

Software Modeling Techniques and the Semantic Web

Jin Song DONG

(www.comp.nus.edu.sg/~dongjs)

Computer Science Department

National University of Singapore

(Joint work with Yuan Fang LI, Hai WANG and others)

May 2004

Objectives

- To learn Software Modeling Techniques, i.e., Z, Alloy ...
- To learn Semantic Web Languages, i.e., RDF, DAML ...
- To study the Connections Between the Two Areas.

Semantic Web

“The Semantic Web is an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. It is the idea of having data on the Web defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications.” – W3C (www.w3.org/2001/sw)

Semantic Web is the main focus of WWW'04 May 18-22 2004 (last week at NYC).

Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- Semantic Web Environment for Specifications
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - Semantics of DAML+OIL in Z/Alloy
 - Combined Approach to Reasoning about Semantic Web
- Conclusion and Further Work

4

Unified Modeling Language

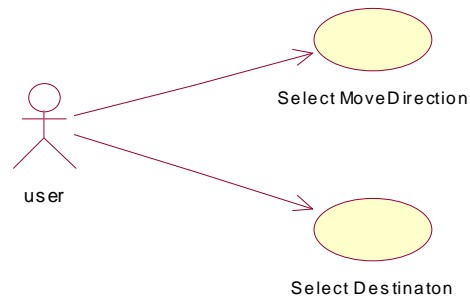
The Unified Modeling Language (UML) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems.

- UML =
 - Booch (G. Booch) + OMT (J. Rumbaugh ...) + OOSE (I. Jacobson) +
 - Statechart (D. Harel) + Syntropy (S. Cook ...) + Catalysis (A. Wills) ...
- Diagrams: Class diagrams, Statechart diagrams, Sequence diagrams, Collaboration diagrams, ...
- Formalism: Object Constraint Languages, like VDM/Z.

5

Use Case Diagram

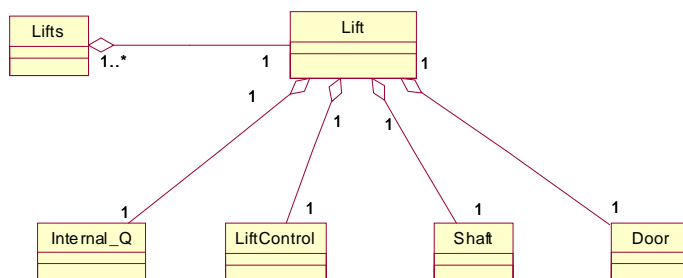
- Use case is a pattern of behavior the system exhibits. Each use case is a sequence of related transactions performed by an actor and the system in a dialogue. Actors are examined to determine their needs. Use case diagrams are created to visualise the relationships between actors and use cases



6

Class Diagram

- A class diagram shows the existence of classes and their relationships in the logical view of a system. It consists of classes and their structure and behavior, association, aggregation, dependency, and inheritance relationships, multiplicity and navigation indicators, and role names



7

Collaboration Diagram – dynamic behavior, message-oriented

- A collaboration diagram displays object interactions organised around objects and their links to one another

Statechart Diagram – dynamic behavior, event-oriented

- A statechart diagram shows the life history of a given class, the events that cause a transition from one state to another, and the actions that result from a state change

8

The Z Specification Language

- developed originally at Programming Research Group, Oxford University
- based on set theory and predicate logic
- system described by introducing fixed sets and variables and specifying the relationships between them using predicates
- declarative, not procedural
- system state determined by values taken by variables subject to restrictions imposed by state invariant
- operations expressed by relationship between values of variables before, and values after, the operation
- variable declarations and related predicates encapsulated into schemas
- schema calculus facilitates the composition of complex specifications
- J. Woodcock and J. Davies, Using Z: Specification, Refinement, and Proof. Prentice-Hall, 1996

9

Types

Z is strongly typed: every expression is given a type.

Any set can be used as a type.

The following are equivalent within set comprehension

$$\begin{aligned} &(x, y) : A \times B \\ &x : A; y : B \\ &x, y : A \quad (\text{when } B = A) \end{aligned}$$

Notice that

$$\forall S : \mathbb{P} A \bullet \dots \quad \text{not} \quad \forall S \subseteq A \bullet \dots$$

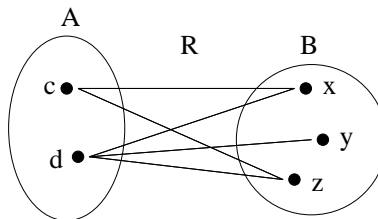
10

Relations

A relation R from A to B , denoted by

$$R : A \leftrightarrow B,$$

is a subset of $A \times B$.



$$R \text{ is the set } \{(c, x), (c, z), (d, x), (d, y), (d, z)\}$$

Notation: the predicates

$$(c, z) \in R \quad \text{and} \quad c \mapsto z \in R \quad \text{and} \quad c \underline{R} z$$

are equivalent.

$$\begin{aligned} \text{dom } R &\text{ is the set } \{a : A \mid \exists b : B \bullet a \underline{R} b\} \\ \text{ran } R &\text{ is the set } \{b : B \mid \exists a : A \bullet a \underline{R} b\} \end{aligned}$$

Examples

11

$$\frac{- \leq - : \mathbb{N} \leftrightarrow \mathbb{N}}{\forall x, y : \mathbb{N} \bullet x \leq y \Leftrightarrow \exists k : \mathbb{N} \bullet x + k = y}$$

i.e. the relation \leq is the infinite subset

$$\{(0, 0), (0, 1), (1, 1), (0, 2), (1, 2), (2, 2), \dots\}$$

of ordered pairs in $\mathbb{N} \times \mathbb{N}$.

$$\frac{\text{divides} : \mathbb{N}_1 \leftrightarrow \mathbb{N}}{\forall x : \mathbb{N}_1; y : \mathbb{N} \bullet x \text{ divides } y \Leftrightarrow \exists k : \mathbb{N} \bullet x k = y}$$

$$3 \text{ divides } 6 \quad \text{but} \quad \neg (3 \text{ divides } 7)$$

Domain and Range Restriction/Subtraction

Suppose $R : A \leftrightarrow B$ and $S \subseteq A$ and $T \subseteq B$; then

$$\begin{aligned} S \triangleleft R & \text{ is the set } \{(a, b) : R \mid a \in S\} \\ R \triangleright T & \text{ is the set } \{(a, b) : R \mid b \in T\} \end{aligned}$$

$$\begin{aligned} S \triangleleft R & \text{ is the set } \{(a, b) : R \mid a \notin S\} \\ R \triangleright T & \text{ is the set } \{(a, b) : R \mid b \notin T\} \end{aligned}$$

e.g. if

$$\text{has_sibling} : \text{People} \leftrightarrow \text{People} \quad \text{then}$$

$$\begin{aligned} \text{female} \triangleleft \text{has_sibling} & \text{ is the relation } \text{is_sister_of} \\ \text{has_sibling} \triangleright \text{female} & \text{ is the relation } \text{has_sister} \end{aligned}$$

$$\begin{aligned} \text{female} \triangleleft \text{has_sibling} & \text{ is the relation } \text{is_brother_of} \\ \text{has_sibling} \triangleright \text{female} & \text{ is the relation } \text{has_brother} \end{aligned}$$

Relational Image

Suppose $R : A \leftrightarrow B$ and $S \subseteq A$

$$R(S) = \{b : B \mid \exists a : S \bullet a R b\}$$

$$R(S) \subseteq B$$

$$\begin{aligned} \text{divides}(\{8, 9\}) &= \{x : \mathbb{N} \mid \exists k : \mathbb{N} \bullet x = 8k \vee x = 9k\} \\ &= \{\text{numbers divided by 8 or 9}\} \end{aligned}$$

$$\leq(\{7, 3, 21\}) = \{x : \mathbb{N} \mid x \geq 3\}$$

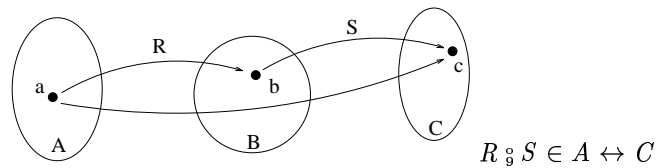
$$\text{has_sibling}(\text{male}) = \{\text{people who have a brother}\}$$

14

Relational Composition

Suppose $R : A \leftrightarrow B$ and $S : B \leftrightarrow C$

$$\begin{aligned} R \circ S &= \{(a, c) : A \times C \mid \exists b : B \bullet a R b \wedge b S c\} \end{aligned}$$



e.g.

$$\text{is_parent_of} \circ \text{is_parent_of} = \text{is_grandparent_of}$$

$$R^0 = \text{id}[A], \quad R^1 = R, \quad R^2 = R \circ R, \quad R^3 = R \circ R \circ R, \dots$$

15

Functions

A (partial) function f from a set A to a set B , denoted by

$$f : A \rightarrow B,$$

is a subset f of $A \times B$ with the property that for each $a \in A$ there is at most one $b \in B$ with $(a, b) \in f$. The function f is a *total* function, denoted

$$f : A \rightarrow B,$$

if and only if $\text{dom } f$ is the set A .

The predicates

$$(a, b) \in f \quad \text{and} \quad f(a) = b$$

are equivalent.

