An XML based architecture for Web 2.0 applications (: all or nothing about XML:)

Daniela Florescu
Oracle Corporation

Disclaimer

- This is not Oracle's point of view
- This is architectural work
- This is an industrial point of view (pragmatic)
- The solutions aren't complete
  - It is almost a research agenda at this point
- This isn't implemented (or partially)
- I have no performance graphs
- I have 80 slides…

Outline

- How do people build applications now ?
- Why is the situation so bad ?
- Requirements for a new architecture
- Can XML help ? Why ?
- The current XML infrastructure, the role of XQuery
- What is missing in the XML stack ?
- Putting things together: a proposal for a new architecture.
- The XML information hub.

Example application

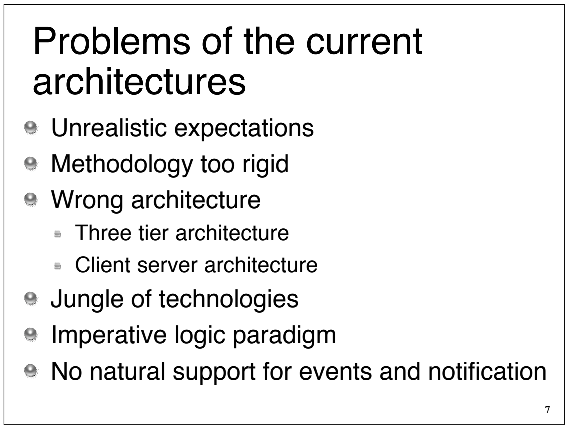
- My pet application: an open, community-based digital review database research system
  - We just finished the Sigmod review cycle :-)
- All papers should be publicly available
  - Storage, archive, index, search
- Everybody should be able to review any paper
- Discussions about a certain paper or topic should be open
  - Correlated information, text not enough
- Blogging, authorities system, etc
- Security, anti-spam, user identification, etc
- Notification system built in the system
- Serious enterprise applications: CRM, ERP, supply management, telcoms, SAP, Siebel, Salesforce, etc

Current implementation

- Probably now: 6 months for 10 people
- My goal: 2 weeks for 2 people
- What functionality do people need
  - Data storage, persistence
  - Application logic
  - Communication with the rest of the world
- What guarantees people need:
  - Reliability, availability
  - Performance
  - Security
- How do they obtain this ?
  - Persistence: relational databases (SQL, Oracle :-)
  - Code: application servers (J2EE, Oracle :-)
  - Communication with the rest of the world: the XML stack -- XML, XML Schema, Web services, etc

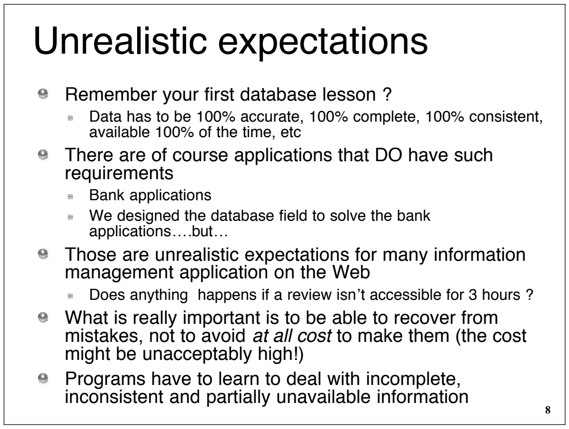
Why is everyone complaining?

- The situation is extremely bad
  - Thousands of developers for a simple application
  - Costs are extremely high
  - The pain reached the point were people are totally blocked. No more new applications are being built now, or very little.
- Too much software to be acquired, installed, maintained, and upgraded
- Productivity
  - Needs skills in a variety of technologies, each VERY difficult (SQL stack, J2EE stack, XML stack…). Thousands or pages of documentations.
  - Lots of new glue clue code to be written
- Performance
  - So many overlaying layers. Glue code is very expensive too.
- Applications are brittle
- Impossible to change, customize and evolve



