

14th European Summer School in
Logic, Language and Information
5-16 August 2002
Trento, Italy



ATTENDANCE REPORT
from Anna Trifonova

Organized by:



In collaboration with:



Sponsored by:



COURSE: **Annotation of Language Resources**

Tomaž Erjavec
Anne-Marie Mineur

The first lecture introduced the eXtended Markup Language (XML), the motivation for its development, its history and building blocks - elements, attributes and entities, and how they fit together.

XML is a “metalanguage” - a language for describing other languages - which lets you design your own customized markup languages for different types of documents. XML is a project of the World Wide Web Consortium (W3C), and the development of the specification is being supervised by their XML Working Group; hence, it is an open and non-proprietary specification. XML is designed to improve the functionality of the Web by providing more flexible and adaptable information identification. XML is not just useful for Web pages, but also for uses traditionally given to SGML - managing large amounts of valuable, (predominately) textual data; ensuring longevity of the texts; enabling interchange of data between computer platforms; allowing multiple exploitation of texts.

Some history (the evolution of the standard) of SGML (Standard Generalized Markup Language) was given.

The XML elements and the content model were discussed and presented as an example. The validity of an XML document could be checked by a validating parser. The formal statement of the document type grammar may be provided by a Document Type Definition (DTD) or by an XML schema. The DTD elements and the content model were also discussed.

The second lecture discussed developments related to XML, in particular XML Schemas, XML Namespaces, XPath, and the XML transformation language, XSLT. Links to the specifications and examples were given.

XML Namespaces – document constructs should have universal names, whose scope extends beyond their containing document so that elements and attributes could be used by multiple software modules.

XML Schemas – XML Schemas were introduced to adjust some problems with DTDs (DTDs can impose only weak constraints on attribute and element content; DTDs themselves are not written in XML, so tools to process (edit, validate, present) XML do not work with them). Several proposals were introduced - Academia Sinica's Schematron; OASIS / ISO DIS RELAX NG; W3C XML Schema.

XPath – defines a mechanism for locating information in XML documents, and has many other uses besides that in formatting documents. XPath operates on the abstract, logical structure of an XML document (its InfoSet), rather than its surface syntax; it models an XML document as a tree of nodes. There are different types of nodes, including element nodes, attribute nodes and text nodes.

XSLT is a W3C Recommendation. It defines a means of transforming XML documents into other data formats (the most popular use is to convert XML into HTML).

In the third lecture some XML-related software was listed, i.e. parsers, transformation engines and XML modules to use with general purpose programming languages.

SP - the best known suite of basic SGML tools; written by James Clark (available from <http://jclark.com/sp/>). SP consists of a validating parser; a converter from SGML to XML; a markup normaliser; a markup stream editor; an interface to SP entity manager.

Other software written by James Clark was also mentioned:

- jade - implementation of style language part of DSSSL. Extensively used for formatting DocBook documents

- expat - the "XML Parser Toolkit", a library for XML parsing in C. This parser is used to add XML support to Netscape 5 and Perl

- XP - a high-performance XML parser in Java

- XT - a Java implementation of XSLT

Sub-projects of the Apache XML Project that focus on different aspects of XML were also mentioned:

- Xerces - XML parsers in Java and C++ (with Perl and COM bindings)

- Xalan - XSLT stylesheet processors, in Java and C++

- Cocoon - XML-based web publishing, in Java

- AxKit - XML-based web publishing, in mod_perl

- FOP - XSL formatting objects, in Java

- Xang - Rapid development of dynamic server pages, in JavaScript

- SOAP - Simple Object Access Protocol

- Batik - A Java based toolkit for Scalable Vector Graphics (SVG)

- Crimson - A Java XML parser derived from the Sun Project X Parser

A section was devoted to editors, in particular Emacs. Other XML editors were also mentioned – jEdit, XXE, oXygen, XMLwriter, XMLpro, XMLspy.

Some linguistic annotation software was also mentioned, in particular several corpus workbenches and statistic tool packages.

The next lecture presented the XML-based Text Encoding Initiative Guidelines and other language encoding recommendations. TEI can be used to annotate a wide variety of language resources. TEI is used by different libraries and text archives do describe and search through the resources. The lecturer presented the history, organization and architecture of TEI and illustrated it with applications to multilingual corpora, lexical databases and feature structures.

There was a list of other encoding recommendations - first some language engineering standards that came about as a result of EU projects, i.e. EAGLES/ISLE with (X)CES and then a few lexicon exchange initiatives, i.e. MARTIF, TMX and OLIF.

The last lecture was presented by Anne-Marie Mineur. It was dedicated on metadata. She gave a short list of metadata proposals and projects: DCMI: Dublin Core Metadata Initiative (1995); OAI: Open Archives Initiative (1999); OLAC: Open Linguistic Archives Community (2001); LTRC: Language Typology Resource Center (2002); TDP: Typological Database System (in progress).

She listed the element set and also the element attributes of DCMI, OLAC, etc.