Sequences

A sequence s of elements from a set A , denoted

$$s : \text{seq } A,$$

is a function $s : \mathbb{N} \rightarrow A$ where $\text{dom } s = 1 \dots n$ for some natural number n . For example,

$$\langle b, a, c, b \rangle \text{ denotes the sequence (function) } \{1 \mapsto b, 2 \mapsto a, 3 \mapsto c, 4 \mapsto b\}$$

The empty sequence is denoted by $\langle \rangle$.

The set of all sequences of elements from A is denoted $\text{seq } A$ and is defined to be

$$\text{seq } A == \{s : \mathbb{N} \rightarrow A \mid \exists n : \mathbb{N} \bullet \text{dom } s = 1 \dots n\}$$

We define $\text{seq}_1 A$ to be the set of all non-empty sequences, i.e.

$$\text{seq}_1 A == \text{seq } A - \{\langle \rangle\}$$

Notice that: $\langle a, b, a \rangle \neq \langle a, a, b \rangle \neq \langle a, b \rangle$

Special Functions for Sequences

Concatenation

$$\langle a, b \rangle \hat{\ } \langle b, a, c \rangle = \langle a, b, b, a, c \rangle$$

Head, Last

$$\left| \begin{array}{l} \text{head, last : seq}_1 A \rightarrow A \\ \hline \forall s : \text{seq}_1 A \bullet \text{head}(s) = s(1) \wedge \text{last}(s) = s(\#s) \end{array} \right.$$

$$\text{head}\langle c, b, b \rangle = c \quad \text{last}\langle c, b, b \rangle = b$$

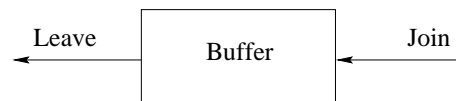
Tail

$$\left| \begin{array}{l} \text{tail : seq}_1 A \rightarrow \text{seq } A \\ \hline \forall s : \text{seq}_1 A \bullet \langle \text{head}(s) \rangle \hat{\ } \text{tail}(s) = s \end{array} \right.$$

$$\text{tail}\langle c, b, b \rangle = \langle b, b \rangle$$

18

Z Schemas: A Message Buffer Example



- A number of messages are transmitted from one location to another.
- Because of other traffic on the line each message for transmission is placed in a buffer which outputs the message when the line is free.
- This buffer may contain several messages at any time, but there is a fixed upper limit on the number of messages the buffer may contain.
- The buffer operates on a first in/first out (FIFO) principle.

19

Formal Specification

The State Schema

$[MSG]$ (The exact nature of these messages is not important)

is the set of all possible messages that could ever be transmitted.

| $max : \mathbb{N}$ (The actual value of max is not important)

is the constant maximum number of messages that can be held in the buffer at any one time.

$Buffer$	
$items : seq\ MSG$	declaration
$\#items \leq max$	predicate

e.g. suppose $MSG = \{m_1, m_2, m_3\}$ and $max = 4$

Then $items = \langle m_1, m_2 \rangle$ is an instance, but $items = \langle m_3, m_1, m_1, m_2, m_2 \rangle$ is not

Schema Inclusion and Operation/Initial Schemas

$\Delta Buffer$	$\Delta Buffer$
$Buffer$	$items, items' : seq\ MSG$
$Buffer'$	$\#items \leq max \wedge \#items' \leq max$

$Join$	$Leave$	$Buffer_{INIT}$
$\Delta Buffer$	$\Delta Buffer$	$Buffer$
$msg? : MSG$	$msg! : MSG$	$items = \langle \rangle$
$\#items < max$	$items \neq \emptyset$	
$items' = items \hat{\ } \langle msg? \rangle$	$items = \langle msg! \rangle \hat{\ } items'$	

Combining Formal Specification with Object-Oriented Design

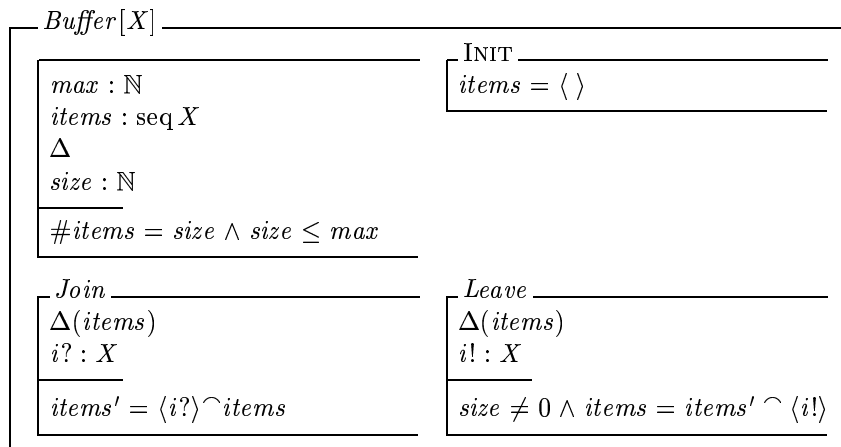
goes against the conventional view of separating the concerns of functionality and design, but

- adds clarity and leads to simplification of large systems;
- helps with system abstraction and suggests a refinement into object-oriented code.

Object-Z is an extension to Z developed at the University of Queensland. It supports the design of object-oriented design.

- R. Duke and G. Rose, Formal Object Oriented Specification Using Object-Z. Cornerstones of Computing Series (editors: R. Bird, C.A.R. Hoare), Macmillan Press, March 2000.
- G. Smith, The Object-Z Specification Language, Kluwer Academic Publishers, 2000.

Object-Z Basics: Buffer Example



Exercise: For this *Buffer* class specify:

- (a) an operation *count* which, given a message, outputs the number of times that message occurs in the buffer;
- (b) an operation *duplicate* which appends to the buffer the message currently at the head of the buffer, provided the buffer is not empty or already full;
- (c) an operation *titanic* whereby a sequence of messages is appended to the buffer except those messages for which there is no room are discarded (the buffer is like a life-boat on the Titanic: people queue to get on, but once the boat is full all the remaining people are left behind);
- (d) an operation *penguin* whereby, like the operation *titanic*, a sequence of messages is input to the buffer, but this time the messages on the end of the sequence are accepted while those at the front are discarded if there is no room (the messages are acting like penguins, pushing out the messages already in the buffer once the buffer is full).

24

Solution

$$\frac{\begin{array}{l} \textit{count} \\ m? : X \\ \textit{count}! : \mathbb{N} \end{array}}{\textit{count}! = \#(\textit{items} \triangleright \{m?\})}$$

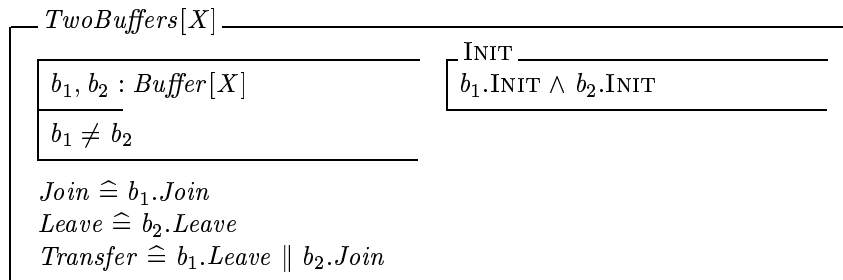
$$\frac{\begin{array}{l} \textit{duplicate} \\ \Delta(\textit{items}) \end{array}}{\begin{array}{l} \# \textit{items} \in 1..(\textit{max} - 1) \\ \textit{items}' = \textit{items} \hat{\ } \langle \textit{head} \textit{items} \rangle \end{array}}$$

$$\frac{\begin{array}{l} \textit{titanic} \\ \Delta(\textit{items}) \\ s? : \textit{seq} X \end{array}}{\textit{items}' = (1.. \textit{max}) \triangleleft (\textit{items} \hat{\ } s?)}$$

$$\frac{\begin{array}{l} \textit{penguin} \\ \Delta(\textit{items}) \\ s? : \textit{seq} X \end{array}}{\begin{array}{l} \exists s : \textit{seq} X \bullet \\ s \hat{\ } \textit{items}' = \textit{items} \hat{\ } s? \\ s \neq \langle \rangle \Rightarrow \# \textit{items}' = \textit{max} \end{array}}$$

25

Two Linked Buffers (single thread)



26

Alloy Overview

Alloy (developed at MIT by D. Jackson's group) is a structural modelling language based on first-order logic (a subset of Z) and specifications organised in a tree of *modules*

Signature: A signature (**sig**) paragraph introduces a basic type and a collection of relation (called field) in it along with the types of the fields and constraints on their value. A signature may inherit fields and constraints from another signature.

Function: A function (**fun**) captures behaviour constraints. It is a parameterised formula that can be "applied" elsewhere,

Fact: Fact (**fact**) constrains the relations and objects. A **fact** is a formula that takes no arguments and need not to be invoked explicitly; it is always true.

Assertion: An assertion (**assert**) specifies an intended property. It is a formula whose correctness needs to be checked, assuming the facts in the model.

