XML in software development

Technical overview

Lars Marius Garshol,
development manager Ontopia,
larsga@ontopia.net

Who speaks?

- **Lars Marius Garshol**
  - Development manager at Ontopia, and one of the founders
  - Author of *Definitive XML Application Development*, published by Prentice-Hall
  - Wrote the xmlproc validating parser in Python
  - Responsible for translation of SAX to Python
  - Editor of parts of the topic map standard (ISO 13250-2 og 13250-3)
  - Editor of the TMQL standard (topic map query language, ISO 18048)
- **Ontopia**
  - Leading vendor of topic map software
  - “The Oracle of Topic Maps”
  - Norwegian company with partners world-wide

My personal XML history

- **Started with XML in 1997**
  - started my MSc thesis on content management just as XML work was taking off
  - followed the XML process from the start
  - believed all the promises that XML would make it possible to find information and exchange anything with anyone
- **Now I work with topic maps**
  - XML turned out not to be what I was looking for
  - many of the supporting standards I do not think good enough
  - am now a bitter and disappointed man

Overview

- **Introduction**
- **XML and application architecture**
  - impedance mismatch
  - web services
- **Common XML-related tasks**
  - XML tools and standards
- **Conclusion**

Introduction

What is XML really?
Data models
Interchange and storage

XML is a way to organize data

- **XML is one of many ways to do this**
- **XML is a data format (or syntax)**
  - used when storing XML in files
  - also used when transmitting XML
- **XML has a data model**
  - used in XML databases and query languages
  - some support for this, not main usage

Other data representations

- **Relational**
  - tabular, rows and columns
  - used by relational databases
  - primary focus on storage, limited interchange with CSV files
- **Object-oriented**
  - objects with properties and methods
  - used by most programming languages today
  - primary focus on application-internal representation
  - some interchange, also some database support
- **XML**
  - tree of labeled nodes
  - primary focus on interchange
  - some database support

So, what is XML good for?

- **Well, it was created for documents...**
  - <p>allows <term>mixed content</term>, which is unusual</p>
  - also strictly preserves order everywhere (except for attributes)
- **XML works very well for documents**
- **XML also works for data**
  - however, the document features make it more complicated than necessary
  - for storage it is not optimal
  - for interchange it is still the best alternative

Why XML is good for interchange

- **Standard is done right**
  - short, implementable, precise, formal, readable, hackable
  - everything is Unicode all the way: no internationalization problems
  - Draconian error handling forces users to do things right
  - schema languages make validation simple and effective
- **Everyone agrees on the standard**
  - Microsoft, Sun, IBM, Oracle, you-name-it
- **Lots of high-quality tools**
  - parsers tend to be fast, highly conformant, and robust
  - lots and lots of higher-level tools make life easier
  - tools available for all languages and platforms

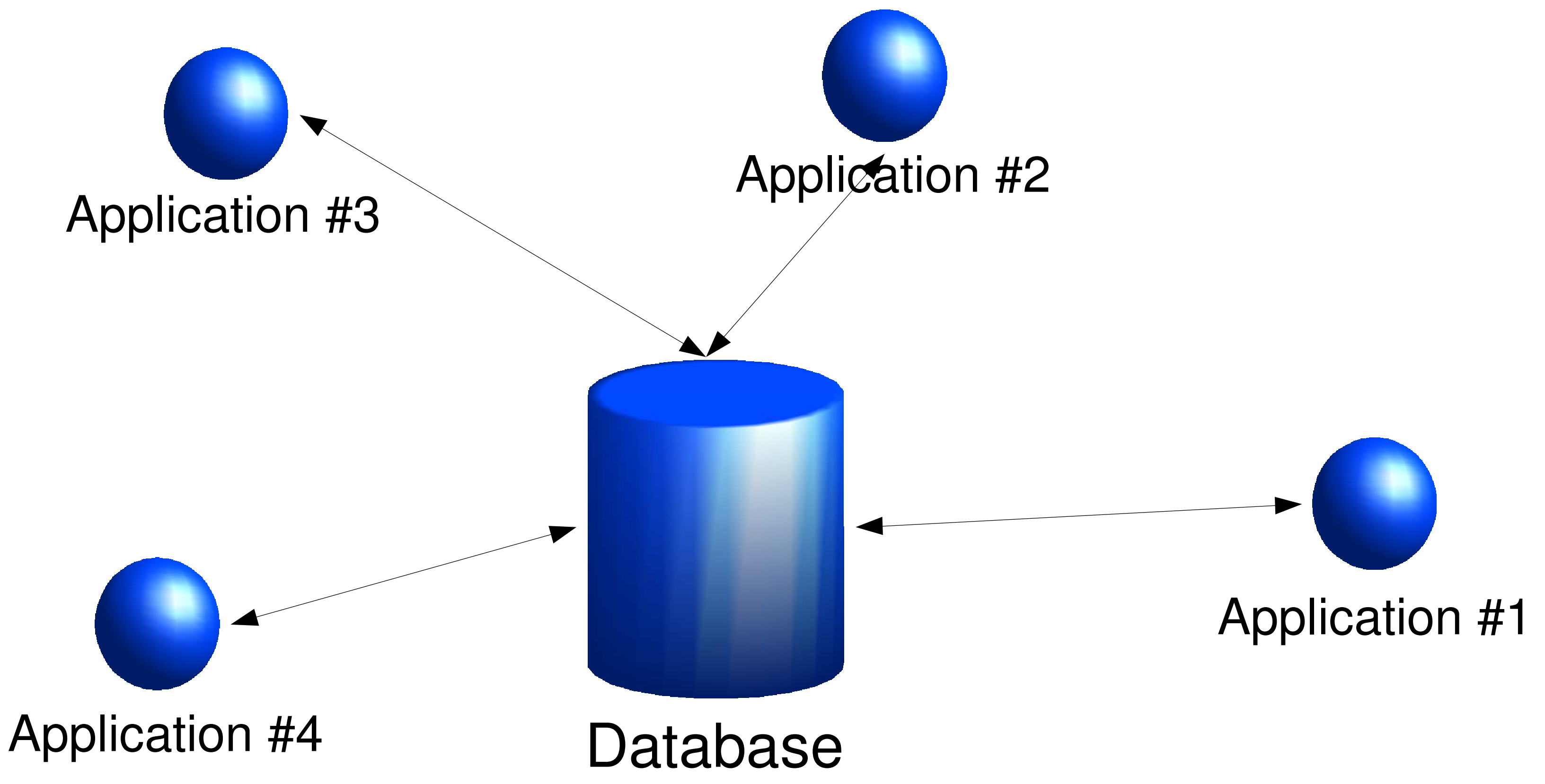
XML and architecture

Traditional information systems
The impedance mismatch
An example XML application