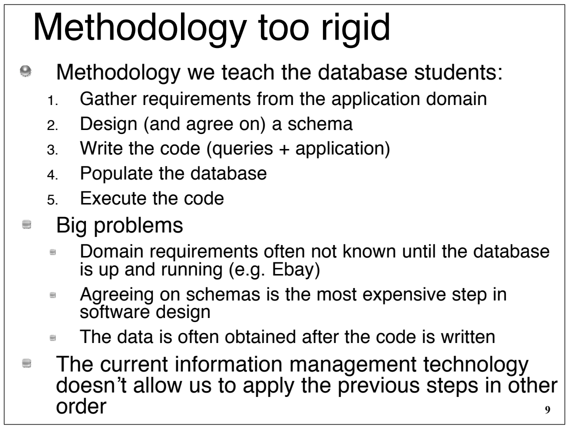
Problems of the current architectures

- Unrealistic expectations
- Methodology too rigid
- Wrong architecture
  - Three tier architecture
  - Client server architecture
- Jungle of technologies
- Imperative logic paradigm
- No natural support for events and notification



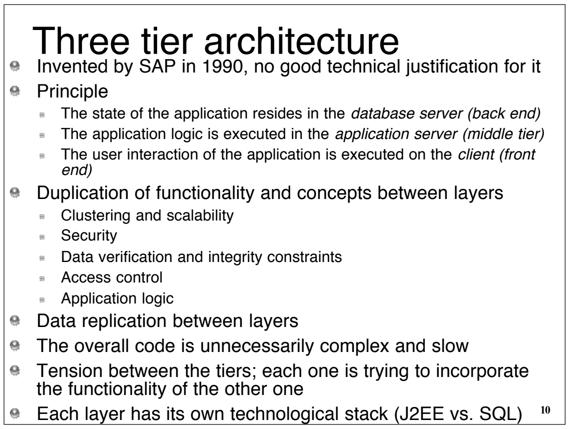
Unrealistic expectations

- Remember your first database lesson ?
  - Data has to be 100% accurate, 100% complete, 100% consistent, available 100% of the time, etc
- There are of course applications that DO have such requirements
  - Bank applications
  - We designed the database field to solve the bank applications….but…
- Those are unrealistic expectations for many information management application on the Web
  - Does anything happens if a review isn't accessible for 3 hours ?
- What is really important is to be able to recover from mistakes, not to avoid *at all cost* to make them (the cost might be unacceptably high!)
- Programs have to learn to deal with incomplete, inconsistent and partially unavailable information



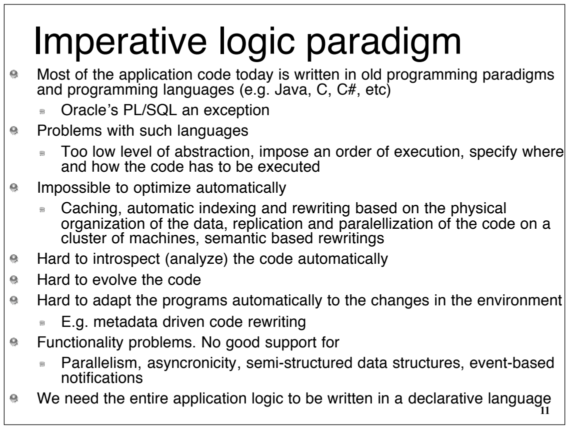
Methodology too rigid

- Methodology we teach the database students:
  1. Gather requirements from the application domain
  2. Design (and agree on) a schema
  3. Write the code (queries + application)
  4. Populate the database
  5. Execute the code
- Big problems
  - Domain requirements often not known until the database is up and running (e.g. Ebay)
  - Agreeing on schemas is the most expensive step in software design
  - The data is often obtained after the code is written
- The current information management technology doesn't allow us to apply the previous steps in other order