27

Alloy Analyser (AA)

- Constraint solver with automated simulation & checking
- Transforms a problem into a (usually huge) boolean formula
- A *scope* (finite bound) must be given

28

Alloy Basics

x (a scalar), $\{x\}$ (a singleton set containing a scalar), (x) (a tuple) and $\{(x)\}$ (a relation) are all treated as the same as $\{x\}$. The relational composition (or join) and product:

$$\{(X_1, \dots, X_m, S)\} \cdot \{(S, Y_1, \dots, Y_n)\} = \{(X_1, \dots, X_m, Y_1, \dots, Y_n)\}$$

$$\{(X_1, \dots, X_m, S)\} \rightarrow \{(S, Y_1, \dots, Y_n)\} = \{(X_1, \dots, X_m, S, S, Y_1, \dots, Y_n)\}$$

29

Alloy Expression Examples

```
children = ~parents
ancestors = ^parents
descendants = ~ancestors
Man = Person - Woman
mother = parents & (Person->Woman)
father = parents & (Person->Man)
siblings = parents.~parents - iden [Person]
cousins = grandparents.~grandparents - siblings - iden [Person]
```

30

Alloy Logical Operators

```
!F // negation: not F
F && G // conjunction: F and G
F || G // disjunction: F or G
F => G // implication: F implies G; same as !F || G
F <=> G // biimplication: F when G; same as F =>G && G => F
F => G,H // if F then G else H; same as F => G && !F => H
```

Quantifiers

```
all x: e | F
some x: e | F
no x: e | F
sole x: e | F
one x: e | F
one x:e, y:f | F
all disj x,y: e | F
```

31

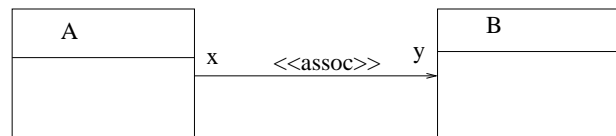
Examples

```
// no polygamy
all p: Person | sole p.spouse
// a married person is his or her spouse's spouse
all p: Person | some p.spouse => p.spouse.spouse = p
// no incest
no p: Person | some (p.spouse.parents & p.parents)
// a person's siblings are those persons with the same parents
all p: Person | p.siblings = {q: Person | q.parents = p.parents} - p
// everybody has one mother
all p: Person | one p.parents & Woman
// somebody is everybody's ancestor
some x: Person | all p: Person | x in p.*parent
```

32

Alloy, UML and Z

Given the UML Class diagram



The corresponding Alloy expression:

```
assoc: A x -> y B
```

Given the Z expressions, the corresponding Alloy expressions:

```
in Z:  $T_1 \rightarrow T_2$ 
```

```
in Alloy:  $T_1 \rightarrow! T_2$ 
```

```
in Z:  $T_1 \leftrightarrow T_2$ 
```

```
in Alloy:  $T_1 \rightarrow? T_2$ 
```

33

Module, Sig, Fact, Fun and Assert (example)

```
module CeilingsAndFloors
sig Platform {}
sig Man {ceiling, floor: Platform}
fact {all m: Man | some n: Man | Above (n,m)}
fun Above (m, n: Man) {m.floor = n.ceiling}
assert BelowToo {all m: Man | some n: Man | Above (m,n)}
run Above for 2
check BelowToo for 2
```

34

CSP/Timed CSP

- Hoare's CSP (Communicating Sequential Processes) an *event* based notation primarily aimed at describing the sequencing of behaviour within a process and the synchronisation of behaviour (or *communication*) between processes.
- Timed CSP extends CSP by introducing a capability to quantify temporal aspects of sequencing and synchronisation.
- S. Schneider. *Concurrent and Real-time Systems: The CSP Approach*, Wiley, 1999.
- A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.
- C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall International, 1985.

35

Prefix

A process which may participate in event a then act according to process description P is written

$$a@t \rightarrow P(t).$$

Other CSP/Timed-CSP primitives:

- $P; Q$ (sequential composition)
- $P \parallel [X] Q$ (synchronous), $P \parallel\parallel Q$ (asynchronous)
- $a \rightarrow P \square b \rightarrow Q$ (external choice), $a \rightarrow P \sqcap b \rightarrow Q$ (internal choice)
- $P_1 \nabla e \rightarrow P_2$ (interrupt process)
- $\text{WAIT } t; P$ (delay), $a \rightarrow P \triangleright\{t\} Q$ (time-out)

Channel

A channel is a collection of events of the form $c.n$: the prefix c is called the *channel name* and the collection of suffixes is called the *values* of the channel.

When an event $c.n$ occurs it is said that *the value n is communicated on channel c* . When the value of a communication on a channel is determined by the environment (external choice) it is called an *input* and when it is determined by the internal state of the process (internal choice) it is called an *output*.

Recursion

Recursion is used to given finite representations of non-terminating processes. The process expression

$$\mu P \bullet a?n : \mathbb{N} \rightarrow b!f(n) \rightarrow P$$

describes a process which repeatedly inputs a natural on channel a , calculates some function f of the input, and then outputs the result on channel b .

Recall Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- ✓ Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- Semantic Web Environment for Software Specifications
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - Semantics of DAML+OIL in Z/Alloy
 - Combined Approach to Reasoning about Semantic Web
- Conclusion and Further Work

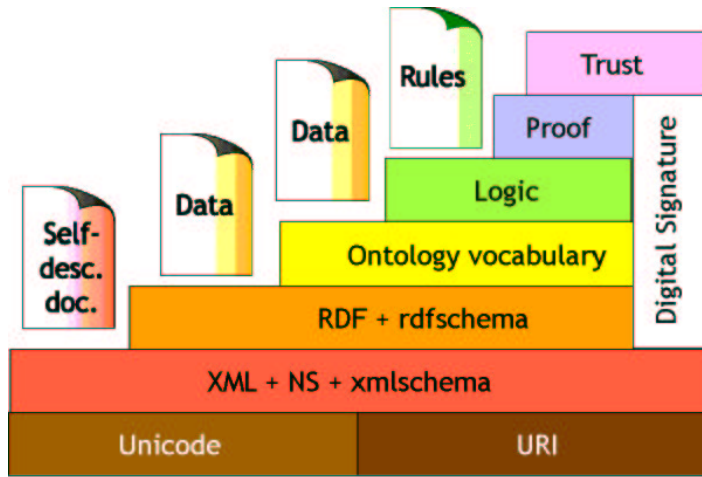
38

Semantic Web

- Goals
 - Realizing the full potential of the Web
 - Making it possible for tools (agents) to effectively process information.
 - Ultimate goal - effective and efficient global information/knowledge exchange
- Building on proven ideas
 - Combines XML, RDF, hypertext and metadata approaches to linked information
 - Focuses on general principles of Web automation and data aggregation

39

Semantic Web Architectural Dependencies



www.w3c.org (by Tim Berners-Lee)

40

WWW'04 Observations

- Semantic Web is the most popular topic (Tim attended every Semantic Web session)
- More tools are reported with major companies involved (IBM, HP ...)
- Some showcase applications are demonstrated (magazine, museum, biomedical ...)
- New companies in specializing Semantic Web appeared (e.g. Semagix)

41

RDF, DAML+OIL and OWL

- Resource Description Framework (RDF) — 1999
 - An RDF document is a collection of assertions in *subject verb object* form for describing web resources
 - Provides interoperability between applications that exchange machine-understandable information on the Web
 - Use XML as a syntax, include XMLNS, and URIs
- DARPA Agent Markup Language (DAML+OIL) — 2001
 - Semantic markup language based on RDF, and
 - Extends RDF(S) with richer modelling primitives
 - DAML combines Ontology Interchange Language (OIL).
- OWL Web Ontology Language — 2003 (become W3C rec)
 - Based on DAML+OIL
 - Three levels support: Lite, DL, Full

42

HTML and XML

- HTML

```
<H1> Semantic Web and Formal Methods</H1>
<UL>
  <LI> Teacher: Jin Song Dong
  <LI> Students: s19908, s20015
  <LI> Requirements: discrete maths
</UL>
```
- XML

```
<course>
  <title> Semantic Web and Formal Methods </title>
  <teacher> Jin Song Dong </teacher>
  <students> s19908, s20015 </students>
  <req> discrete maths </req>
</course>
```

43

Lack semantics in XML

- The XML is accepted as the emerging standard for data interchange on the Web. XML allows authors to create their own markup (e.g. `<course>`), which seems to carry some semantics.
- However, from a computational perspective tags like `<course>` carries as much semantics as a tag like `<H1>`. A computer simply does not know, what a course is and how the concept course is related to other concepts.
- XML may help humans predict what information might lie “between the tags” in the case of `<students> </students>`, but XML can only help.
- Only feasible for closed collaboration, e.g., agents in a small and stable community/intranet

RDF Basics

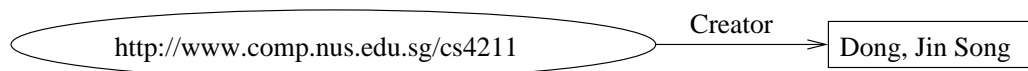
- Resources — Things being described by RDF expressions. Resources are always named by URIs, e.g.
 - HTML Document
 - Specific XML element within the document source.
 - Collection of pages
- Properties — Specific aspect, characteristic, attribute or relation used to describe a resource, e.g. Creator, Title ...
- Statements —
Resource (Subject) + Property (Predicate) + Property Value (Object)

RDF Statement Example 1

Dong, Jin Song is the creator of the web page

`http://www.comp.nus.edu.sg/cs4211`

- Subject (Resource) - `http://www.comp.nus.edu.sg/cs4211`
- Predicate (Property) - Creator
- Object (Literal) Dong, Jin Song