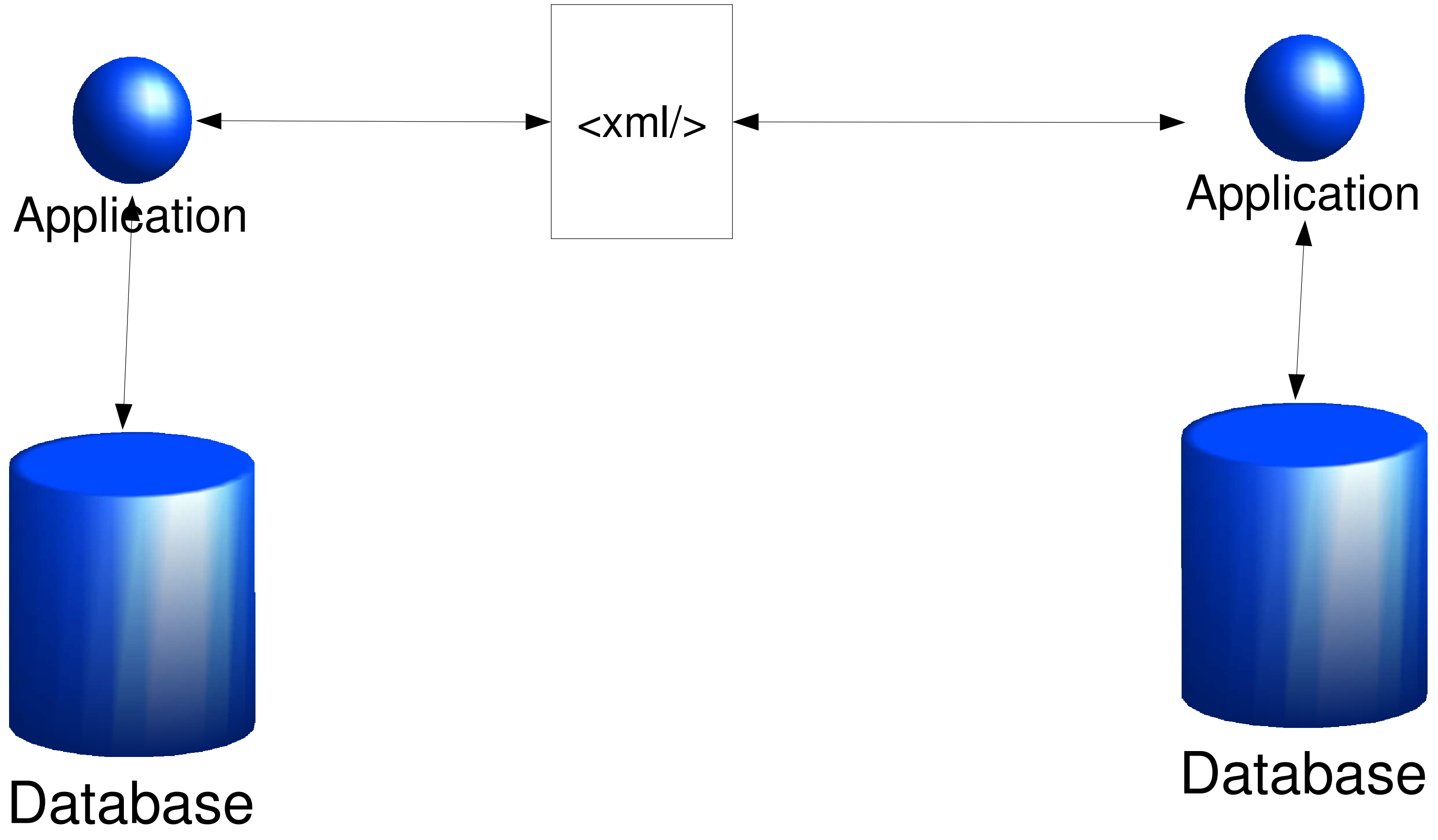
Information systems

- Information-centric computing has traditionally been about information systems
- Typically, these were clusters of applications with a database at the center
- Originally, the business logic would reside in the database
- With n-tier architecture it was encapsulated by an object layer
- The basic concept has remained the same, however