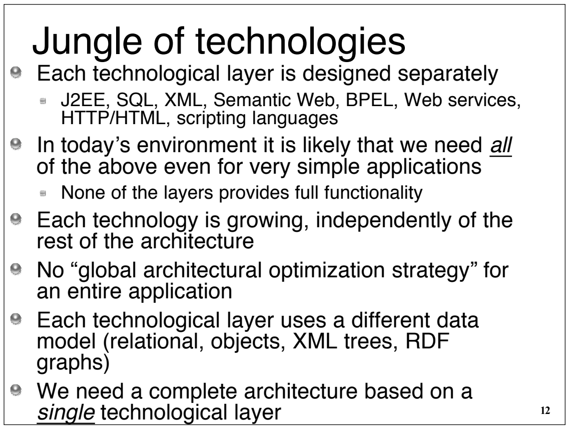
Three tier architecture

- Invented by SAP in 1990, no good technical justification for it
- Principle
  - The state of the application resides in the *database server (back end)*
  - The application logic is executed in the *application server (middle tier)*
  - The user interaction of the application is executed on the *client (front end)*
- Duplication of functionality and concepts between layers
  - Clustering and scalability
  - Security
  - Data verification and integrity constraints
  - Access control
  - Application logic
- Data replication between layers
- The overall code is unnecessarily complex and slow
- Tension between the tiers; each one is trying to incorporate the functionality of the other one
- Each layer has its own technological stack (J2EE vs. SQL)



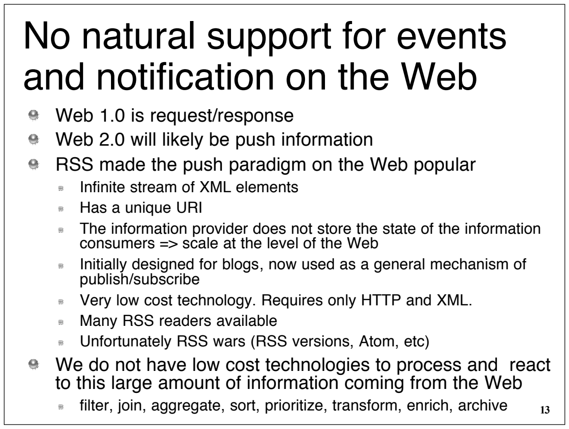
Imperative logic paradigm

- Most of the application code today is written in old programming paradigms and programming languages (e.g. Java, C, C#, etc)
  - Oracle's PL/SQL an exception
- Problems with such languages
  - Too low level of abstraction, impose an order of execution, specify where and how the code has to be executed
- Impossible to optimize automatically
  - Caching, automatic indexing and rewriting based on the physical organization of the data, replication and paralellization of the code on a cluster of machines, semantic based rewritings
- Hard to introspect (analyze) the code automatically
- Hard to evolve the code
- Hard to adapt the programs automatically to the changes in the environment
  - E.g. metadata driven code rewriting
- Functionality problems. No good support for
  - Parallelism, asyncronicity, semi-structured data structures, event-based notifications
- We need the entire application logic to be written in a declarative language



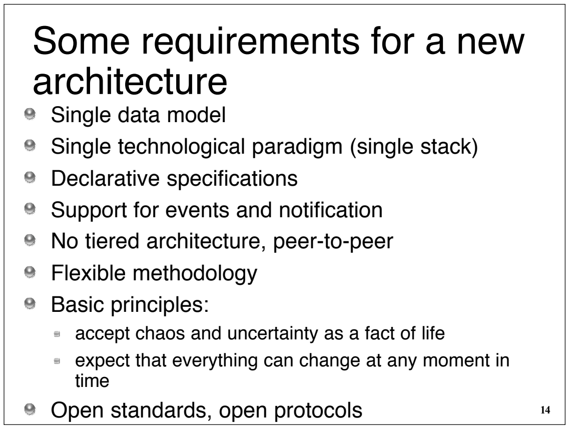
Jungle of technologies

- Each technological layer is designed separately
  - J2EE, SQL, XML, Semantic Web, BPEL, Web services, HTTP/HTML, scripting languages
- In today's environment it is likely that we need *all* of the above even for very simple applications
  - None of the layers provides full functionality
- Each technology is growing, independently of the rest of the architecture
- No "global architectural optimization strategy" for an entire application
- Each technological layer uses a different data model (relational, objects, XML trees, RDF graphs)
- We need a complete architecture based on a *single* technological layer