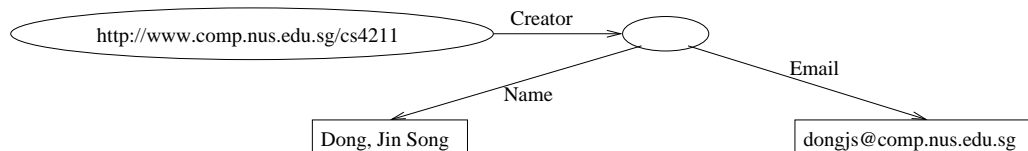


46

RDF Statement Example 2

Dong, Jin Song whose e-mail is `dongjs@comp.nus.edu.sg` is the creator of the web

page `http://www.comp.nus.edu.sg/cs4211`



47

RDF in XML syntax

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

  <rdf:Description about="http://www.comp.nus.edu.sg/cs4211">
    <dc:creator>Dong, Jin Song</dc:creator>
    <dc:title>Advanced Software Engineering</dc:title>
    <dc:date>2000-07-01</dc:date>
  </rdf:Description>

</rdf:RDF>
```

48

RDF Containers

- Bag - An unordered list of resources or literals
- Sequence - An ordered list of resources or literals
- Alternative - A list of resources or literals that represent alternatives for the value of a property

49

Container example: Sequence

Statement: The students of the course CS4211 in alphabetical order are Yuanfang Li, Jun Sun and Hai Wang .

```
<rdf:RDF xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:s="http://www.schemas.org/Course/">
  <rdf:Description about=http://www.comp.nus.edu.sg/~cs4211>
    <s:students>
      <rdf:Seq>
        <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~liyf"/>
        <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~sunj"/>
        <rdf:li rdf:resource="http://www.comp.nus.edu.sg/~wangh"/>
      </rdf:Seq>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

50

RDF Schema

- Basic vocabulary to describe RDF vocabularies, e.g.,
Class, subclassOf, Property, subPropertyOf, domain, range
- Defines properties of the resources (e.g., title, author, subject, etc)
- Defines kinds of resources being described (books, Web pages, people, etc)
- XML Schema gives specific constraints on the structure of an XML document
RDF Schema provides information about the interpretation of the RDF statements
- RDF schema uses XML syntax, but could theoretically use any other syntax

51

RDF Schema Example (Class)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <rdfs:Class rdf:ID="Person">
    <rdfs:comment>Person Class</rdfs:comment>
    <rdfs:subClassOf
      rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Resource"/>
  </rdfs:Class>

  <rdfs:Class rdf:ID="Student">
    <rdfs:comment>Student Class</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Person"/>
  </rdfs:Class>
```

52

RDF Schema Example (Property)

```
<rdf:Property rdf:ID="teacher">
  <rdfs:comment>Teacher of a course</rdfs:comment>
  <rdfs:domain rdf:resource="#Course"/>
  <rdfs:range rdf:resource="#Person"/>
</rdf:Property>

<rdf:Property rdf:ID="students">
  <rdfs:comment>List of Students in alphabetical order</rdfs:comment>
  <rdfs:domain rdf:resource="#Course"/>
  <rdfs:range rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Seq"/>
</rdf:Property>
```

53

Why RDF(S) is not enough

- Only range/domain constraints on properties (need others)
- No properties of properties (unique, transitive, inverse, etc.)
- No equivalence, disjointness, etc.
- No necessary and sufficient conditions (for class membership)

DAML+OIL

- Europe: Ontology Inference Language (OIL) extends RDF Schema to a fully-fledged knowledge representation language.
- US: DARPA Agent Markup Language (DAML)
- Merged as DAML+OIL in 2001
 - logical expressions
 - data-typing
 - cardinality
 - quantifiers
- Becomes OWL — W3C 2004

DAML: Setting up the namespaces

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
>
```

DAML: Define Classes

```
<rdfs:Class rdf:ID="Animal"> <rdfs:label>Animal</rdfs:label> </rdfs:Class>
<rdfs:Class rdf:ID="Male">
  <rdfs:subClassOf rdf:resource="#Animal"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Female">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <daml:disjointWith rdf:resource="#Male"/>
</rdfs:Class>
<rdfs:Class rdf:ID="Man">
  <rdfs:subClassOf rdf:resource="#Person"/>
  <rdfs:subClassOf rdf:resource="#Male"/> </rdfs:Class>
```

DAML: Define Properties

```
<rdf:Property rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="#Animal"/>
</rdf:Property>
<rdf:Property rdf:ID="hasFather">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Male"/>
</rdf:Property>
```

DAML: Define Restrictions

```
<rdfs:Class rdf:ID="Person"> <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasParent"/>
      <daml:toClass rdf:resource="#Person"/>
    </daml:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#hasFather"/>
    </daml:Restriction> </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:maxcardinality="1">
      <daml:onProperty rdf:resource="#hasSpouse"/>
    </daml:Restriction> </rdfs:subClassOf>
</rdfs:Class>
```

DAML: UniqueProperty and Transitive

```
<daml:UniqueProperty rdf:ID="hasMother">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Female"/>
</daml:UniqueProperty>
```

```
<daml:TransitiveProperty rdf:ID="hasAncestor">
  <rdfs:label>hasAncestor</rdfs:label>
</daml:TransitiveProperty>
```

60

DAML: oneOf

```
<rdf:Property rdf:ID="hasHeight">
  <rdfs:range rdf:resource="#Height"/>
</rdf:Property>
```

```
<rdfs:Class rdf:ID="Height">
  <daml:oneOf rdf:parseType="daml:collection">
    <Height rdf:ID="short"/>
    <Height rdf:ID="medium"/>
    <Height rdf:ID="tall"/>
  </daml:oneOf>
</rdfs:Class>
```

61

DAML: hasValue and intersectionOf

```
<rdfs:Class rdf:ID="TallThing">
  <daml:sameClassAs>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasHeight"/>
      <daml:hasValue rdf:resource="#tall"/>
    </daml:Restriction>
  </daml:sameClassAs>
</rdfs:Class>
<rdfs:Class rdf:ID="TallMan">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <rdfs:Class rdf:about="#TallThing"/>
    <rdfs:Class rdf:about="#Man"/>
  </daml:intersectionOf>
</rdfs:Class>
```

62

DAML: instances

```
<Person rdf:ID="Adam">
  <rdfs:label>Adam</rdfs:label>
  <rdfs:comment>Adam is a person.</rdfs:comment>
  <hasHeight rdf:resource=#medium/>
</Person>
```

63

OWL: The three sublanguages

- *OWL Lite* supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1.
- *OWL DL* supports those users who want the maximum expressiveness while retaining computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class).
- *OWL Full* is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right.

OWL: Changes from DAML+OIL

- With respect to the three sublanguages, the DAML+OIL semantics is closest to the OWL DL semantics.
- The namespace was changed to <http://www.w3.org/2002/07/owl>
- Cyclic subclasses are now allowed
- multiple `rdfs:domain` and `rdfs:range` properties are handled as intersection
- Various properties and classes were renamed, e.g., `daml:UniqueProperty` is replaced by `owl:FunctionalProperty`
- ... <http://www.w3.org/TR/owl-ref/>

Beyond OWL: Ontology Rule Language (ORL)

- Decidability vs Expressiveness
- OWL is weak in express composite properties
- ORL extends OWL DL with a form of rules while maintaining compatibility with OWLs existing syntax and semantics.
- I. Horrocks and P. F. Patel-Schneider, A Proposal for an OWL Rules Language, ACM WWW'04, NY, May 2004

66

ORL Example

```
<owlx:Rule>
  <owlx:antecedent>
    <owlx:individualPropertyAtom owlx:property="hasParent">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x2" />
    </owlx:individualPropertyAtom>
    <owlx:individualPropertyAtom owlx:property="hasBrother">
      <owlx:Variable owlx:name="x2" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:antecedent>
  <owlx:consequent>
    <owlx:individualPropertyAtom owlx:property="hasUncle">
      <owlx:Variable owlx:name="x1" />
      <owlx:Variable owlx:name="x3" />
    </owlx:individualPropertyAtom>
  </owlx:consequent>
</owlx:Rule>
```

67

Other Rule Languages at WWW'04

Rule system examples presented at WWW'04 conference in May 2004:

- cwm rules and SweetRules (MIT/W3C)
- Jena2 rules (HP)
- CommonRule (IBM)
- ROWL (CMU)

68

Recall Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- ✓ **Semantic Web Environment for Software Specifications**
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - Semantics of DAML+OIL in Z/Alloy
 - Combined Approach to Reasoning about Semantic Web
- Conclusion and Further Work

69

Specification Languages and Their Integrations

- Many formal specification techniques exist for modeling different aspects of software systems, however,
- it is difficult to find a single notation that can model all functionalities of a complex system.
- E.g., B/VDM/Z are designed for modeling system data/states, while CSP/ π -calculus are designed for modeling system behaviour/interactions.
- Various formal notations are often extended and combined for modeling large and complex systems. In recent years, *integrated formal method* (IFM) has been a popular research topic, i.e. IFM'99 (York), IFM'00 (Dagstuhl), IFM'02 (Turku), IFM'04 (Kent).
- Due to different motivations, there are possible different semantic links between two formalisms, which can lead to different integrations between the two.