Traditional 1-tier architecture

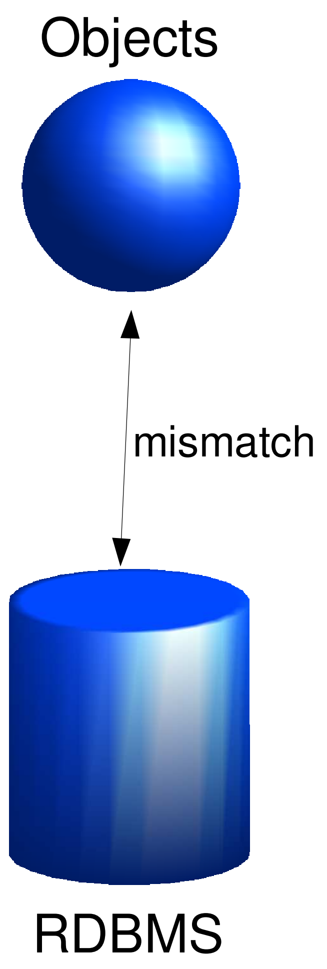


XML enters the picture



Impedance mismatch

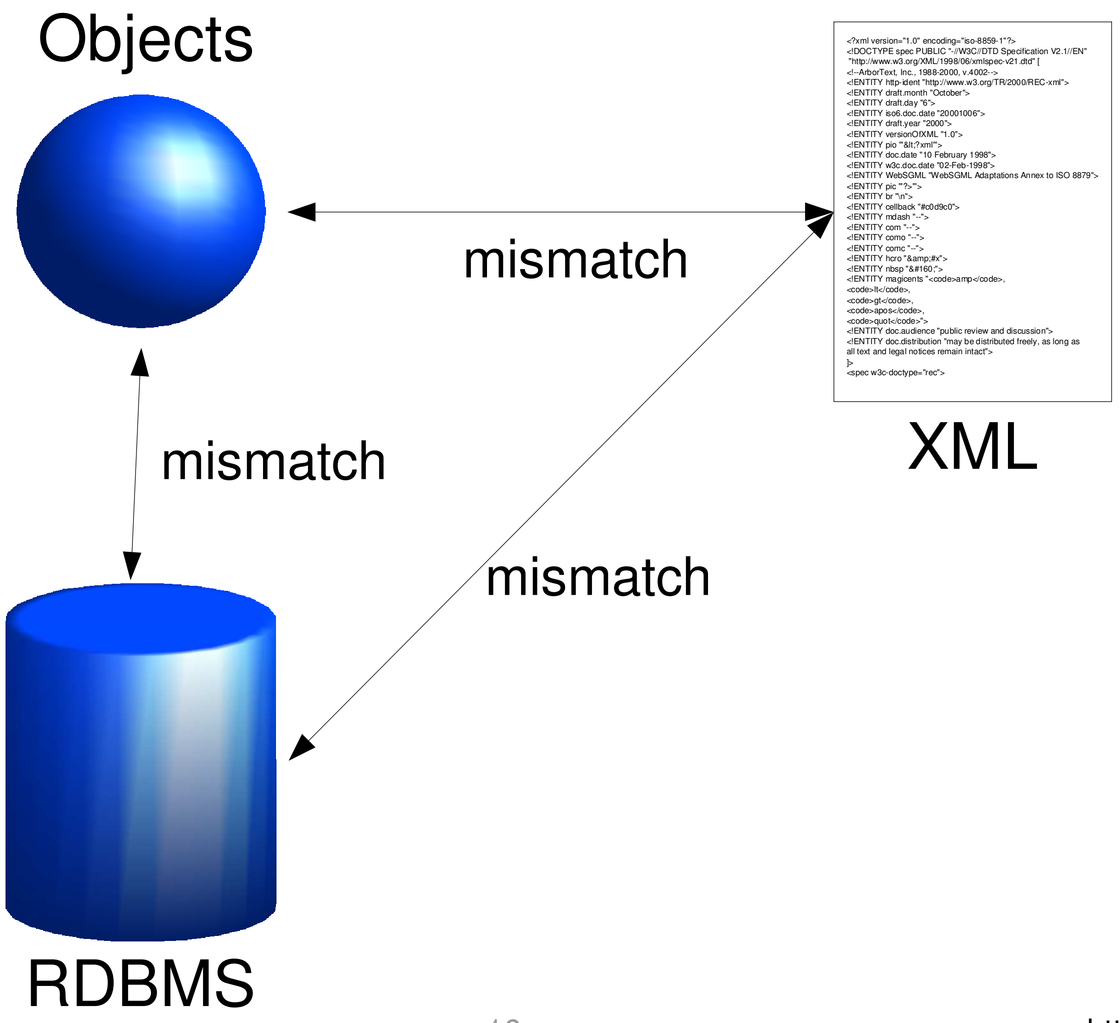
- **The OO/RDBMS impedance mismatch**
  - object-oriented languages use objects with properties
  - RDBMSs use tables
  - these two data models do not match, and mapping between them requires substantial effort
- **Common solutions**
  - attempt to isolate RDBMS interaction in an application module
  - use object-relational mapping tools
  - give up, just plunge in, and create a horrible mess
- **Conclusion**
  - the problem is real, but with effort it can be handled



The brave new world of XML

- **Originally we had the OO-RDBMS mismatch**
- **XML adds the OO-XML and XML-RDBMS mismatches**
  - in other words: yet another issue for developers to deal with
- **Solutions are much the same**
  - use data binding tools (we'll return to these)
  - restrict XML code to a specific module
  - give up and create a mess
- **Conclusion**
  - interchange is complicated, and there is no silver bullet

A very common architecture



```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE spec PUBLIC "-//W3C//DTD Specification V2.1//EN"
 "http://www.w3.org/XML/1998/06/xmlspec-v21.dtd" [
<!--ArborText, Inc., 1988-2000, v.4002-->
<!ENTITY http-ident "http://www.w3.org/TR/2000/REC-xml">
<!ENTITY draft.month "October">
<!ENTITY draft.day "6">
<!ENTITY iso6.doc.date "20001006">
<!ENTITY draft.year "2000">
<!ENTITY versionOfXML "1.0">
<!ENTITY pio "'&lt;?xml'">
<!ENTITY doc.date "10 February 1998">
<!ENTITY w3c.doc.date "02-Feb-1998">
<!ENTITY WebSGML "WebSGML Adaptations Annex to ISO 8879">
<!ENTITY pic "'?>'">
<!ENTITY br "\n">
<!ENTITY cellback "#c0d9c0">
<!ENTITY mdash "--">
<!ENTITY com "--">
<!ENTITY como "--">
<!ENTITY comc "--">
<!ENTITY hcro "&amp;#x">
<!ENTITY nbsp "&#160;">
<!ENTITY magicents "<code>amp</code>,
<code>lt</code>,
<code>gt</code>,
<code>apos</code>,
<code>quot</code>">
<!ENTITY doc.audience "public review and discussion">
<!ENTITY doc.distribution "may be distributed freely, as long as
all text and legal notices remain intact">
]>
<spec w3c-doctype="rec">
```

