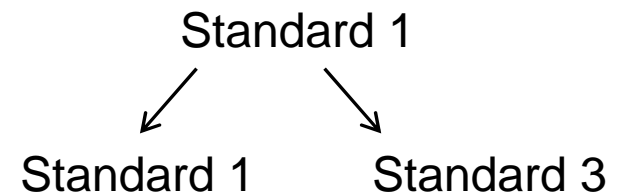
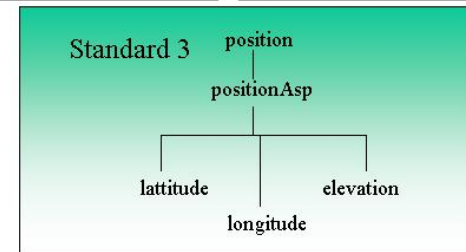
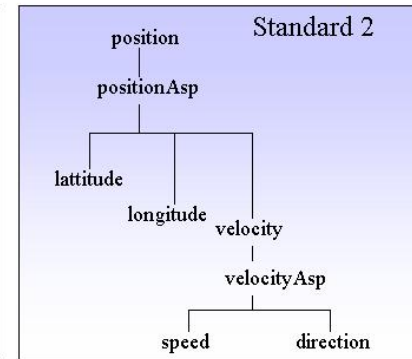
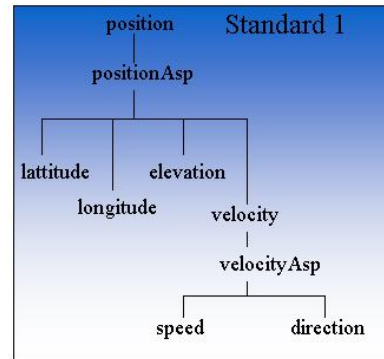
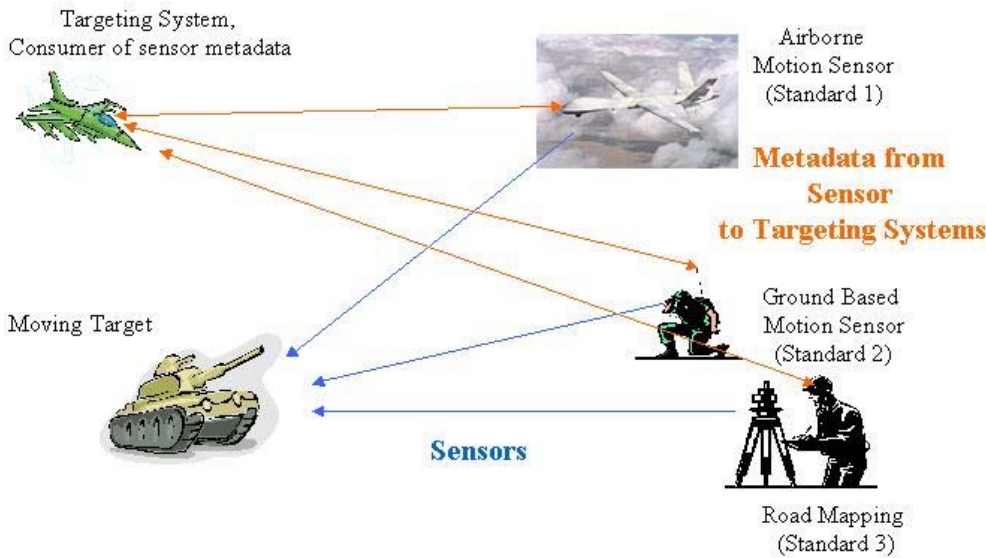
Car Purchase (cont'd)

