LX — Layered Architectures for XML-based Data Interchange

Part 1: Previous Research and Track Record

Introduction

There is evidently a significant convergence between two important technologies: markup for the World Wide Web (XML, the Extensible Markup Language (see `http://www.w3.org/TR/REC-xml`). The Internet is just beginning to make an impact on the database world, and the fact that data as much as if not more so than documents will use XML to travel the Internet is just beginning to significantly influence the design and development of the XML family of standards.

Data means not just data in standard database formats (i.e. accessible via SQL) but also data in record structures and object hierarchies in application programs, each with idiosyncratic dump formats for archiving and interchange. The value of an XML representation for data of both kinds, both internally (for archiving and interchange) and externally (if you can show the world your data in the form of XML, they'll buy it) is becoming apparent in many commercial sectors (e-commerce not the least). The problem the work proposed here seeks to address is that a myriad of more or less ad-hoc mechanisms are springing up to manage the translation from internal data representations to XML and back again. We propose to use emerging international standards, in particular the W3C's XSLT and XML Schema (Clark 1999, Thompson *et al.* 2000), to provide a uniform, standards-based solution. (Strictly speaking the W3C (The World Wide Web Consortium, a membership organisation responsible for managing the higher levels of Internet use) does not promulgate standards, but rather Recommendations, because it is not an official international body, but its recommendations are the closest to standards there are for the areas they cover.)

The proposed work starts with the most tractable aspects of this problem to prototype a solution architecture, and progresses to address a number of more complex aspects, bringing to bear a number of strands of existing research and development work.

Previous Research

The Language Technology Group and the proposer have been involved with relevant research into markup, markup architectures and markup application architectures for over 10 years.

We published the first freely available research corpus annotated using the TEI Guidelines for SGML markup of such material (Burnard and Sperberg-McQueen 1990, Bard *et al.* 1992), developed a simplified version of SGML for use in language processing and distributed a free API and toolkit based on this (LTNSL 1996). Normalised SGML, as we called this simplification, was an important input into the design process for XML itself, which the proposer participated in as one of the original members of what was initially called the SGML Working Group of the W3C in 1997.

The LTG migrated its normalised SGML tools to XML and started developing a suite of XML-aware tools for use in data-intensive applications (for an overview, see McKelvie *et al.* 1998). The core of these tools is LT XML (Thompson *et al.* 1998), an integrated set of XML tools and a developers' tool-kit, including a C-based API, running on UNIX and WIN32 platforms. LT XML has now been licenced to over 4500 individual worldwide in academic and industrial environ-

ments. The core validating XML parser at the heart of that system, called RXP (Tobin 1999), is widely acknowledged as one of the two best high-performance validating parsers available.

The proposer has been involved with the core XML technologies of the proposed work, XSLT and XML Schema (see below) from the very start, having collaborated in the initial research and design work (Adler *et al.* 1997) which led to the formation of the XSL Working Group which wrote the XSLT Recommendation, and similarly published an early exposition of the need for what became XML Schema (Thompson 1998) which included a number of key design ideas now incorporated in the XML Schema draft recommendation (Thompson *et al.* 2000) of which he is a co-editor.

Support for our work on XML and related markup technologies has come from EPSRC (project NSCOPE, GR/L29125, with the proposer as Principal Investigator and the named RA as major contributor), the European Union (Project MATE, on a Multilevel Annotation Tool) and the ESRC (the HCRC Core Grant, 1989-1999). Our work in this area has also attracted industrial support, with direct grants from Sun Microsystems and Microsoft.

Part 2: Proposed Research and its Context

1. Technical Background

Layering and the Cambridge Communiqué

A preliminary outline of the place of XML in a world wide web of data is documented in the Cambridge Communiqué (Swick & Thompson 1999). We observe that XML initially shared a design philosophy with SGML, focussed on a simple data model for a tree-structured view of documents and a flexible linearisation or transfer syntax for documents represented using that model. The majority of energy going in to XML has, however, been sustained by the empirical observation that tree-structured documents are a pretty good transfer syntax for just about anything, and that by converting from application data models to the document data model, the utility and ubiquity of XML become available for moving application data around and making it available outside its original producer/consumer community. The Cambridge Communiqué says it this way:

1. XML has defined a transfer syntax for tree-structured documents;
2. Many data-oriented applications are being defined which build their own data structures on top of an XML document layer, effectively using XML documents as a transfer mechanism for structured data.

It goes on to recommend that XML Schema make provision for facilitating this layered approach.