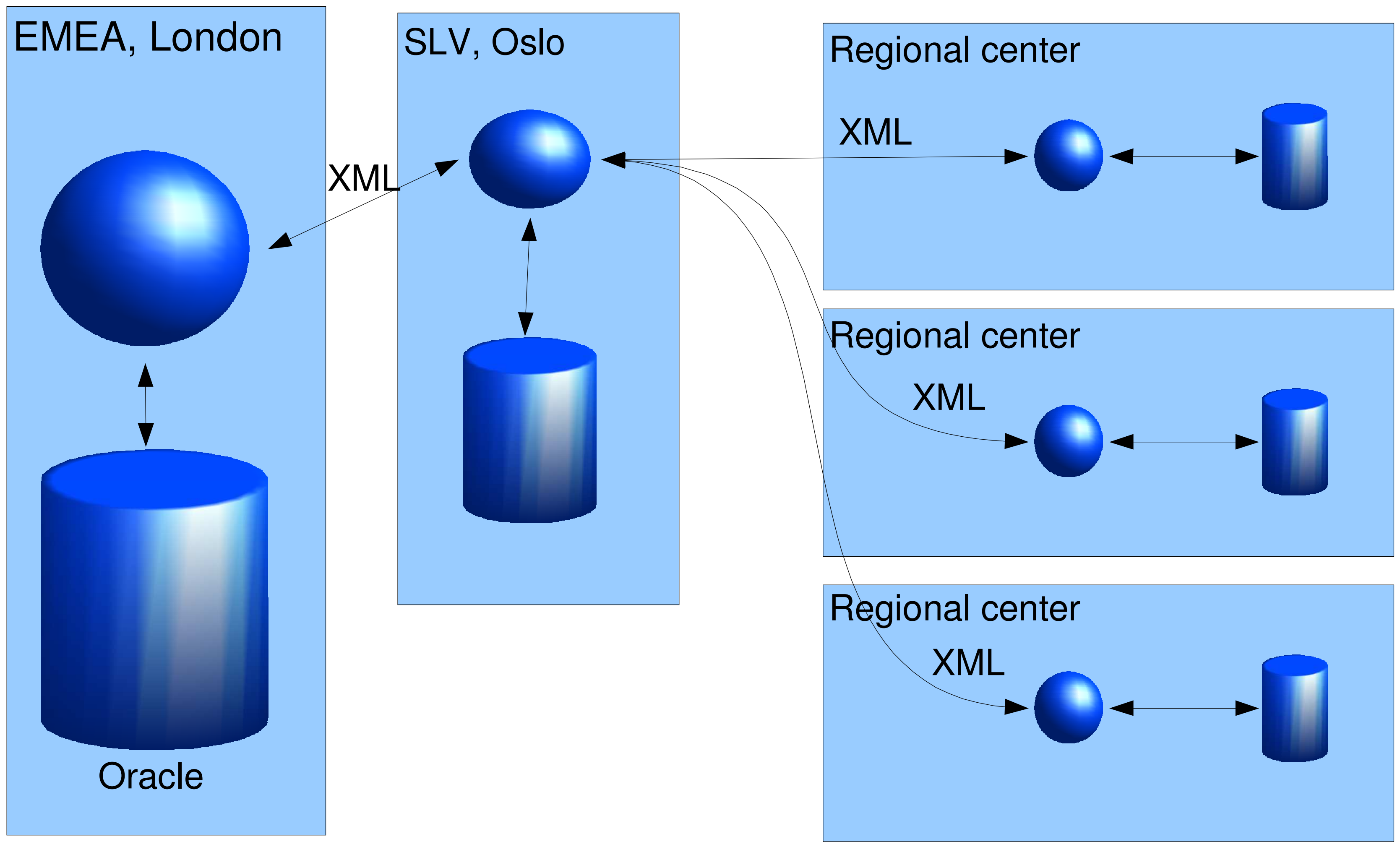
So, what to do?

- **XML is already here**
  - all the big vendors are pushing it
  - government standards and customers require it
  - the open source community has embraced it
- **In short, we just have to live with it now**

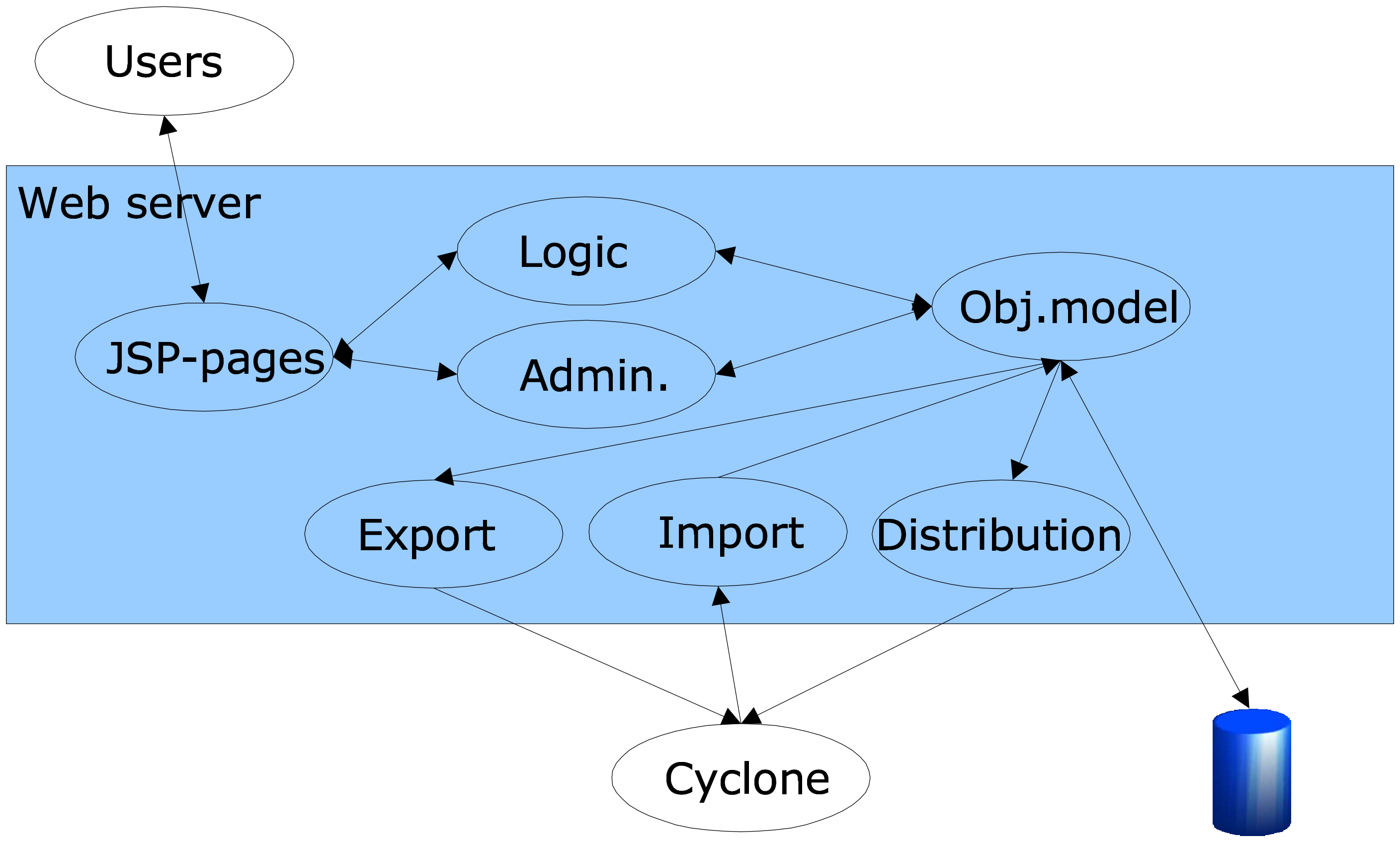
An example application

- From January 2003 the EU required all member states to submit individual case safety reports for drugs
- Basically, every time someone suffers side-effects from a drug, this is to be reported to EMEA in London
- A standardized XML format is used for this
- Ontopia developed the solution used by Norwegian authorities