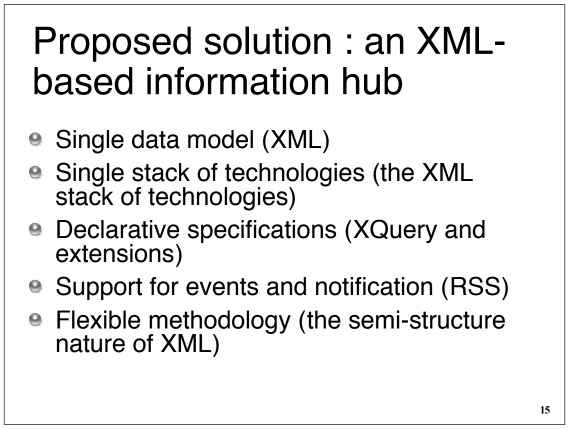
No natural support for events and notification on the Web

- Web 1.0 is request/response
- Web 2.0 will likely be push information
- RSS made the push paradigm on the Web popular
  - Infinite stream of XML elements
  - Has a unique URI
  - The information provider does not store the state of the information consumers => scale at the level of the Web
  - Initially designed for blogs, now used as a general mechanism of publish/subscribe
  - Very low cost technology. Requires only HTTP and XML.
  - Many RSS readers available
  - Unfortunately RSS wars (RSS versions, Atom, etc)
- We do not have low cost technologies to process and react to this large amount of information coming from the Web
  - filter, join, aggregate, sort, prioritize, transform, enrich, archive



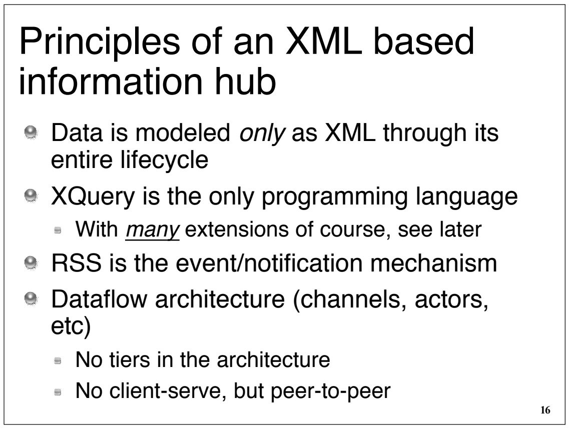
Some requirements for a new architecture

- Single data model
- Single technological paradigm (single stack)
- Declarative specifications
- Support for events and notification
- No tiered architecture, peer-to-peer
- Flexible methodology
- Basic principles:
  - accept chaos and uncertainty as a fact of life
  - expect that everything can change at any moment in time
- Open standards, open protocols



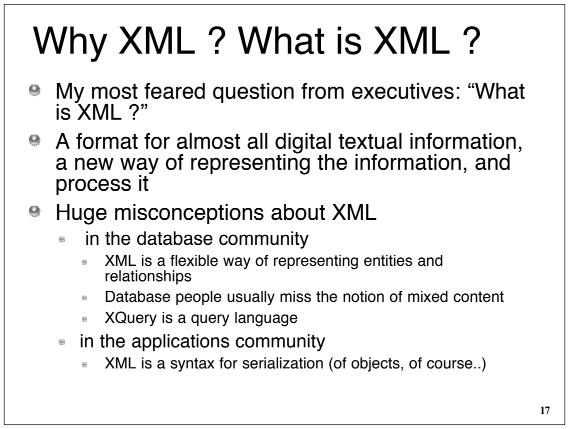
Proposed solution : an XML-based information hub

- Single data model (XML)
- Single stack of technologies (the XML stack of technologies)
- Declarative specifications (XQuery and extensions)
- Support for events and notification (RSS)
- Flexible methodology (the semi-structure nature of XML)



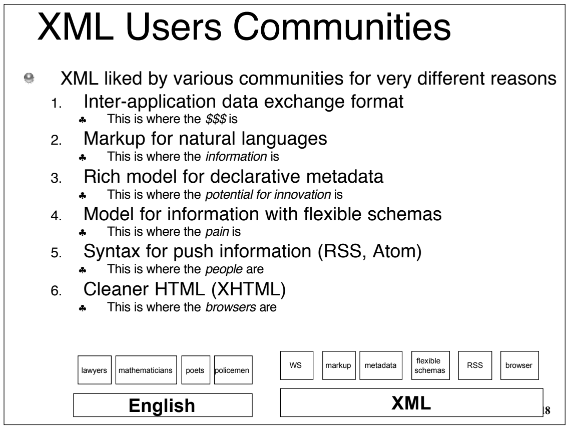
Principles of an XML based information hub

- Data is modeled *only* as XML through its entire lifecycle
- XQuery is the only programming language
  - With *many* extensions of course, see later
- RSS is the event/notification mechanism
- Dataflow architecture (channels, actors, etc)
  - No tiers in the architecture
  - No client-serve, but peer-to-peer



Why XML ? What is XML ?