70

Integrated Formal Methods

- Unlike UML, an industrial effort for standardising diagrammatic notations, a single dominating integrated formal method may not exist in the near future. The reason may be partially due to the fact that
 - there are many different well established individual schools,
 - the open nature of the research community, i.e. FME, which is different from the industrial 'globalisation' community, i.e. OMG.
- Regardless of whether there will be or there should be an ultimate integrated formal method (like UML), *diversity* seems to be the current reality for formal methods and their integrations. Such a diversity may have an advantage, that is, different formal methods and their combinations may be suitable for different kinds of complex systems modeling.
- Challenge: to develop environment/tools for extending and combining various formal specification techniques — [Semantic Web](#)

71

Semantic Web environment for Z

```

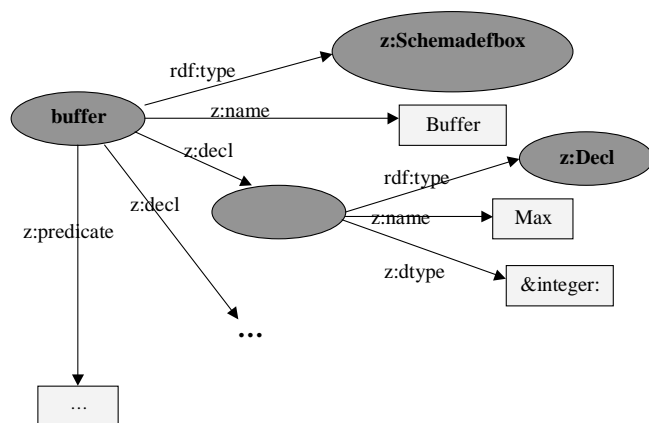
<rdf:RDF
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml = "http://www.daml.org/2001/03/daml+oil#"
  xmlns:z = "http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#"
  <!-- ... -->
  <rdfs:Class rdf:ID="Schemadef">
    <rdfs:label>Schemadef</rdfs:label> </rdfs:Class>
  <rdfs:Class rdf:ID="Schemadefbox">
    <rdfs:label>Schemadefbox</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Schemadef"/>
    <rdfs:subClassOf>
      <daml:Restriction daml:cardinalityQ="1">
        <daml:onProperty rdf:resource="#name"/></daml:Restriction></rdfs:subClassOf>
    <rdfs:subClassOf>
      <daml:Restriction daml:minCardinality="0">
        <daml:onProperty rdf:resource="#delta"/>
        <daml:toClass rdf:resource="#Schemadef"/></daml:Restriction></rdfs:subClassOf>
  <!-- ... -->

```

72

Example: Semantic Web environment for Z

<i>Buffer</i> _____ <i>Max</i> : \mathbb{Z} <i>items</i> : seq <i>MSG</i> <hr/> $\#items \leq Max$



Z Semantic Web environments can be easily extended for Object-Z. Similarly, other formalisms, i.e. CSP and TCSP, can be also constructed. Linking different formalisms is an interesting issue.

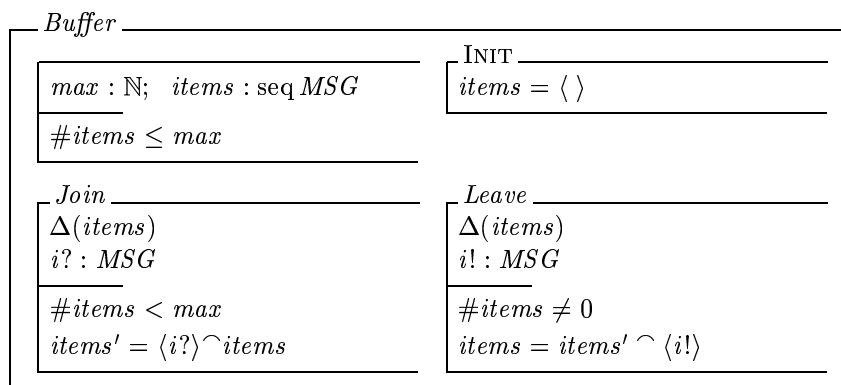
73

Semantics Links

- Various modeling methods can be used in an effective combination for designing complex systems if the semantic links between those methods can be clearly established and defined.
 - e.g. ‘mapping sets of Z operations into CSP actions’ A. Hall [FME’02]
- Given two sets of formalisms, say state-based ones and event-based ones, it’s not too surprising to see that different possible integrations are more than the cross-product of the two sets. This is simply because the different semantic links between the two formalisms lead to different integrations.
- Furthermore, the semantic links can be directional and bi-directional.

74

Object-Z $\hat{\ } \text{CSP} : \text{class} \implies \text{process}$ (Smith and Derrick, FME’97)



$Buffer_1 \hat{=} Buffer[Transfer/Leave]$
 $Buffer_2 \hat{=} Buffer[Transfer/Join]$
 $TwoLinkedBuffers \hat{=} Buffer_1 \parallel [Transfer] \parallel Buffer_2$

75

Object-Z $\hat{\ } \text{CSP} : \text{operation} \iff \text{process}$ (Mahony and Dong, ICSE'98)

```
TBuffer
Buffer
-----
left, right : chan [input and output channels]
MAIN  $\hat{=} \mu Q \bullet ([i : \text{MSG}] \bullet \text{left}?i \rightarrow \text{Join} \square$ 
       $[\#\text{items} \neq 0] \bullet \text{right}!\text{last}(\text{items}) \rightarrow \text{Leave}); Q$ 
```

Two Communicating Buffers

```
TwoLinkedBuffersa
-----
l : TBuffer[middle/right]
r : TBuffer[middle/left]
MAIN  $\hat{=} l \parallel \text{middle} \parallel r$ 
```

76

Semantic Web for linking Object-Z and CSP

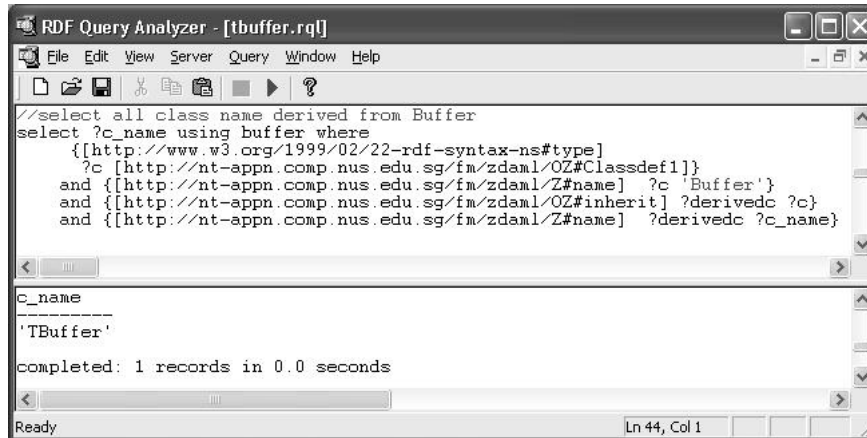
class \implies *process*

```
<daml:Ontology rdf:about="">
  <daml:imports rdf:resource="http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ"/>
  <daml:imports rdf:resource="http://nt-appn.comp.nus.edu.sg/fm/zdaml/CSP"/>
</daml:Ontology>
<rdfs:Class rdf:about="oz:Classdef">
  <rdfs:subClassOf rdf:resource="csp:Pro"/> </rdfs:Class>
```

operation \iff *process*

```
<daml:ObjectProperty rdf:ID="MAIN">
  <rdfs:range rdf:resource="csp:Process"/>
  <rdfs:domain rdf:resource="#Classdef"/> </daml:ObjectProperty>
<rdfs:Class rdf:about="oz:OP">
  <daml:sameClassAs rdf:resource="csp:Process"/> </rdfs:Class>
```

77



The screenshot shows a window titled "RDF Query Analyzer - [tbuffer.rq]". The window contains a menu bar (File, Edit, View, Server, Query, Window, Help) and a toolbar with icons for file operations and execution. The main text area contains the following SPARQL query:

```
//select all class name derived from Buffer
select ?c_name using buffer where
{[http://www.w3.org/1999/02/22-rdf-syntax-ns#type]
 ?c [http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#Classdef1]}
and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name] ?c 'Buffer'}
and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/OZ#inherit] ?derivedc ?c}
and {[http://nt-appn.comp.nus.edu.sg/fm/zdaml/Z#name] ?derivedc ?c_name}
```

Below the query, the results are displayed in a table with one row:

c_name
'TBuffer'

At the bottom of the results area, it says "completed: 1 records in 0.0 seconds". The status bar at the bottom of the window shows "Ready" and "Ln 44, Col 1".

Find all the sub-classes of the *Buffer*

Specification Comprehension

78

Recall Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- Semantic Web Environment for Software Specifications
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- ✓ Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - Semantics of DAML+OIL in Z/Alloy
 - Combined Approach to Reasoning about Semantic Web
- Conclusion and Further Work

79

Problems in designing Semantic Web ontology/services

- Semantic Web languages are not expressive enough for designing Semantic Web complex ontology properties and service/agents.