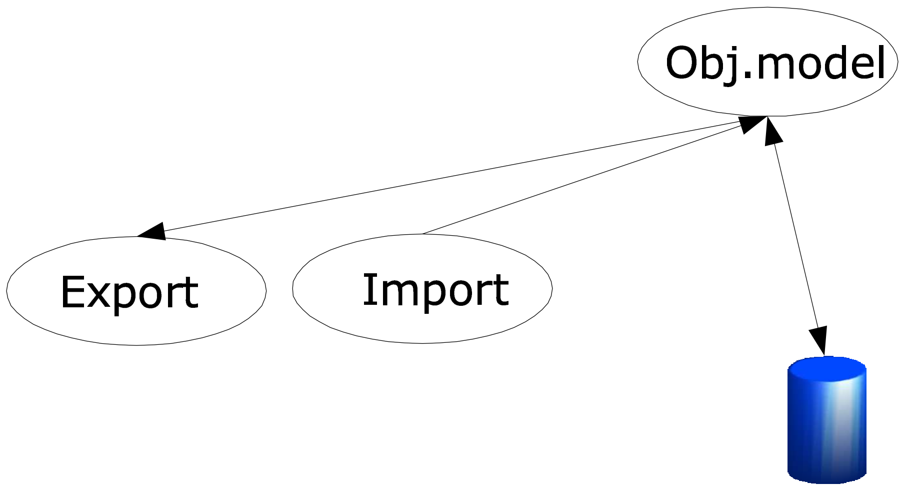
Architecture of the application



The internals of the application



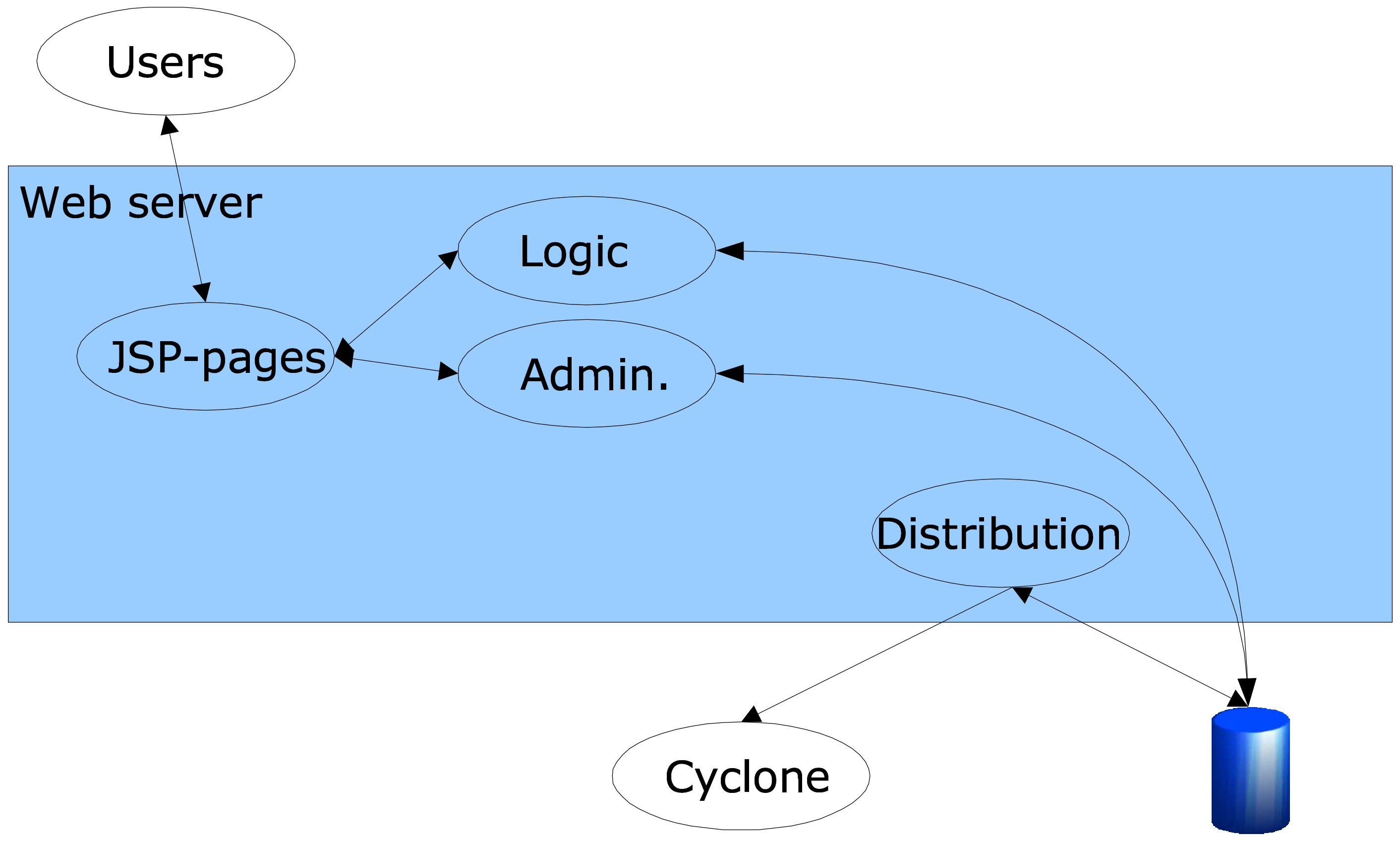
The XML part



Native XML databases

- **XML databases have been on the rise for the past few years**
  - these are databases whose storage model is XML
  - in other words, they store XML directly
  - query languages tend to be XPath and/or XQuery
- **Reasons for using XML databases include**
  - supports semi-structured data
  - may be faster when only specific views wanted (fewer joins)
  - no impedance mismatch with interchange format
  - well suited to document storage
- **Reasons not to use them are**
  - few mature products yet
  - SQL and RDBMSs usually do the same job better