- My most feared question from executives: “What is XML ?”
- A format for almost all digital textual information, a new way of representing the information, and process it
- Huge misconceptions about XML
  - in the database community
    - XML is a flexible way of representing entities and relationships
    - Database people usually miss the notion of mixed content
    - XQuery is a query language
  - in the applications community
    - XML is a syntax for serialization (of objects, of course..)



XML Users Communities

- XML liked by various communities for very different reasons
  1. Inter-application data exchange format
     - This is where the *$$$* is
  2. Markup for natural languages
     - This is where the *information* is
  3. Rich model for declarative metadata
     - This is where the *potential for innovation* is
  4. Model for information with flexible schemas
     - This is where the *pain* is
  5. Syntax for push information (RSS, Atom)
     - This is where the *people* are
  6. Cleaner HTML (XHTML)
     - This is where the *browsers* are

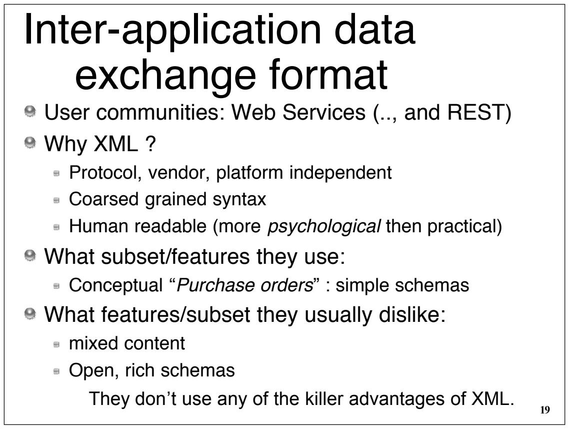
lawyers | mathematicians | poets | policemen

English

WS | markup | metadata | flexible schemas | RSS | browser

XML

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

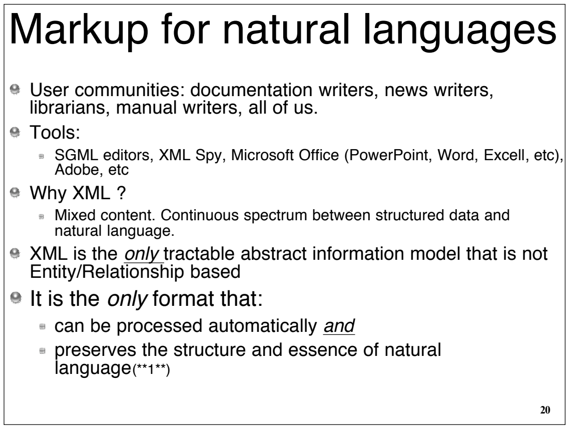


Inter-application data exchange format

- User communities: Web Services (.., and REST)
- Why XML ?
  - Protocol, vendor, platform independent
  - Coarsed grained syntax
  - Human readable (more *psychological* then practical)
- What subset/features they use:
  - Conceptual "*Purchase orders*" : simple schemas
- What features/subset they usually dislike:
  - mixed content
  - Open, rich schemas

They don't use any of the killer advantages of XML.