See Chapter 1:
p11



Example: Harmonizing Standards for Targeting

See Chapter 16:
p300

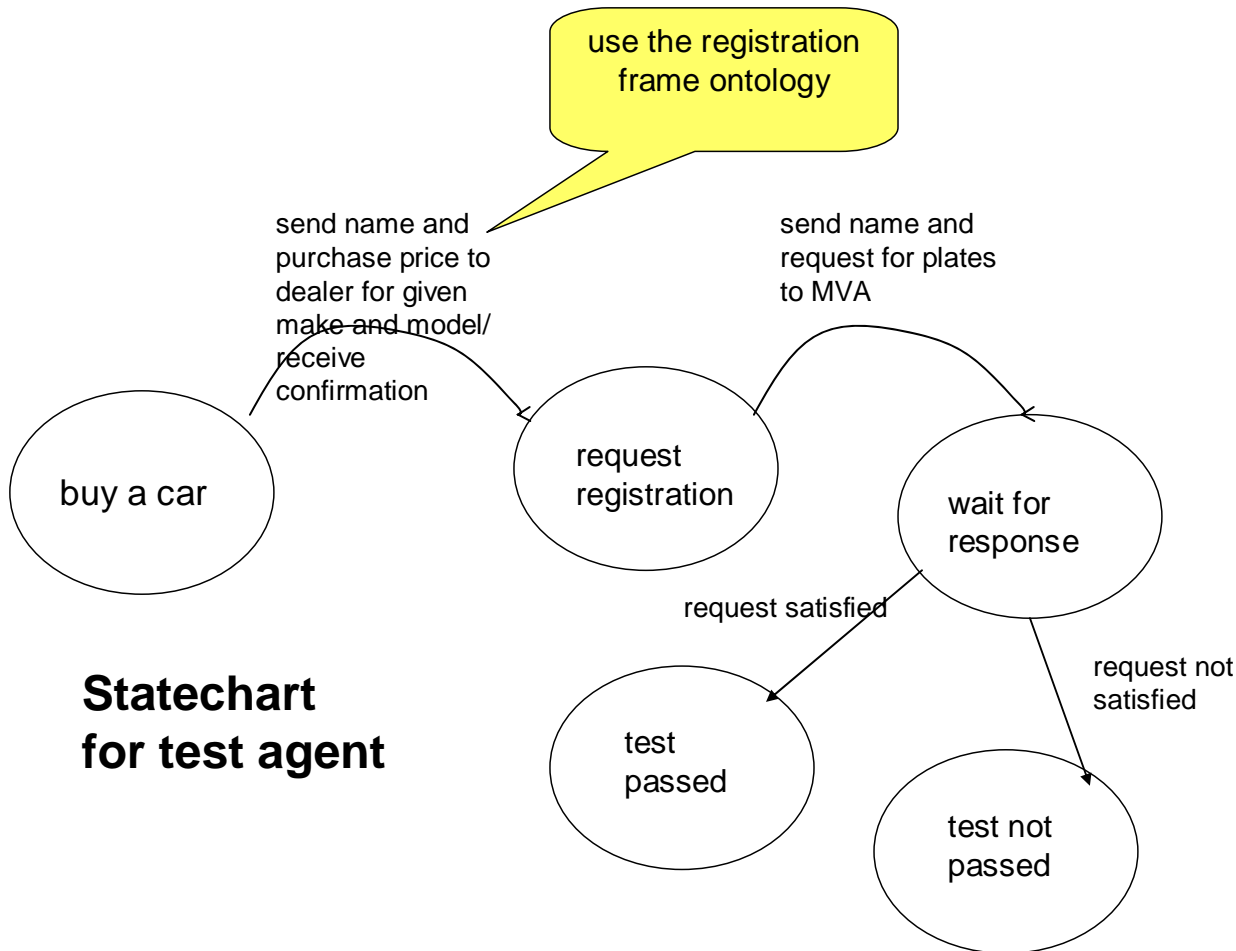


Summary

- An ontology should be designed to support one or more pragmatic frames.
- Such an ontology provides the symbolic means (e.g., logical language or XML) to describe world state changes that a producer sends in the contents of a message to a consumer.
- The contents of the message encode a state changes in a way that can be subsequently processed by a consumer for its use in one of the supported frames.
- The data engineer must decide on the proper balance between conciseness and extensibility in such ontology design.

Testing Ontology in Pragmatic Frame

See Chapter 16:
p300



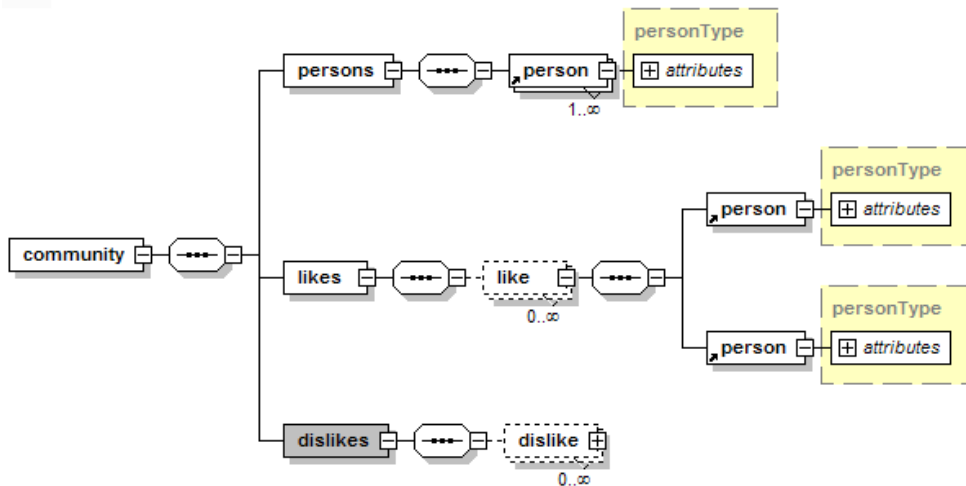
Vehicle Registration Frame:

For correct processing, the make, model and Vehicle Identification Number (VIN) must be filed under the buyer's name.