Using an XML database



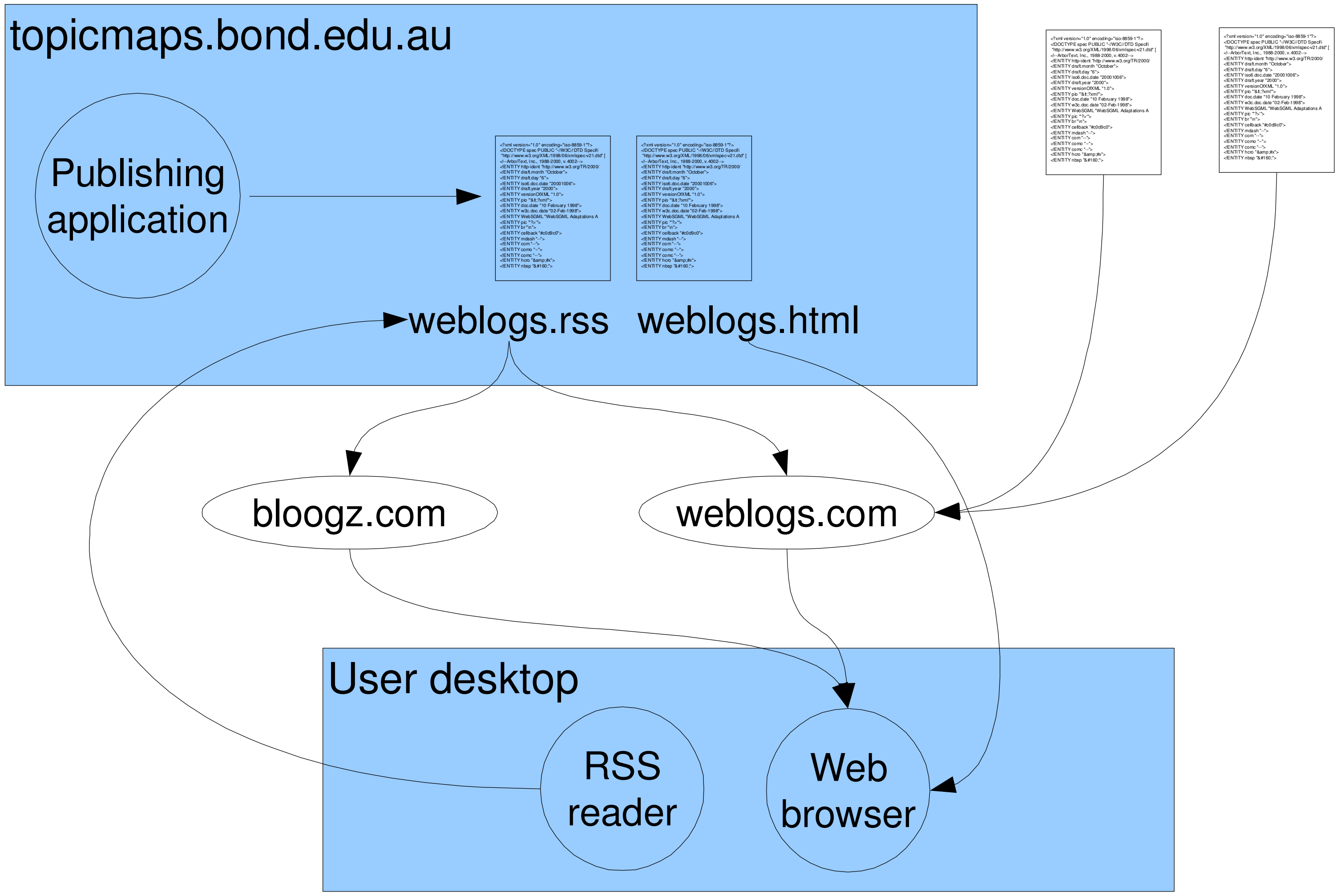
Other considerations

- **Using an XML database would have simplified the regional applications**
  - no need for the object model, since application is simple editor
  - however, validation would have been somewhat awkward to add
- **The central application is different, however**
  - limited need for editing
  - main need is advanced reporting
  - advanced reporting means complex queries and joins
  - XML databases are not well suited for this
  - solution also needs support for replication, which few XML DBs have

A different kind of information system

- **RSS is**
  - a simple XML format for newsfeeds
  - probably the simplest useful XML application there is
  - probably the most widespread XML application
- **Today there are**
  - tens of thousands of RSS feeds
  - lots of news aggregation sites using RSS
  - lots of desktop tools for reading RSS feeds directly

Information system?



Web services

What they are
The promise of web services

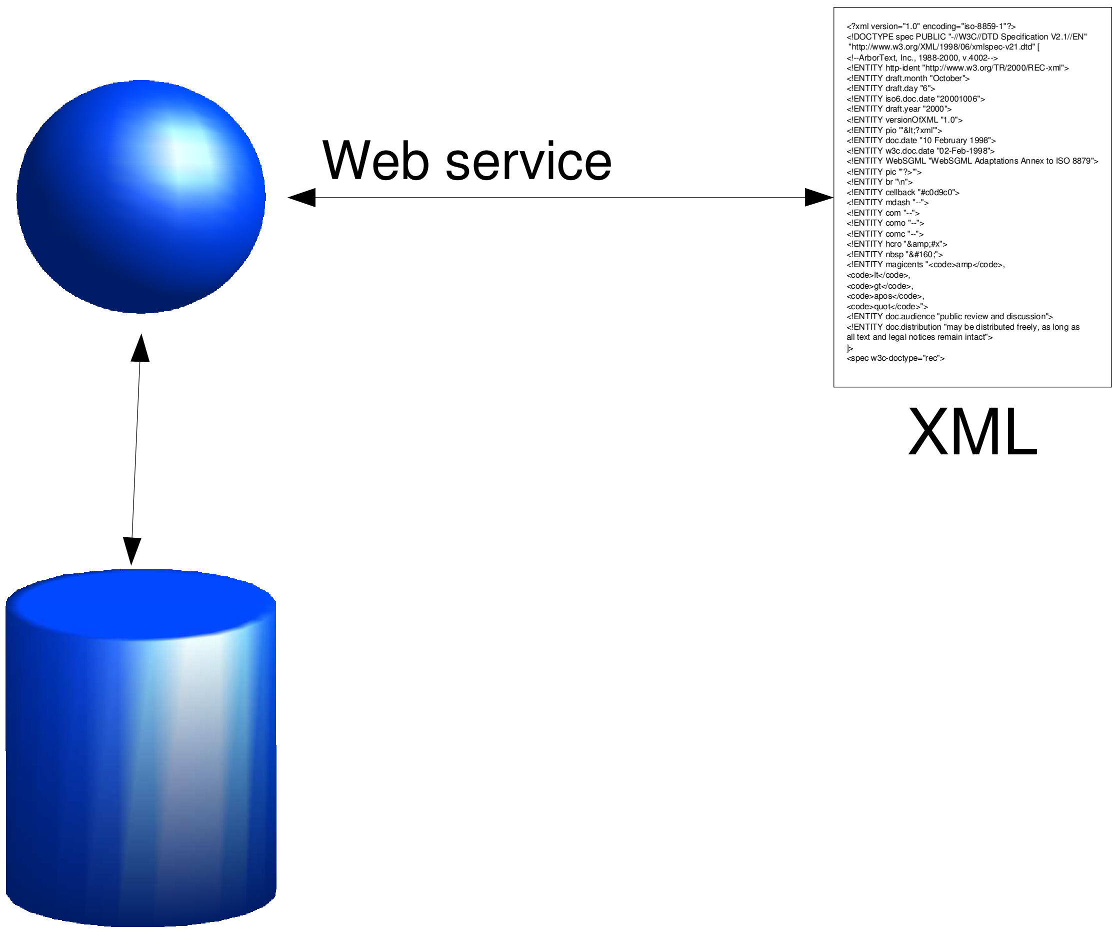
What is a web service, anyway?