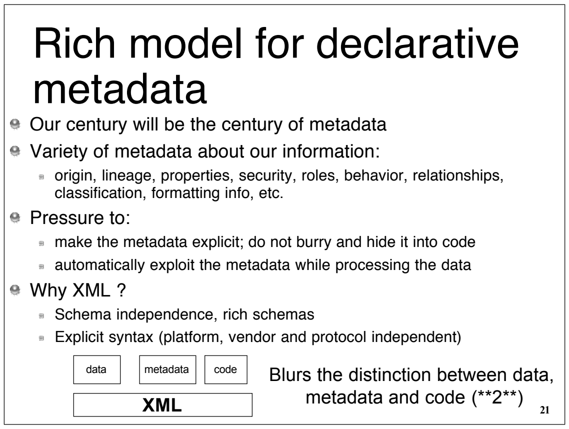
19



Markup for natural languages

- User communities: documentation writers, news writers, librarians, manual writers, all of us.
- Tools:
  - SGML editors, XML Spy, Microsoft Office (PowerPoint, Word, Excell, etc), Adobe, etc
- Why XML ?
  - Mixed content. Continuous spectrum between structured data and natural language.
- XML is the *only* tractable abstract information model that is not Entity/Relationship based
- It is the *only* format that:
  - can be processed automatically *and*
  - preserves the structure and essence of natural language(**1**)

20



Rich model for declarative metadata

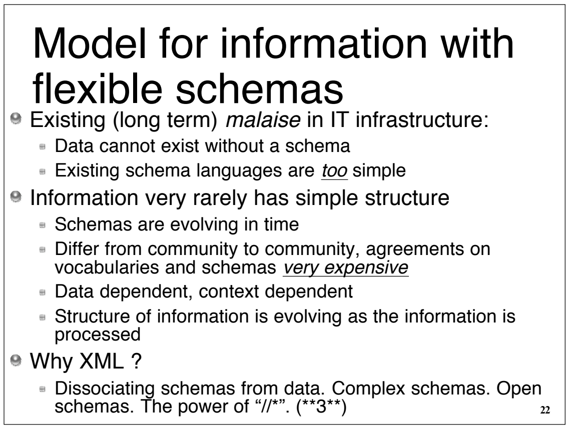
- Our century will be the century of metadata
- Variety of metadata about our information:
  - origin, lineage, properties, security, roles, behavior, relationships, classification, formatting info, etc.
- Pressure to:
  - make the metadata explicit; do not burry and hide it into code
  - automatically exploit the metadata while processing the data
- Why XML ?
  - Schema independence, rich schemas
  - Explicit syntax (platform, vendor and protocol independent)

data | metadata | code

XML

Blurs the distinction between data, metadata and code (**2**)

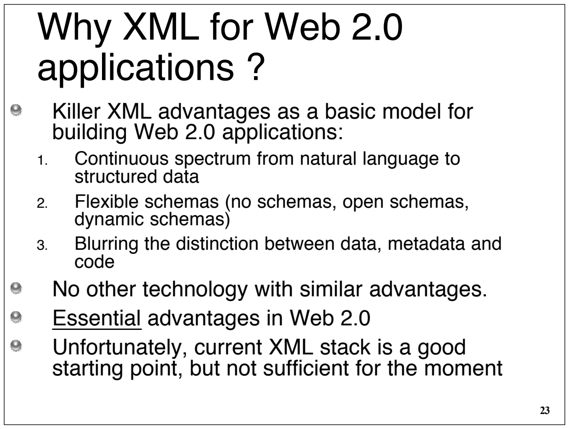
21



Model for information with flexible schemas

- Existing (long term) *malaise* in IT infrastructure:
  - Data cannot exist without a schema
  - Existing schema languages are *too* simple
- Information very rarely has simple structure
  - Schemas are evolving in time
  - Differ from community to community, agreements on vocabularies and schemas *very expensive*
  - Data dependent, context dependent
  - Structure of information is evolving as the information is processed
- Why XML ?
  - Dissociating schemas from data. Complex schemas. Open schemas. The power of "//*". (**3**)

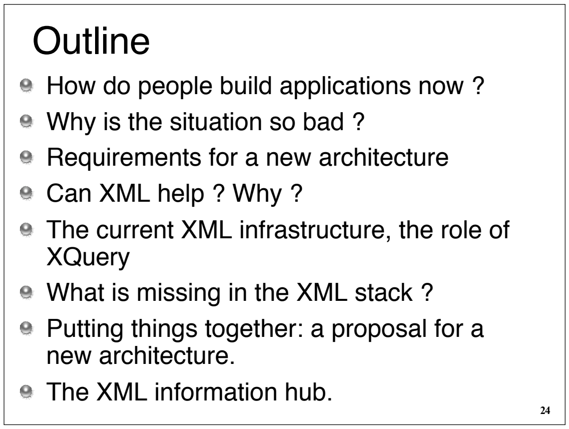
22



Why XML for Web 2.0 applications ?