Require a systematic design process with expressive high level modeling techniques

Solution: software specifications

80

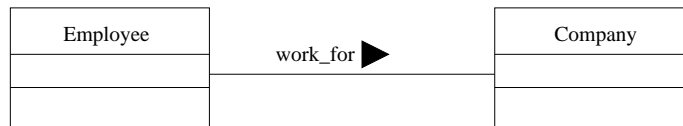
Some DAML constructs in Abstract Form

Abstract DAML constructs	Description
<i>daml_class</i>	classes
<i>daml_subclass</i> [<i>C</i>]	subclasses of <i>C</i>
<i>daml_objectproperty</i> [<i>D</i> ↔ <i>R</i>]	relation properties with domain <i>D</i> , range <i>R</i>
<i>daml_objectproperty</i> [<i>D</i> → <i>R</i>]	function properties with domain <i>D</i> , range <i>R</i>
<i>daml_subproperty</i> [<i>P</i>]	sub properties of <i>P</i>
<i>instanceof</i> [<i>C</i>]	instances of the DAML class <i>C</i>

81

Extracting DAML ontology from UML Models

There are a few approaches (e.g. [2, 10]) to transform UML class models to DAML ontology. One particular approach [1] is to extend UML meta model to directly include the notion of 'Property' and 'Restriction'. For example, a property restriction 'work_for' can be represented as:



82

Extracting DAML ontology from the Z model

Z can be used to model web-based ontology at various levels. The Z conceptual domain models can be transformed to DAML+OIL ontology via XSLT technology.

Given type transformation

$$\frac{[T]}{T \in \text{daml_class}}$$

e.g.

[*Author*]

```
<daml:class rdf:ID="author">
  <rdfs:label>Author</rdfs:label> </daml:Class>
```

83

Z schema transformation

$$\frac{S}{X : T_1; Y : \mathbb{P} T_2} \quad T_1, T_2 \in \text{daml_class}$$

$S \in \text{daml_class}, X \in \text{daml_objectproperty}[S \rightarrow T_1], Y \in \text{daml_objectproperty}[S \leftrightarrow T_2]$

$$\frac{\text{Paper}}{\text{title} : \text{Title}; \text{authors} : \mathbb{P} \text{Author}}$$

```
<daml:class rdf:ID="paper"> <rdfs:label>Paper</rdfs:label> </daml:Class>
<daml:ObjectProperty rdf:ID="paper_title"> <rdf:type rdf:resource="
  http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#paper"/>
  <rdf:range rdf:resource="#title"/> </daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="paper_authors">
  <rdf:domain rdf:resource="#paper"/>
  <rdf:range rdf:resource="#author"/> </daml:ObjectProperty>
```

84

Z axiomatic definition transformation (relation/functions)

$$\frac{R : B \leftrightarrow (\rightarrow, \Rightarrow) C}{\dots} \quad B, C \in \text{daml_class}$$

$R \in \text{daml_objectproperty}[B \leftrightarrow (\rightarrow, \Rightarrow) C]$

```
| reference : Paper ↔ Paper      <daml:ObjectProperty rdf:ID="paper_reference">
  <rdfs:domain rdf:resource="#paper"/>
  <rdfs:range rdf:resource="#paper"/>
  </daml:ObjectProperty>
```

85

Z axiomatic definition transformation (subset)

$$\left| \begin{array}{l} M : \mathbb{P} N \\ \dots \end{array} \right. \quad N \in \text{daml_class}$$

$M \in \text{daml_subclass}[N]$

$$\left| \begin{array}{l} \text{Biannual} : \mathbb{P} \text{ConfSeries} \\ \dots \end{array} \right. \quad \begin{array}{l} <\text{daml:class rdf:ID="biannual"}> \\ <\text{rdfs:subClassOf rdf:resource="\#confseries"}> \\ </\text{daml:class}> \end{array}$$

Exercise: Convert Z spec to DAML

$$\left[\text{Students}, \text{Code}, \text{Title} \right] \quad \left[\begin{array}{l} \text{Course} \text{ —————} \\ \text{code} : \text{Code} \\ \text{title} : \text{Title} \end{array} \right. \quad \left| \begin{array}{l} \text{GraduateCourse} : \mathbb{P} \text{Course} \\ \text{enrolment} : \text{Students} \leftrightarrow \text{Course} \end{array} \right.$$

Convert the Z spec to DAML:

```

<daml:class rdf:ID="student"> <rdfs:label>Student</rdfs:label> </daml:Class>
<daml:class rdf:ID="code"> <rdfs:label>Code</rdfs:label> </daml:Class>
<daml:class rdf:ID="title"> <rdfs:label>Title</rdfs:label> </daml:Class>

<daml:class rdf:ID="course"> <rdfs:label>Course</rdfs:label> </daml:Class>
<daml:ObjectProperty rdf:ID="course_code">
<rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#course"/> <rdf:range rdf:resource="#code"/>
</daml:ObjectProperty>
<daml:ObjectProperty rdf:ID="course_title">
<rdf:type rdf:resource="http://www.daml.org/2001/03/daml+oil#UniqueProperty"/>
  <rdf:domain rdf:resource="#course"/> <rdf:range rdf:resource="#coursetitle"/>
</daml:ObjectProperty>

<daml:class rdf:ID="graduatecourse">
  <rdfs:subClassOf rdf:resource="#course"/> </daml:class>
<daml:ObjectProperty rdf:ID="enrolment">
  <rdfs:domain rdf:resource="#student"/> <rdfs:range rdf:resource="#course"/>
</daml:ObjectProperty>

```

Improve the ontology quality through Z tools

Z/EVES tool is an interactive system for composing, checking, and analyzing Z specifications. It supports the analysis of Z specifications in several ways: syntax and type checking, schema expansion, precondition calculation, domain checking, and general theorem proving. Some ontology related flaws in Z model can be detected and removed with the assistance of Z/EVES so that the transformed DAML ontology from checked Z model will have better quality.

Alternatively, one can develop reverse transformation tools from DAML ontology to the formal specifications then to use formal specification tools to detect domain and logical errors that the current DAML reasoner is not able to detect.

Checking Military Plan Ontology Experience

- Singapore DSO has developed an IE engine which has been used to generate ontologies (in DAML) from military formation and plan (in natural language).
- A military ontology is made up of the following four main ingredient sets.
 - military operations and tasks, which define the logic order, type , and phases of a military campaign.
 - military units, which are the participants of the military operations and tasks,
 - geographic locations, where such operations take place and
 - time points for constraining the timing of such operations.
- We have developed an auto transformation tool that takes DAML document and produces Z specifications, then we use Z/EVES tool to check the type errors and ontology consistency issues.
- Checking beyond web ontology (e.g. one military unit assigned two different tasks at the same time period)

90

A Real Military Case Study Statistics in Z/EVES

Items	Numbers
Resources	138
Operations, tasks, phases	56
Units	47
Geographic areas	35
Statements (in RDF)	592
Transformed Axiomatic Defs (in Z)	138
Transformed Predicates (in Z)	410
Type errors	22
DAML related ontology errors	0
errors beyond DAML	2

91

Recall Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- Semantic Web Environment for Software Specifications
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - ✓ [Semantics of DAML+OIL in Z/Alloy](#)
 - Combined Approach to Reasoning about Semantic Web
- Conclusion and Further Work

92

Ontology Tools: A Brief Survey

- RDF reasoner: Cwm, Triple
- **F**ast **C**lassification of **T**erminologies (FaCT)
 - Supports consistency & subsumption reasoning (TBox)
 - Does not support instantiation reasoning (ABox)
- **R**enamed **A**Box and **C**oncept **E**xpression **R**easoner (RACER)
 - Supports TBox & ABox reasoning
 - Includes richer functionalities compared to FaCT
- FaCT & RACER are fully automated
- OilEd: graphical ontology editor that supports FaCT & RACER

93

Z/Alloy Semantics for DAML+OIL

Basic Concepts

- Resource

$[Resource]$ `sig Resource {}`

- Class & instances

$\left| \begin{array}{l} Class : \mathbb{P} Resource \\ instances : \\ Class \rightarrow \mathbb{P} Resource \end{array} \right.$ `disj sig Class extends Resource`
`{instances: set Resource}`

- Property & sub_val

$\left| \begin{array}{l} Property : \mathbb{P} Resource \\ Class \cap Property = \emptyset \end{array} \right.$ `disj sig Property extends Resource`
`{sub_val: Resource -> Resource}`

$\left| \begin{array}{l} sub_val : Property \\ \rightarrow (Resource \leftrightarrow Resource) \end{array} \right.$

94

Z/Alloy Semantics for DAML+OIL

Class Relationships

- subClassOf & disjointWith

$\left| \begin{array}{l} subClassOf : Class \leftrightarrow Class \\ disjointWith : Class \leftrightarrow Class \end{array} \right.$

$\forall c_1, c_2 : Class \bullet$
 $c_1 \underline{subClassOf} c_2 \Leftrightarrow instances(c_1) \in \mathbb{P} instances(c_2)$
 $c_1 \underline{disjointWith} c_2 \Leftrightarrow instances(c_1) \cap instances(c_2) = \emptyset$

```
fun subClassOf(c1, c2: Class)
  {c2.instances in c1.instances}
fun disjointWith (c1, c2: Class)
  {no c1.instances & c2.instances}
```