COURSE: **Programming Logics for Object-oriented Languages**

F. de Boer and C. Pierik

The first lecture was an introduction of the basic concepts of object-oriented programming. Few Object-oriented programming languages were mentioned: Small-Talk, C++, and Java. Some differences between them (like multiple inheritance in C++ which does not exist in Java) were mentioned.

Classes, instances, attributes and methods were exemplified. A class consists of class name (to refer to), attributes (to store data) and methods (operations that can be applied to objects). A class is a definition, whereas an object is a declaration of an instance of a class type with an identity.

The data encapsulation principle was discussed. The data is encapsulated in every object so it can be accessed only through the methods, defined in the class.

Examples of basic data types (INT, BOOLEAN, REAL, STRING), complex data types (Arrays) and typed variables (var x : integer) were given.

Special expressions were introduced – this (object identity), nil (undefined value), etc. Basic statements (assignments, object creation, and method calls) were shown and also statements (like sequential composition $S_1; S_2$; deterministic choice: if b then S_1 else S_2 fi; if b then S fi; iteration: while b do S od; etc.). With basic statements more complex statements can be built with blocks.

Part of the lecture was devoted to method and class definitions:

$m(u_1, \dots, u_n): S; e$ – where m is the method name, u_1, \dots, u_n are the formal parameters, S is the body of the method and e is the result expression (optional).

Class definition = set of method definitions + set of instance variables (attributes)

Threads were discussed. The program is logically divided into different tasks which can run separately and if the machine has more than one processor the program runs faster. The main program creates objects and applies a method to a known object. Then the control is passed to the object. The object may create other objects and execute methods on them (and also pass the control). In order to “know” other objects created by the root the object has to receive it as parameter.

Inheritance and methods overriding were also discussed. In Java a class can extend only one class (no multiple inheritance). If a class C extends a class D then C is a subtype of D .

A lecture was devoted to formal semantic that the lecturer used to describe objects, variables, contexts and etc. for the rest of the course. Expressions were evaluated in proper context (local configuration; global configuration).

Some basic knowledge and examples of logics were given: boolean logical expressions; boolean connectives ($\neg l$; $l_1 \wedge l_2$); quantification $\exists z P$; and etc.

One lecture was an introduction of the state-of-the-art languages for modeling and specification.

UML : Visual Modeling Language – the first complete version (Booch, Rumbaugh, Jacobson) was from 1997. UML is a visual modeling language for specifying, visualizing, constructing, document object-oriented systems

OCL : UML Specification Language - Object Constraint Language is a subset of UML (a standard part of the Unified Modeling Language). Version 1.0 is introduced in September 1997. The last stable is version 1.4 (Jan 2001). OCL is an assertion language for describing and reasoning about navigation in object structures.

JML : Java Modeling Language. JML is developed by G.T. Leavens, A.L. Baker and C. Ruby (Iowa State University). Its current version is Version 3.0 (May 2002) and the tools for JML are open source. It is a behavioral interface specification language for Java. It allows assertions like invariants, constraints, pre- and post-conditions, and modifiable clauses as annotations to Java classes, in a design-by-contract style. These constraints are particularly useful, as they allow a developer to create a highly specific set of rules that governs the aspect of an individual object.

? All these languages were presented to show the work done in the field of computer-aided specification and verification.

One lecture was a presentation and demonstration of a tool, developed by C.Pierik, for verification of OO-programs. The tool is implemented in Java. The aim of the tool is to support the specification and verification of a class of flowcharts that captures the basic dynamics of object-oriented programs. It is based on an implementation of a weakest precondition calculus for reasoning about semantics of basic assertions in object-oriented programs. As an input this verification tool takes flowcharts (formalism for modeling programs), the flowchart specification, a class library, and a library of macro definitions. It contains lexical analyzers and parsers for annotated flowcharts, macros and class descriptions. The tool is a theorem prover which is used to check the verification conditions validity. The abstraction level of the assertion language they define corresponds to OCL, but there are some differences.

The tool is still under development. As a future work the author has an intention to implement message passing, basics of multi-threaded control flow of Java; and the Java mechanism of inheritance.

There was an introduction of Hoare logic - a logic for proving properties of programs. It was developed by C. A. R. Hoare in the late 1960s.

It shows how programs can be given an axiomatic basis. Programs in procedural language acquire a semantics by pre- and postconditions expressed in first-order predicate logic. These pre- and postconditions can be considered as specifications of the programs. Hoare defines correctness formulas:

$$\{P\} S \{Q\}$$

where S is a program in a procedural language and the precondition P and a postcondition Q are logical formulas describing properties of data manipulated by the program S . The correctness formula means that if execution of S is started in a state satisfying P and if the execution terminates then Q will be true. Often the postcondition Q involves both initial and final states.

Defining the semantics of individual statements of a procedural programming language by correctness formulas and providing correct rules for the composition of statements to programs enables us to construct a verified program from its specifications.