Later when the buyer appears at the Department of Motor Vehicles (DMV), the clerk can retrieve the vehicle information by querying the buyer's name and proceed to compute the correct tax.

XML Schema and Instances – Island Community Example

See Chapter 2
p19



Schema

```
<xs:element name="community">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="persons"/>
      <xs:element ref="likes"/>
      <xs:element ref="dislikes"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Instance

```
<community>
  <persons>
    <person name="a"></person>
    <person name="b"></person>
    <person name="c"></person>
  </persons>

  <likes>
    <like>
      <person name="a"></person>
      <person name="b"></person>
    </like>
    <like>
      <person name="a"></person>
      <person name="c"></person>
    </like>
  </likes>

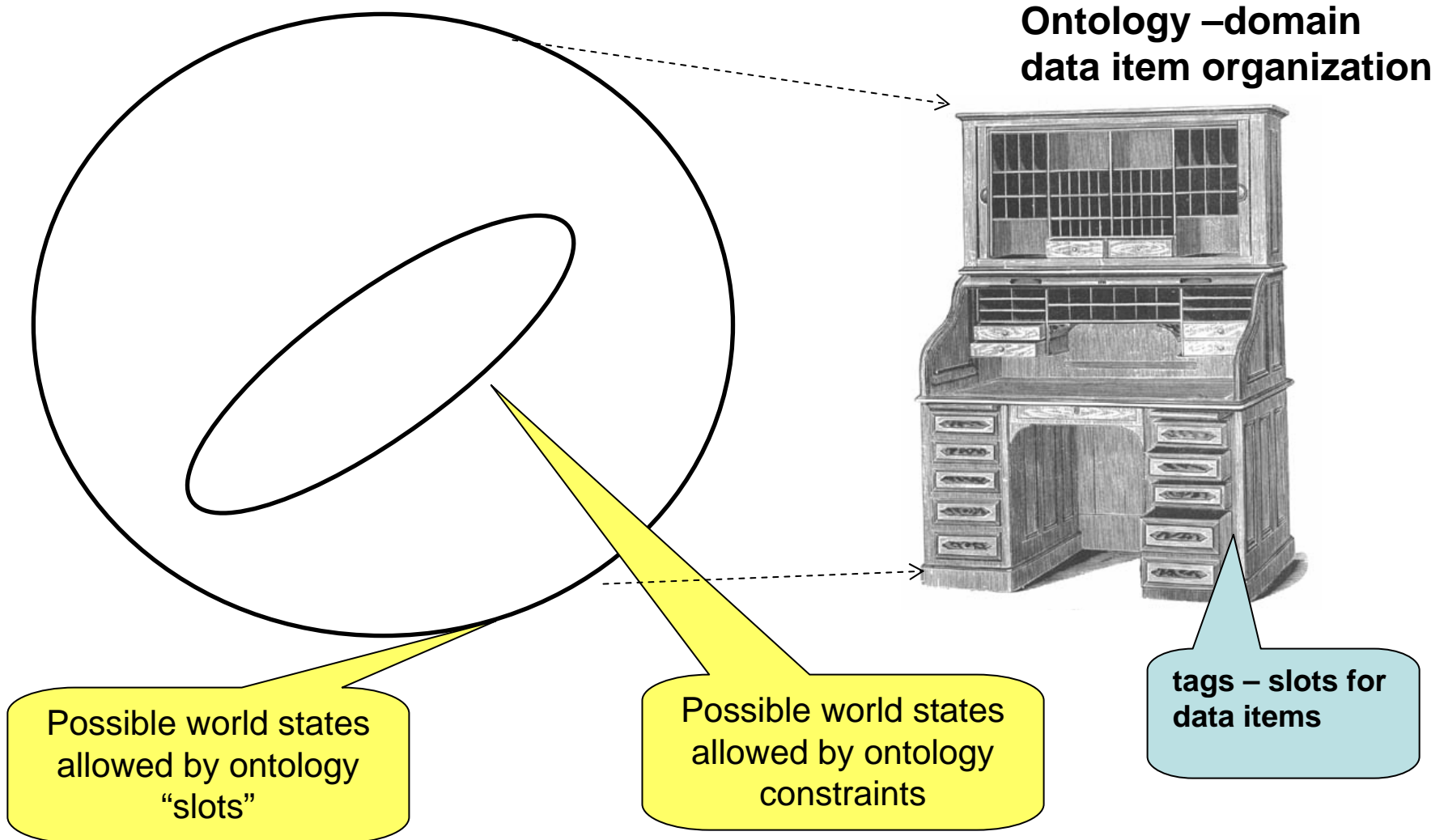
  <dislikes>
    <dislike>
      <person name="b"></person>
      <person name="c"></person>
    </dislike>
  </dislikes>
</community>
```

Problems with XML-only approach – can send this data