- **Basically any software service made available over http**
  - must be intended to be invoked by another piece of software
  - line is somewhat blurry: is Google a web service? MapQuest?
- **Two schools of thought:**
  - REST holds that http + XML has all that is needed
  - the SOAP camp wants special protocols and standards
- **In practice we see both**
  - REST is good because it fits seamlessly into the existing web
  - SOAP is good because it has better tool support
- **Make your choice based on what is important for you**

SOAP

- **Essentially a wrapper for XML messages**
- **Consists of**
  - a header (with routing information etc)
  - a body (which holds the message)
- **Very little is defined in terms of message structure**
- **Effectively, SOAP encapsulates XML, and you must figure out how to deal with the XML yourself**

Web services and architecture



```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE spec PUBLIC "-//W3C//DTD Specification V2.1//EN"
 "http://www.w3.org/XML/1998/06/xmlspec-v21.dtd" [
<!--ArborText, Inc., 1988-2000, v.4002-->
<!ENTITY http-ident "http://www.w3.org/TR/2000/REC-xml">
<!ENTITY draft.month "October">
<!ENTITY draft.day "6">
<!ENTITY iso6.doc.date "20001006">
<!ENTITY draft.year "2000">
<!ENTITY versionOfXML "1.0">
<!ENTITY pio "'&lt;?xml'">
<!ENTITY doc.date "10 February 1998">
<!ENTITY w3c.doc.date "02-Feb-1998">
<!ENTITY WebSGML "WebSGML Adaptations Annex to ISO 8879">
<!ENTITY pic "'?>'">
<!ENTITY br "\n">
<!ENTITY cellback "#c0d9c0">
<!ENTITY mdash "--">
<!ENTITY com "--">
<!ENTITY como "--">
<!ENTITY comc "--">
<!ENTITY hcro "&amp;#x">
<!ENTITY nbsp "&#160;">
<!ENTITY magicents "<code>amp</code>,
<code>lt</code>,
<code>gt</code>,
<code>apos</code>,
<code>quot</code>">
<!ENTITY doc.audience "public review and discussion">
<!ENTITY doc.distribution "may be distributed freely, as long as
all text and legal notices remain intact">
]>
<spec w3c-doctype="rec">
```

The promise of web services

- **Connect legacy applications**
- **Create services anyone can connect to and use**
- **Integrate disparate applications across the enterprise**
- **Publish your service in a web service marketplace**
  - people can find it using UDDI and bind to it dynamically with WSDL
  - you will, of course, charge them for this

A word of caution

- **We've heard all this before**
- **CORBA was widely touted as doing the same thing in the '90s**
  - applications connecting to each other over the net
  - CORBA as the enterprise-wide “bus” connecting all applications
  - directory services and dynamic service binding
  - component brokers and online trading
- **CORBA did the first, but not the last three**
  - political, economic, and legal issues intruded
  - information integration turns out to be difficult
  - dynamic service binding was harder than anyone thought
- **In short, exposing services on the net works**
  - be skeptical about the rest

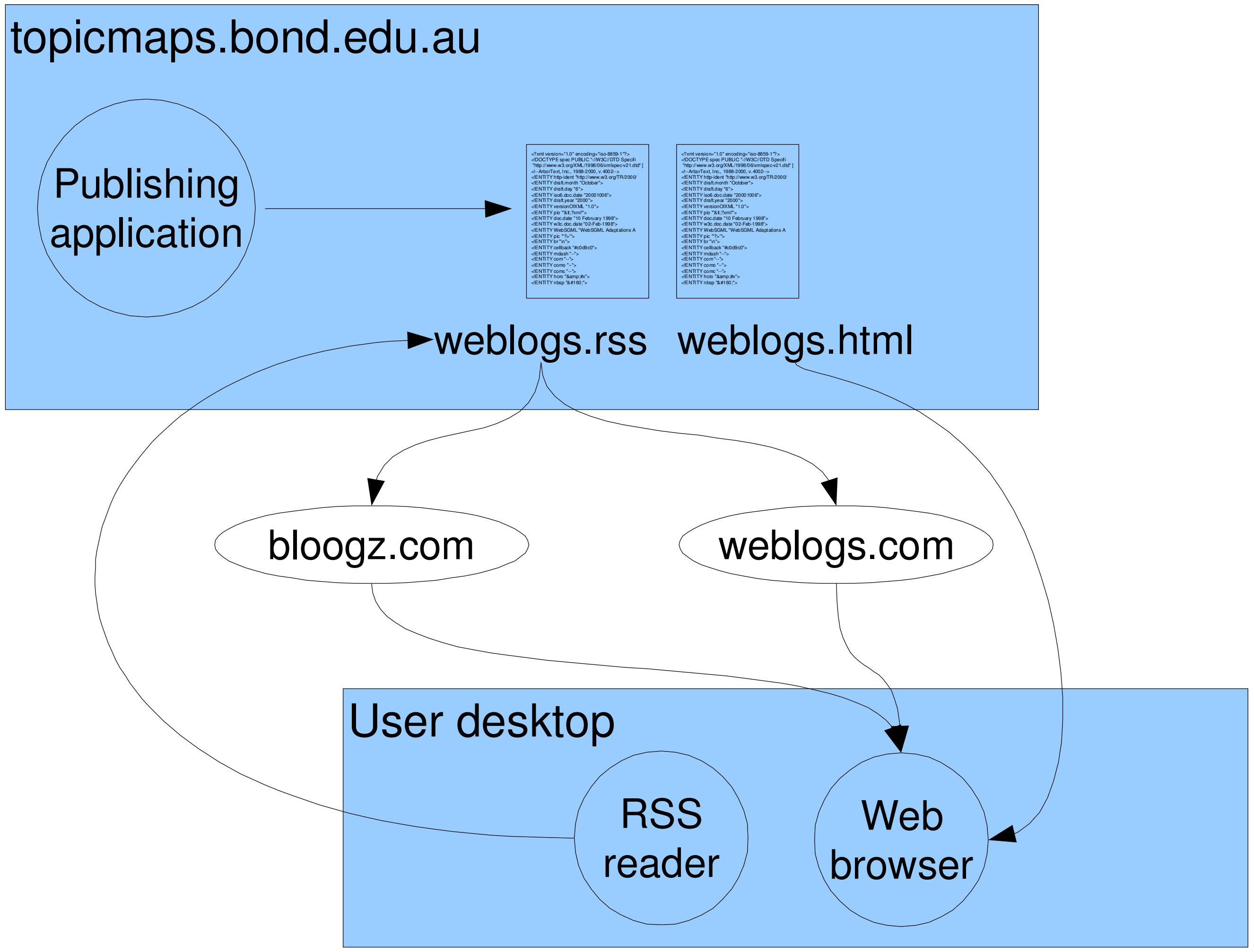
Another caution

- **Integrating applications is not really the issue**
  - what is necessary is to integrate the information
  - XML is about information, but it's not really designed for integration
- **XML has no notion of identity**
  - no way to say when two elements represent the same thing
  - nothing tells you what to do when two elements *do* represent the same thing
- **Knowledge technologies are about identity**
  - they have rules for identity and merging
  - better suited for information integration
  - thus also for application integration