The layered approach can be seen as complementing a more constrained approach to data interchange based on abstract APIs and Remote Procedural Call, e.g. COM, CORBA, IDL.

XML Schema

XML Schema is a W3C initiative to provide a replacement for the existing document structure definition mechanism of XML, known as the Document Type Definition (or DTD), using XML markup itself instead of the idiosyncratic DTD notation. One of its principal official requirements was the provision of state-of-the-art mechanisms for definition structuring, composition and reuse, and these mechanisms in turn are intended to be modelled on, and a good impedance match with, existing structure definition systems from other branches of computer science, including programming languages and database systems.

As the above discussion of the Cambridge Communiqué suggested, the design of XML Schema makes explicit provision for annotation of the constituent parts from which a schema, that is a formal definition notated in XML of the structure of a family of XML documents, is composed.

XSLT

XSLT (Extensible Stylesheet Language: Transformations) is a W3C Recommendation, one of two components of a mechanism for associating information about intended appearance with XML documents. The 'Transformation' part of XSLT is what is of interest here: it provides a very powerful functional language, using pattern-directed invocation, to specify a mapping from one family of structured documents into another.

Although XSLT's primary purpose if for transforming from XML to XML or HTML, it also makes provision for transforming XML into arbitrary application-internal structures, via an extension mechanism.

Semi-structured Data, Data Guides and Schema Derivation

Semi-structured data is the name given to a relatively recent attempt to bridge the gap between relational and object-oriented approaches to database design. It has been suggested that it might also help bridge the gap between databases and XML (Abiteboul *et al.* 1999). In the first instance it is a way of abstracting away from implementation detail in the presentation of database contents, but in at least some cases it supports step-wise refinement via Data Guides ([LORE ??]), which can be seen as a summary of the access paths available with respect to a particular collection of semi-structured data to a class-based schema which

is much closer to an Entity-Relation [Chen ???] data model, related via simulation to the original dataset, but also standing as a hypothesis about its underlying structure.

2. Programme and Methodology

2.1 Overall goals

Although the overall shape of the desired architecture for data publishing and interchange via XML is clear, and many more or less *ad hoc* efforts are already under way to instantiate it for particular application/programming language pairs (see e.g. Reinhold (1999), Box (2000)), what is really wanted is support for a declarative specification of the relation between an application data model and an XML Schema, each independently defined. In concrete terms this support should yield implementations of language-independent marshalling and unmarshalling, that is, bi-directional conversion between XML instance and application data. Some aspects of a solution are already clear in outline – others will require exploration of possibilities for application of research results from other related disciplines.

We see the proposed research as necessary preparation for standardisation work in this area: Member companies of the W3C have recently requested that it undertake work on standardising XML protocols (Larry Masinter, personal communication), while at the same time clarifying that XML-encoded RPC is *not* what is required: such a move would leave the XML-structure to/from application structure correspondence issue to be solved.

The following questions each need to be addressed to arrive at the desired architecture:

- Is the mapping to be specified by annotations within an individual XML Schema document, e.g. by adding mapping information to each element and attribute declaration? Alternatively, should the mapping be specified externally, possibly exploiting XSLT?
- The directionality aspect of the mechanism deserves special consideration: Both XML Schema and XSLT are by construction good matches for the XML → application (unmarshalling) direction. What kind of conditions must be imposed on either type of solution to guarantee reversibility, that is, the application → XML (marshalling) direction? Again, XML Schema and XSLT both imply a control structure from which an implementation of unmarshalling naturally emerges. What control structure would be required for marshalling?
- What are the tradeoffs between specifying the application side of the mapping in implementation-level terms (e.g. Java class instance/variable or relational table/row) versus specifying it in more abstract terms (e.g. Entity-Relation, EXPRESS (ref??) or UML [UML ???]?
- Would specifying an abstract mapping in the schema, and concrete language-specific bindings from abstract model to implementation independently of the schema, give the right modularity?
- When only one or the other model is specified in advance (i.e. XML Schema or application data model), can we automatically derive the other? If so, what conventions should be used in doing so?
- Does the intermediate position between XML and traditional databases occupied by semi-structured data offer any leverage for the solution to this problem?
- What constraints, if any, are required to allow an implementation of unmarshalling to work in a streaming fashion, i.e. to build application structures as an XML document is processed, without building a complete internal representation of the document before application structure construction can begin?