```
<dislike>  
<person name="a"></person>  
<person name="a"></person> → Person a dislikes himself  
</like>
```

```
<like>  
<person name="a"></person>  
<person name="d"></person> → Person a likes person d – not a  
</like> member of the community
```

Model-Based Data Approach



Axioms (Constraints) for a Community Social Relations Ontology

No one likes or dislikes himself or herself

Affection is mutual and so is disaffection (for example, you like someone who likes you, and the same applies to disliking someone)

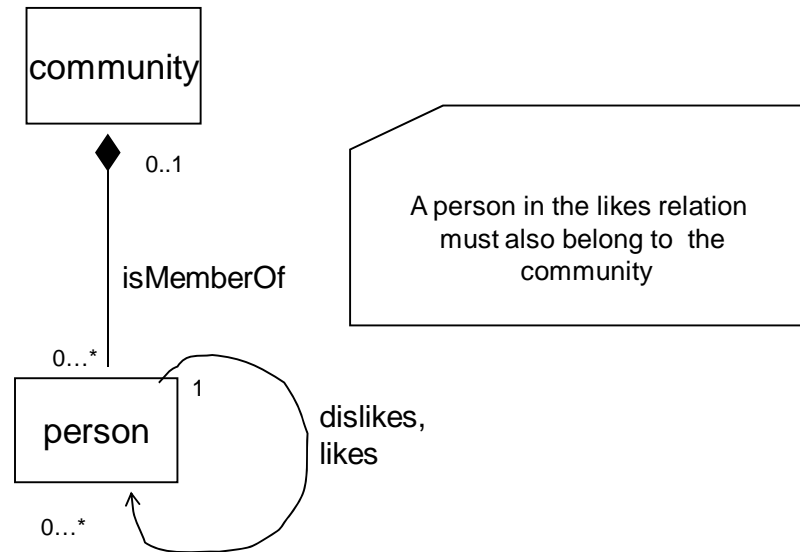
Liking and disliking are mutually exclusive—you can't like and dislike someone at the same time

To quote an ancient proverb, “The enemy of my enemy is my friend.”

Interpreting an XML Schema as an Ontology

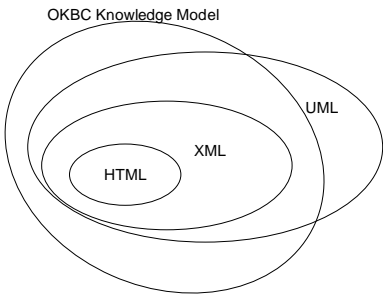
XML segment	Maps to
<pre> <persons> <person name="x1"/> <person name="x2"/> ... <person name="xn"/></persons> </pre>	<p>community members = $\{x_1, x_2, \dots, x_n\}$</p>
<pre> <likes> <like><person name="x1"/><person name="x2"/></like> ... <like><person name="xn"/><person name="xm"/></like> </likes> </pre>	<p>Likes = $\{(x_1, x_2), \dots, (x_n, x_m)\}$</p>
ditto for dislikes	

UML is also limited in constraint expression



Languages/Environments for Ontology Development

See Chapter 2
p16



Language/ Environment for Developing Ontology	Features	Tools
XML (dtd)	tags and attributes relate to object classes and instance variables	Syntax –well-formedness checker –validity checker
XML (schema)	–simple data types (strings, integers, etc.) –complex data types (composites of simple ones) with regular expression restrictions –element multiplicities	validity checker extended to check –whether data conforms to type specification –if all multiplicities are honored
UML 1.4	objects, classes, attributes, methods, generalization, associations with multiplicities, interactions, state charts	–methods check for consistency among various diagrams (class, interaction, collaboration, etc) –XMI provides mapping into XML (See Chapter 19 for consideration of UML 2.0)
OKBC (Open Knowledge Base Connectivity) Knowledge Model	–object-oriented representation of knowledge –constants, frames, slots, facets, classes, individuals, and knowledge bases –specified within Knowledge Interchange Format (KIF), a first-order predicate logic language with set theory	Protégé, Ontolingua
OWL	-classes, attributes, class hierarchies -properties (standalone binary relations)	Protégé, Ontolingua (See Chapters 13 and 19 for further consideration of OWL)

Summary

- We have briefly reviewed some commonly employed languages/environments for ontology development
- Relative rankings of frameworks reflect the distinction between lightweight and heavyweight ontologies
- Heavyweight ontologies include axioms and constraints to restrict the ontology's set of models
- From this point of view, XML by itself is a lightweight ontology; UML is more toward the heavyweight side; and a logic-based framework is a true heavyweight