95

Z/Alloy Semantics for DAML+OIL

Class & Property

- toClass

$$\frac{}{toClass : (Class \times Property) \leftrightarrow Class}$$
$$\frac{\forall c_1, c_2 : Class; p : Property \bullet (c_1, p) \underline{toClass} c_2 \Leftrightarrow (\forall a_1, a_2 : Resource \bullet a_1 \in instances(c_1) \Leftrightarrow ((a_1, a_2) \in sub_val(p) \Rightarrow a_2 \in instances(c_2)))}{}$$

```
fun toClass (p:Property, c1:Class, c2:Class)
{all a1, a2: Resource | a1 in c1.instances <=>
  a2 in a1.(p.sub_val) => a2 in c2.instances}
```

- Example: Anything that breathes
by gill is a fish, including all
those don't breathe at all!

$$\frac{Fish, Gill : Class \quad Breathe_by : Property}{(Fish, Breathe_by) \underline{toClass} Gill}$$

96

Z/Alloy Semantics for DAML+OIL

Class & Property (continued)

- hasValue

$$\frac{}{hasValue : (Class \times Property) \leftrightarrow Resource}$$
$$\frac{\forall c : Class; p : Property; r : Resource \bullet (c, p) \underline{hasValue} r \Leftrightarrow (\forall a : instances(c) \bullet (a, r) \in sub_val(p))}{}$$

```
fun hasValue (p:Property, c:Class, r:Resource)
{all a:Resource |
  a in c.instances => a.(p.sub_val) = r}
```

97

Z/Alloy Semantics for DAML+OIL

Property Relationships

- subPropertyOf

$$\frac{\text{subPropertyOf} : \text{Property} \leftrightarrow \text{Property}}{\forall p_1, p_2 : \text{Property} \bullet \frac{p_1 \text{ subPropertyOf } p_2 \Leftrightarrow \text{sub_val}(p_1) \in \mathbb{P} \text{ sub_val}(p_2)}}{}$$

```
fun subPropertyOf (p1, p2:Property)
  {p1.sub_val in p2.sub_val}
```

98

Import Mechanism & Proof Support for Z/EVES

- Import mechanism
 - Z definitions are put into a *section* `dam12z`
 - Alloy definitions are put into a *module* `DAML`
 - Other transformed ontologies have these definitions as parents
- Proof support for Z/EVES
 - Definitions alone are not adequate
 - Trivial proof goals should be automated
 - A *section* `DAML2ZRules` of *rewrite*, *assumption* & *forward* rules are constructed

99

Military Plan Ontology

- Developed by DSO Singapore, defining concepts in military domain:
`military.daml`
- Instance ontologies generated from plain text by IE engine
- Contains sets of
 - Military operations & tasks
 - Military units
 - Geographic locations
 - Time points

100

Transformation

- DAML+OIL to Z
 - Developed a Java tool for automatic transformation
 - Supports both plan & instance ontologies
 - A number of enhancements made
 - * Z predicates marked by *labels* as (rewrite or assumption) rules
 - * Time points modeled as natural numbers \mathbb{N}
 - * Domain-specific theorems are added
 - * Supports Unique Name Assumption
 - * Additional predicates added to facilitate proof
- DAML+OIL to Alloy
 - More straightforward
 - Using an XSLT stylesheet

101

Transformation: Example

- DAML+OIL:

```
<daml:Class rdf:about="http://www.dso.org.sg/  
  PlanOntology#MilitaryTask">  
  <rdfs:label>MilitaryTask</rdfs:label>  
  <rdfs:subClassOf>  
    <daml:Class rdf:about="http://www.dso.org.sg/  
      PlanOntology#MilitaryProcess"/>  
  </rdfs:subClassOf>  
</daml:Class>
```

- Z:

```
    MilitaryTask : Class  
    ────────────  
    ⟨⟨grule MilitaryTask_subClassOf_MilitaryProcess⟩⟩  
    (MilitaryTask, MilitaryProcess) ∈ subClassOf
```

- Alloy:

```
static disj sig MilitaryTask extends Class {}  
fact{subClass(MilitaryProcess, MilitaryTask)}
```

102

Recall Overview

- Introduction to Software Modeling Techniques
 - UML, Z, Alloy and CSP
- Introduction to Semantic Web
 - RDF, DAML+OIL, OWL and ORL
- Semantic Web Environment for Software Specifications
 - Linking Different Specification Languages through Semantic Web
 - Specification Comprehension via RDF Query
- Software Design Method/Tools for Semantic Web
 - Extracting DAML ontology from UML/Z models
 - Semantics of DAML+OIL in Z/Alloy
 - ✓ **The Combined Approach**
- Conclusion and Further Work

103

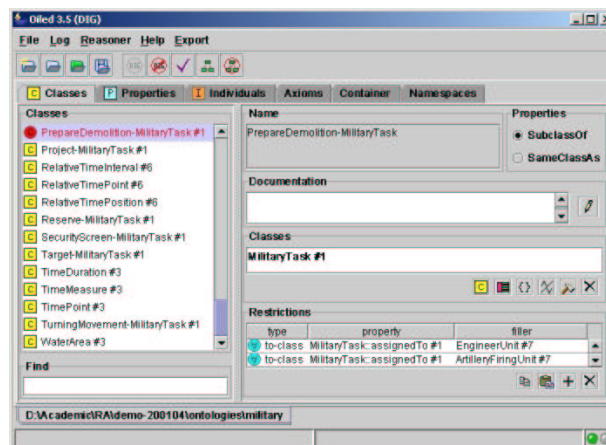
The Combined Approach

1. Transforms ontology to Z & type-check using Z/EVES
 - Semi-automated
2. Use RACER & OilEd to check for ontological inconsistencies
3. If inconsistencies found, use AA to pinpoint them
 - Iterate steps 2 & 3 until RACER finds no inconsistency
4. If an instance ontology, use Z/EVES to check for properties inexpressible in DAML+OIL & Alloy
 - Interactive...

104

Standard SW Reasoning

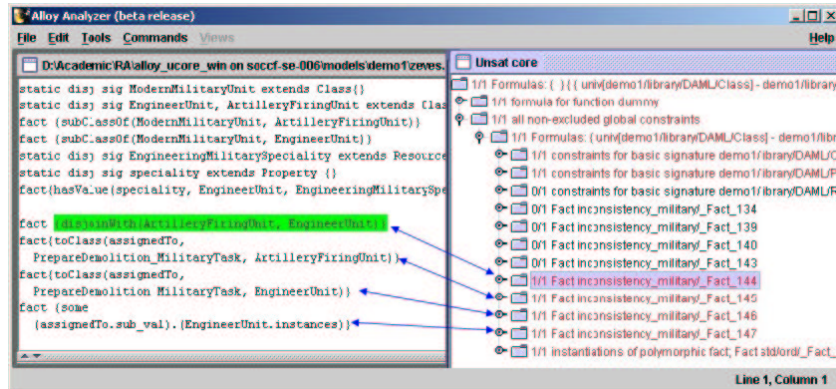
- Step 1: Z/EVES finds no type errors in (transformed) `military.daml`
- Step 2: RACER complains about an inconsistent class, `PrepareDemolition-MilitaryTask` on the left



105

Standard SW Reasoning (continued)

- However, RACER cannot tell where the inconsistency is
- Step 3: Extract fragment of ontology according to OilEd
- AA finds the inconsistency, and it gives the possible cause in red color



106

More Advanced Reasoning

- Applied to instance ontology planA.daml: 954 RDF statements, 195 subjects
- Ontology fragment:

```
<rdf:Description rdf:about='G. SMILAX'>
  <rdf:type rdf:resource='http://www.dso.org.sg/PlanOntology#AxisOfAdvance' />
</rdf:Description>
<rdf:Description rdf:about='InfantryBattalion_aa5'>
  <rdf:type rdf:resource='http://www.dso.org.sg/PlanOntology#InfantryBattalion' />
</rdf:Description>
```

$G_SMILAX : Resource$	$InfantryBattalion_aa5 : Resource$
$\langle\langle grule G_SMILAX_type \rangle\rangle$	$\langle\langle grule InfantryBattalion_aa5_type \rangle\rangle$
$G_SMILAX \in instances(AxisOfAdvance)$	$InfantryBattalion_aa5 \in instances(InfantryBattalion)$

- 28 type errors discovered by Z/EVES: mostly caused by re-definition
- No ontological errors found by RACER

107

More Advanced Reasoning (continued)

- Use domain-specific theorems to systematically test the consistency of the ontology
- E.g., “no military task should be the sub task of itself and its start time should be less than or equal to its end time”.
- Once a goal cannot be proved: negate the theorem and prove
- 14 *hidden errors* found by Z/EVES in step 4
 - 2: military task’s start time greater than end time
 - 4: military task doesn’t have end time defined
 - 3: military unit assigned to different tasks simultaneously
 - 5: military tasks with more than one start or end time point

108

Local Consistency

- “No military task should be the sub task of itself and its start time should be less than or equal to its end time” – local consistency of military tasks

theorem MilitaryTaskTimeSubTaskTest1

$\forall x : instances(MilitaryTask) \bullet$
 $start(x) < end(x) \wedge x \notin (sub_val(subTaskOf))(\{x\})$

- Systematically test all instances of **MilitaryTask**
- Example: the remaining goal of one inconsistent example: **ECA_P3_P3_S1**

$\neg x = ECA_P3_P3_S1$

- Apparent contradiction: negate the theorem & prove again

theorem negatedMilitaryTaskTimeSubTaskTest1

$\exists x : instances(MilitaryTask) \bullet$
 $\neg (start(x) < end(x) \wedge x \notin (sub_val(subTaskOf))(\{x\}))$