The work proposed here aims at answering these questions, structuring the effort in terms of three broad goals:

- Design a declarative approach based as far as possible on existing public standards which supports automatic generation of implementation-language-appropriate unmarshalling of XML documents into application data structures;

- Extend the above to support marshalling in a congruent way, i.e. the construction of XML documents from application data structures;
- Explore approaches to (semi-)automatically generating appropriately annotated XML schemas from schemas expressed in one or more data modelling languages.

2.2 Deliverables and Work Packages

Work Package 1:Identification and Description of Test Cases

We will need a number of independently designed document types, application data models, and pairs of the two to use in the other work of the project. Examples will be sought from existing document-oriented DTDs, such as the XHTML DTD (XHTML 2000), from existing data-oriented DTDs, such as the OAG e-commerce transaction DTDs (OAG ??) in addition to the schema, DTD and data model for XML Schema itself, which we have already worked with in the unmarshalling direction. Starting from the application data end, possible examples include the CEN??? patient record model (in UML, (CEN? ??), the Dublin Core bibliographic metadata model (in RDF) and the teachers-courses-students database model (Entity-Relation) developed jointly with Microsoft (A1 ??). The deliverables will include not only an inventory of designs, but at least some example documents in each case, as well as schemas or partial schemas, which will have to be created in many cases.

Work Package 2: Schema Annotation

XML Schema provides for arbitrary annotations to be added to the declarations and definitions (called schema components) which together define a document type, that is, a set of documents with a common structure and common purpose. The deliverable from this work package is a design for a set of annotations which specify the correspondence between schema components and application data model components. This breaks down into two tasks:
a) design the syntax of the annotation mechanism, that is, whether to use elements or attributes, which aspect of schema extensibility to exploit, etc.;
b) determine what vocabularies are needed to identify the different kinds of application model components, e.g. `entity`, `relation`, `attribute` etc. for Entity-Relation, or `class instance`, `instance variable`, `list` for an O-O programming language.

Work Package 3: Language-specific Unmarshalling: Direct prototype

This workpackage is the proof of concept for the architecture we have in mind, developing a prototype for the simplest part of the overall design, and testing it on real examples.

Task 3.1: Design and Construction

The deliverable of this workpackage is an implementation of a compiler from XML Schemas containing language-specific annotations as designed in WP 2 above into an XSLT stylesheet for unmarshalling, using the XSLT element extension mechanism as implemented in our existing streaming XSLT processor. Three tasks here:
a) design one set of language-specific extension elements;
b) Implement the schema → stylesheet compiler, specialised to one language-specific vocabulary and targeting those extension elements;
c) implement the extension classes in the context of an XSLT implementation. The choice of implementation will determine the choice of language: C if we use our own streaming XSLT processor, Java if we use XT, James Clark's public domain XSLT processor.

Task 3.2: Deployment and Testing

This implementation gives us the basis for actually annotating schemas and unmarshalling from real document instances to test the viability of our approach, and explore its appropriateness to the three broad use-cases as outlined above in Workpackage 1. The deliverables will be annotated versions of schemas for a number of the test cases delivered by WP 1 above, along with improved versions of the deliverables from 3.1

Work Package 4: From Low- to High-level Annotation

To achieve our goal of appropriate modularisation, we need to elaborate the first deliverable from workpackage 3 above (the schema-annotation to stylesheet compiler) so that instead of implementation-language-dependent annotation vocabularies which can be com-

piled directly to application-structure-building stylesheets, we can annotate a schema in a domain-appropriate high-level modelling language such as Entity-Relation, EXPRESS or UML, and then parameterise the compilation process by a further specification of the correspondence between terms in that high-level model and implementation terms in a particular operating environment. The deliverables are thus parallel to those of Tasks 3.1 and 3.2 above, but based on high-level annotation and parameterised compilation.

Work Package 5: Marshalling

All the above has focussed on unmarshalling from XML documents into application structures. This workpackage turns to the question of marshalling application structures into XML documents. There are two different approaches to be explored here. The first is based on compiling the annotations of workpackage 3 into e.g. methods for the classes involved to achieve marshalling functionality. We are not aware of any language-independent precedent for this, in cases where the document structure was designed prior to or independently from the application data model.

Ambiguity and overloading are the obvious stumbling blocks. The alternative, although more speculative, also offers more if it is successful. There is some literature in areas of formal language theory applied to layout (e.g. Feng 1993) and applied to translation (e.g. Yellin 1988) which explores the annotation of one of a pair of grammars, either context-free or finite-state, with information about its relation to the other. The work may offer help in at least identifying properties of document schemas (which are isomorphic to context-free grammars) and application data models (which are relatable to grammars in most cases) which would be necessary to allow the automatic creation of marshalling understood as a similar sort of transformation as in the unmarshalling case.