- Killer XML advantages as a basic model for building Web 2.0 applications:
  1. Continuous spectrum from natural language to structured data
  2. Flexible schemas (no schemas, open schemas, dynamic schemas)
  3. Blurring the distinction between data, metadata and code
- No other technology with similar advantages.
- Essential advantages in Web 2.0
- Unfortunately, current XML stack is a good starting point, but not sufficient for the moment

23



Outline

- How do people build applications now ?
- Why is the situation so bad ?
- Requirements for a new architecture
- Can XML help ? Why ?
- The current XML infrastructure, the role of XQuery
- What is missing in the XML stack ?
- Putting things together: a proposal for a new architecture.
- The XML information hub.

24

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

XML as a family of technologies

- *XML Information Set*
- *XML Schema*
- *XML Query*
- *The Extensible Stylesheet Transformation Language (XSLT)*
- *XLink, XPointer*
- *XML Forms*
- *XML Protocol*
- *XML Encryption*
- *XML Signature*
- *Others*
- *… almost all the pieces needed for a good XML-based information hub*

Processing XML

- Let’s assume a perfect world :-)
  - information modeled *only* in XML inside an application
- What do we need to do with it ?
  - Store, replicate, warehouse it
  - Verify the correctness
  - Filter, search, select, join, aggregate
  - Create new data
  - Modify existing data
  - Take actions based on the content of the existing data
  - Exchange the data (send/receive)
  - Create complex execution flows
- Current existing solutions
  - Use generic programming APIs (e.g. DOM, SAX)
  - Manually or automatically map XML to non-generic programming structures (e.g. code generators)
  - Use XML extensions of existing languages (Python, Perl, C#, ECMA) (*)
  - Shredding for relational stores
  - Native XML processing through XSLT and XQuery (***)

What is XQuery

- *Declarative XML to XML mapping language*
- *XML* := abstract XML Data Model (not the syntax)
- XQuery properties
  - Preserves the logical/physical data independence
  - Declarative (describes the “*what*”, not the “*how*”)
  - Turing complete
  - Side-effect free
  - Strongly typed. Typing optional.
  - Processed schema validated and non-validated data
- XQuery is an embeddable expression language
- XQuery is *not* a *query* language. Misnomer.
- XQuery vs. XSLT: same language, different programming paradigms

A fraction of a real customer Xquery

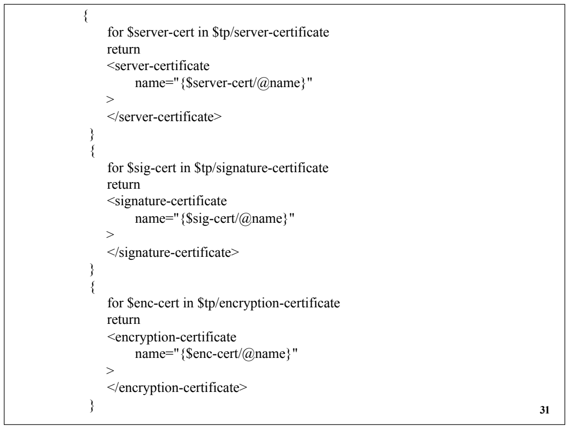
(i.e. *Xquery is not a query language*

Xquery is a programming language)