What web services are, second try

- **In other words, web services are an idea more than anything else**
- **In some cases new technology makes it easier to apply**
- **The idea is what matters, however**
  - seeing the possibilities and trying to make use of them
  - which way you do it always matters less than doing it

Web services?



Common XML challenges

Import/export
Important groups of tools
Validation
Using XML databases

Deserialization

- **That is, building an object structure from XML**
- **Usually involves some level of validation as well**
- **Several ways to do this**
  - use SAX, which is low-level but fast
  - use DOM, which is high-level and awful
  - use XPath, which lets you extract information easily
  - use a data binding tool

SAX

- **Standard for event-based parser APIs**
  - passes the document to the application piece by piece
  - somewhat like staring at a parade through a keyhole
  - very fast, consumes no memory at all
  - suitable for applications where
    - documents may be big
    - documents require heavy processing
- **De-facto standard created by self-appointed group**
  - supported by pretty much every parser there is
  - effectively the foundation for all XML work in Java
  - less standardized in other languages

DOM

- **Presents the document as an object structure**
- **W3C Recommendation**
  - widely supported and widely derided
  - in most programming languages better alternatives are found
  - in Java JDOM and XOM are good alternatives
- **Downsides**
  - this approach requires the entire document to be loaded into memory
  - using an API is awkward, whether tree-based or event-based

SAX vs DOM

- **Or, rather, event-based vs tree-based**
  - most XML technologies use one of these two approaches
  - understanding the difference is important in order to choose correctly
- **Essentially the difference is this**
  - event-based solutions require less resources
  - however, they make many common operations too hard to be practical
  - tree-based solutions are slower and use more memory
  - but there is no limit on what you can do
- **Which approach is the right one depends on the requirements**

XPath

- **A simple query language for XML**
  - remarkably simple to learn given its expressive power
  - graph-traversal semantics
- **Simplifies extracting information from XML enormously**
  - probably the single most important XML specification
  - used in query languages, mapping tools, schema languages, ...
- **Much less powerful than SQL**
  - can't return structured results, only a list of values
  - limited support for handling reference relationships
  - no support for aggregate function

Data binding tools

- **Tools that simplify serialization and deserialization**
  - automate as much as possible of those tasks
  - some generate the object model for you
  - others let you map the XML to your object model
- **Most such tools have limitations**
  - no support for mixed content
  - no support for element order
  - ignore comments, processing instructions, and entities
  - limited support for references
- **When suitable they can simplify development considerably**
  - some event-based, others tree-based

Validation

- **Validation is to ensure the correctness of incoming data**
  - that every <person> has a <birth-date>
  - that every <birth-date> is a valid date
  - that every <death-date> is later than the <birth-date>
  - ...
- **These three constraints can be grouped into**
  - structural constraints
  - type constraints
  - “semantic” constraints
- **Schema language can be used to define the first two**
  - application logic must usually be used for the latter

Schema languages

- **DTDs**
  - part of XML 1.0, but only supports structural constraints
  - serious problem: the document says which schema to use
- **XML Schema**
  - has both structural and type constraints
  - W3C Recommendation, widely supported and widely criticized
- **RELAX-NG**
  - has very strong structural and type constraints
  - ISO standard, growing support and widely praised
- **Schematron**
  - weak structural and type constraints, strong on semantic constraints
  - constraints specified with XPath
  - about to become an ISO standard

Serialization

- **The opposite of deserialization: writing XML from objects**
- **Straightforward, but some pitfalls**
  - remember to quote special XML characters *everywhere*
  - handling character encodings correctly
  - handling namespaces correctly
- **Validation usually part of testing, but otherwise not an issue**
  - one assumes the object structure is already valid
- **Again several ways to do it**
  - use simple print statements, and do all the above yourself
  - use a SAX2XML tool, which will handle the above for you
  - build a DOM instance, then write it out (slow and awkward)
  - use a data binding tool

Importing XML to an RDBMS

- **A form of deserialization, but with issues of its own**
- **Typical issues are**
  - how to represent mixed content, if allowed
  - dealing with referential integrity
  - data typing
  - recognizing null values
  - validation
- **Again, there are many ways to do this**
  - just hack it in
  - having an XML-to-OO mapper and an OO-to-RDBMS mapper
  - using a data binding tool

Writing XML from an RDBMS

- **A special kind of serialization**
- **Much easier than going the other way**
- **Main problem is matching the desired output format**
- **Several tools to do this**
  - template-based approaches where SQL is embedded in the XML
  - extensions to SQL that allow XML element constructors in SELECT
  - some allow XSLT transformations of the initial output

XQuery

- The query language for XML databases in the future
- Embeds XPath inside a functional programming language
- Progress on XQuery is slow, but language highly regarded
- Likely to become an important tool in the future

SQL/XML

- **ISO SC32 is working on adding XML support to SQL**
  - this involves columns whose data type is XML
  - one assumes XPath expressions can be applied to these
  - probably also support for XML output
- **RDBMS vendors are committed to this**
- **SQL/XML is likely to be a key building block in the future**
  - simplifies XML storage in databases
  - does *not,* however, remove the impedance mismatch
- **SQL/XML may well become an XQuery killer**

Wrapping up

What XML means for developers
Resources to learn more

XML and software development

- **The possibilities for interchange and integration are not new**
  - XML makes them easier to achieve
  - XML makes us think of these possibilities in ways we didn't before
- **In practice, this means more work for developers**
  - new lists of acronyms to learn and master
  - new kinds of tasks compared to earlier
- **XML makes life harder, but it's worth it**

Where to learn more

- http://www.xml.com
- http://www.xmlhack.com
- The XML-DEV mailing list
- http://www.w3.org/TR/
- *"Definitive XML Application Development"* by me, published by Prentice-Hall

DEFINITIVE
XML
APPLICATION DEVELOPMENT
CD-ROM INCL
Complete developer's guide to XML programming by a leading XML developer