Work Package 6: Second-order compilation

When an application data model exists, but no document definition work has been done to support XML receipt or delivery, automatic generation of an XML schema complete with annotation would fit well into the overall picture.

Task 6.1 Design Exploration In principle it would be best to start at the modelling language level, then map from UML or EXPRESS or RDF schema to 1) an XML Schema; 2) the necessary schema annotations. The major stumbling block that we envisage is when there are re-entrancies or circularities in the application model: when to choose foreign keys/ID-IDREF/linking vs. embedding. The work on dataguides introduced above offers a promising place to start, going from data → semi-structured data → data guide → schema. How to generate the appropriate mapping annotations as part of this process is an open question.

Task 6.2: Implement and Evaluate

For evaluation, we can take the second-order compiler, apply it to a model for which we actually do have an existing schema, e.g. XML Schema itself, and compare results.

3. Management

The day-to-day management of the project will be the responsibility of the Principal Investigator, Henry Thompson. He has considerable experience in the management of local, national and international R&D projects.

4. Dissemination and exploitation

A version of the software produced will be documented and made available to other labs for R&D purposes on the basis of a non-exclusive licence. The Language Technology Group has considerable experience in packaging, documenting, advertising and licencing language engineering software (see `http://www.ltg.ed.ac.uk/software`).

The overall architecture and the way in which the overall goals set out above are achieved will be presented at appropriate conferences and described in scholarly papers.

should we promise another dissemination workshop?

5. Relevance to beneficiaries

Even a partially successful outcome of the proposed work would be of major benefit to commercial and non-commercial users of the World Wide Web: moving away from an *ad-hoc* scripting-based approach to the interface

between application data and XML to a declarative standards-based approach is both more robust and more cost-effective.

6. Justification of resources

Personpower.
One researcher at the level of RA1 is requested. The researcher must have a wide range of skills, ...

In addition, 25% of Richard Tobin XXX

Computing support is included in the costings to cover the skilled manpower required for the continuing provision of a smoothly running distributed computing infrastructure, including printing, mail, news, backup and archiving facilities, and for the day to day management of project specific hardware and systems software.

Equipment and computing consumables.
The work will be data-intensive, and in our experience a dedicated workstation with large-capacity storage media directly attached is best suited for this type of development work. To this end, the cost of a Sun Workstation has been added to the budget.

Consumables and annual connection charges have been included in the budget. These are for the new machine as well as for machines already in place which will also be used during the project. These costs have been calculated at standard departmental rates. They cover, *inter alia*, laser printing, computer paper, disks, cartridges, network costs, and maintenance of shared file and print servers.

Travel.
The travel money requested will cover dissemination. In particular, we intend to present results at a few major conferences such as XTech and XML Europe (annual XML conferences and **no idea – we need one CS-type conference for credibility – VLDB?**

Part 3: Diagrammatic workplan

see email

References

S. Adler, Henry Thompson and 9 others, *A Proposal for XSL*, W3C Note, W3C, Cambridge MA. Also available as `http://www.w3.org/TR/NOTE-XSL.html`.

Abiteboul, Buneman and Suciu, ????

E. Bard *et al.* (1992) The HCRC Map Task Corpus. In *Language and Speech* 34 (1992) 4, pp351-366.

James Clark, ed., *XSL Transformations (XSLT)*, W3C Recommendation, W3C, Cambridge, MA. Also available as `http://www.w3.org/TR/xslt`.

Don Box 2000

Chen ???

Entity-Relation model

An Feng, Toshiro Wakayama: "SIMON: A Grammar-based Transformation System for Structured Documents," Electronic Publishing 6(4): 361-372 (1993)

Mark Reinhold (1999): JSR-000031 XML Data Binding Specification. Available at `http://java.sun.com/aboutJava/communityprocess/jsr/jsr_031_xmld.html`

Ralph Swick and Henry Thompson eds. (1999): *The Cambridge Communiqué*. W3C, Cambridge, Ma. Also available as `http://www.w3.org/TR/schema-arch`

Henry Thompson *et al.* (2000): XML Schema Part 1: Structures, W3C, Cambridge, MA. Also available as `http://www.w3.org/TR/xmlschema-1/`

XML Query

Daniel M. Yellin (1998): Attribute Grammar Inversion and Source-To-Source Translation. Springer-Verlag, Berlin, New York, 1988.