109

Temporal Relationships Between Tasks

- “Sub tasks’s duration must be within its super tasks’ durations”

theorem subTaskOfTimingTest2

$\forall x : instances(MilitaryTask) \bullet$

$\forall y : \mathbb{P}(instances(MilitaryTask)) \mid$

$y = (sub_val(subTaskOf))(\{x\}) \bullet$

$\forall z : y \bullet start(z) \leq start(x) \wedge end(z) \geq end(x)$

- y is the set of super tasks of x , z is any member of y
- Local consistency ensured by the previous theorem, hence $start(z) \leq start(x) \wedge end(z) \geq end(x)$ is sufficient

Military tasks & units

- “No military task is to be assigned to 2 different tasks at the same time”

theorem MilitaryUnitTest

$\forall x : instances(ModernMilitaryUnit) \bullet \forall y, z : instances(MilitaryTask) \mid$

$x \in (sub_val(assignedTo))(\{y\}) \wedge x \in (sub_val(assignedTo))(\{z\}) \bullet$

$end(y) \leq start(z) \vee end(z) \leq start(y)$

- Since local consistency has been ensured for each military task, predicate $end(y) \leq start(z) \vee end(z) \leq start(y)$ is sufficient
- Example: the remaining goal for military tasks ECA_P3_P5_S1 & ECA_P3_P5_S3 and military unit CHF_1

$z = ECA_P3_P5_S1 \wedge y = ECA_P3_P5_S3$

$\Rightarrow \neg x = CHF_1$

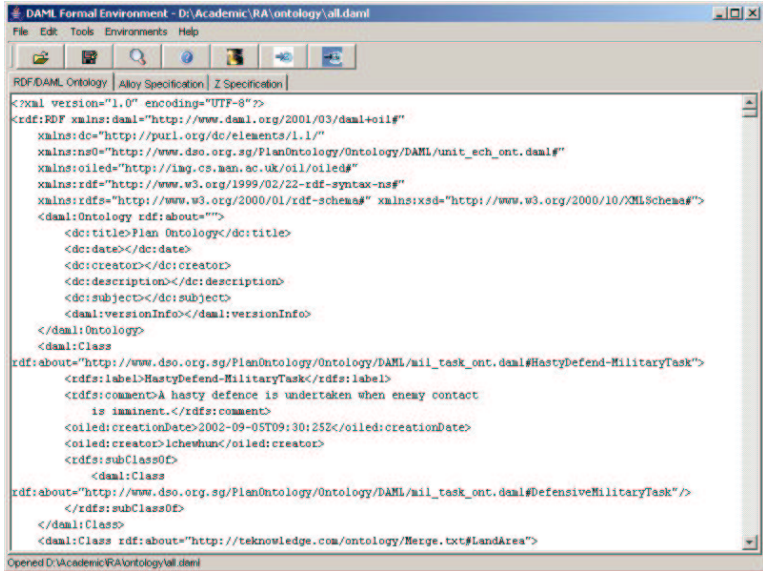
- An obvious contradiction, negate the theorem & prove again
- 3 such errors were found

Summary of the Combined Approach

- The combination of SW & SE reasoning tools effectively checks ontology-related properties
- Results of the synergy
 - Automatically find ontological inconsistencies using RACER
 - Isolate & find the source of the inconsistencies using Alloy Analyser
 - Interactively checks for more complex properties (inexpressible in DAML+OIL) using Z/EVES
- Application to the second military-domain case study revealed 1 ontological inconsistency & 14 *hidden* errors

112

Tool Environment for the Combined Approach (on going)



```
D:\Academic\RA\ontology\oild.daml
File Edit Tools Environments Help
RDF/DAML Ontology Alloy Specification Z Specification
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:daml="http://www.daml.org/2001/03/daml-oil#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:ns0="http://www.dso.org.sg/PlanOntology/Ontology/DAML/unit_sch_ont.daml#"
  xmlns:oiled="http://img.cs.man.ac.uk/oil/oiled#"
  xmlns:rdfs="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" xmlns:xsd="http://www.w3.org/2000/10/XMLSchema#"
  <daml:Ontology rdf:about="">
    <dc:title>Plan Ontology</dc:title>
    <dc:date></dc:date>
    <dc:creator></dc:creator>
    <dc:description></dc:description>
    <dc:subject></dc:subject>
    <daml:versionInfo></daml:versionInfo>
  </daml:Ontology>
  <daml:Class
    rdf:about="http://www.dso.org.sg/PlanOntology/Ontology/DAML/mil_task_ont.daml#HastyDefend-MilitaryTask">
    <rdfs:label>HastyDefend-MilitaryTask</rdfs:label>
    <rdfs:comment>A hasty defence is undertaken when enemy contact
      is imminent.</rdfs:comment>
    <oiled:creationDate>2002-09-05T09:30:25Z</oiled:creationDate>
    <oiled:creator>lcheshum</oiled:creator>
    <rdfs:subClassOf>
      <daml:Class
        rdf:about="http://www.dso.org.sg/PlanOntology/Ontology/DAML/mil_task_ont.daml#DefensiveMilitaryTask">
        </rdfs:subClassOf>
      </daml:Class>
    </rdfs:subClassOf>
    <daml:Class rdf:about="http://teknowledge.com/ontology/Herge.txt#LandArea">
```

113

Tutorial Conclusion

- Semantic Web
 - ✓ good support for automation, collaboration, extension and integration
 - × less expressive and no systematic design process for web ontology/agents
- Software Specifications
 - ✓ expressive, diverse and can be combined effectively
 - × weak in linking various methods for collaborative design
- Approaches
 - ✓ Semantic Web environment for linking various formalisms (FME'02)
 - ✓ Extracting web ontologies systematically from Z specifications (ICFEM'02)
 - ✓ Checking Semantic Web Using Software Tools (FME'03, ICSE'04, WWW'04)

Possible Future Research

- Software Engineering for Semantic Web:
 - Software specification languages (like Z) as Semantic Web languages
 - Web Services (OWL-S) Specifications
 - Model behaviors of intelligent Semantic Web agents using Z, process algebra or integrated formal methods
- Semantic Web for Software Engineering
 - Meta integrating environment for software modeling
 - Intelligent Software Engineering Environment

Recent Publications

The research on Formal methods and Semantic Web has been investigated in [8, 9, 7, 12, 6, 4, 5]. The research on UML and Semantic Web has been investigated in [3, 11, 1, 2, 10].

References

- [1] K. Baclawski, M. Kokar, P. Kogut, L. Hart, J. Smith, W. Holmes, J. Letkowski, and M. Aronson. Extending UML to Support Ontology Engineering for the Semantic Web. In M. Gogolla and C. Kobryn, editors, *UML'01*, Lect. Notes in Comput. Sci. Springer-Verlag, 2001.
- [2] S. Craneffeld. Networked knowledge representation and exchange using uml and rdf. *Journal of Digital Information*, 1(8), 2001.
- [3] S. Craneffeld and M. Purvis. Uml as an ontology modeling language. In *IJCAI Workshop of Intelligent Information Integration*, 1999.
- [4] J. S. Dong, C. H. Lee, H. B. Lee, Y. F. Li, and H. Wang. A Combined Approach to Checking Web Ontology. In *The 13th International World Wide Web Conference (WWW'04), refereed papers track*. ACM Press, May 2004.
- [5] J. S. Dong, C. H. Lee, Y. F. Li, and H. Wang. Verifying DAML+OIL and Beyond in Z/EVES. In *The 26th International Conference on Software Engineering (ICSE'04)*. IEEE Press, May 2004.

- [6] J. S. Dong, J. Sun, and H. Wang. Semantic Web for Extending and Linking Formalisms. In L.-H. Eriksson and P. A. Lindsay, editors, *Proceedings of Formal Methods Europe: FME'02*, pages 587–606, Copenhagen, Denmark, July 2002. LNCS, Springer-Verlag.
- [7] J. S. Dong, J. Sun, and H. Wang. Z Approach to Semantic Web. In C. George and H. Miao, editors, *International Conference on Formal Engineering Methods (ICFEM'02)*, pages 156–167. LNCS, Springer-Verlag, October 2002.
- [8] J. S. Dong, J. Sun, and H. Wang. Checking and Reasoning about Semantic Web through Alloy. In *Proceedings of 12th International Symposium on Formal Methods Europe: FM'03*, pages 796–813, Pisa, Italy, September 2003. LNCS, Springer-Verlag.
- [9] J. S. Dong, J. Sun, H. Wang, C. H. Lee, and H. B. Lee. Analysing Semantic Web: A Military Case Study. In *The 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'03)*, San Francisco, USA, June 2003.
- [10] K. Falkovych, M. Sabou, and H. Stuckenschmidt. Uml for the semantic web: Transformation-based approaches, 2003. in B. Omelayenko and M. Klein: *Knowledge Transformation for the Semantic Web*, IOS Press.
- [11] P. Kogut, S. Craneffeld, L. Hart, M. Dutra, K. Baclawski, M. Kokar, and J. Smith. UML for Ontology Development. *Knowledge Engineering Review*, 17, 2002.
- [12] Hai Wang. *Semantic Web and Formal Design Methods*. PhD thesis, National University of Singapore, 2004. (to appear).