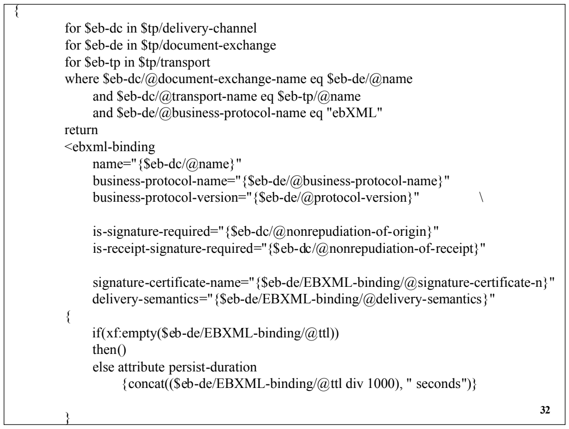
```
let $wlc := document("tests/ebsample/data/ebSample.xml")
let $ctrlPackage := "foo.pkg"
let $wfPath := "test"

let $tp-list :=
for $tp in $wlc/wlc/trading-partner
return
<trading-partner
      name="{$tp/@name}"
      business-id="{$tp/party-identifier/@business-id}"
      description="{$tp/@description}"
      notes="{$tp/@notes}"
      type="{$tp/@type}"
      email="{$tp/@email}"
      phone="{$tp/@phone}"
      fax="{$tp/@fax}"
      username="{$tp/@user-name}"
```

```
{
      for $tp-ad in $tp/address
      return
        $tp-ad
  }
  {
      for $eps in $wlc/extended-property-set
      where $tp/@extended-property-set-name eq $eps/@name
      return
         $eps
  }
  {
      for $client-cert in $tp/client-certificate
      return
      <client-certificate
            name="{$client-cert/@name}"
      >
      </client-certificate>
  }
```

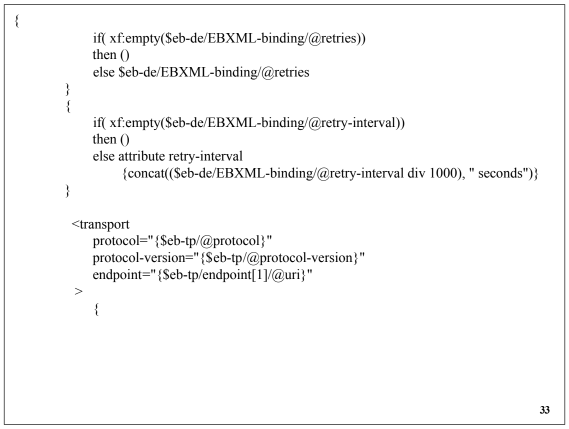
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.



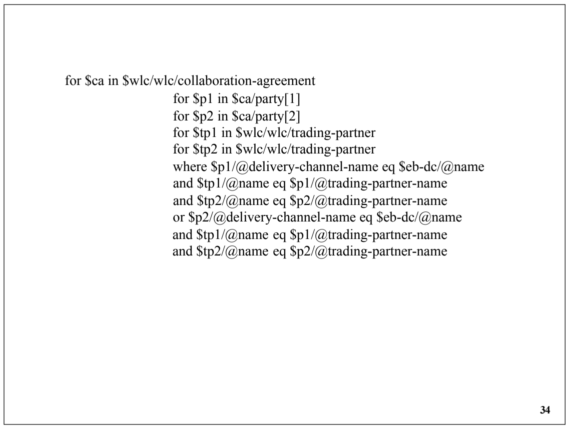
```
        {
          for $server-cert in $tp/server-certificate
          return
          <server-certificate
               name="{$server-cert/@name}"
          >
          </server-certificate>
        }
        {
          for $sig-cert in $tp/signature-certificate
          return
          <signature-certificate
               name="{$sig-cert/@name}"
          >
          </signature-certificate>
        }
        {
          for $enc-cert in $tp/encryption-certificate
          return
          <encryption-certificate
               name="{$enc-cert/@name}"
          >
          </encryption-certificate>
        }
```



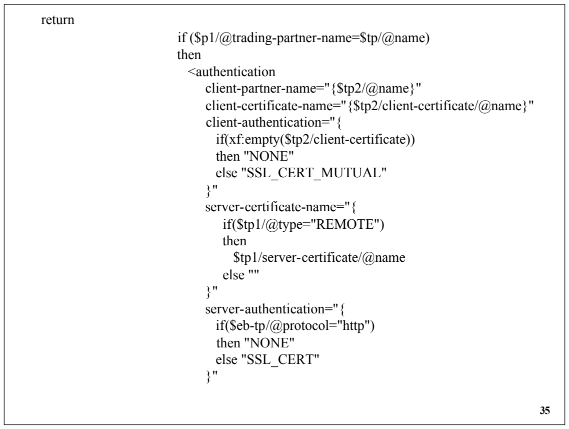
```
{
        for $eb-dc in $tp/delivery-channel
        for $eb-de in $tp/document-exchange
        for $eb-tp in $tp/transport
        where $eb-dc/@document-exchange-name eq $eb-de/@name
              and $eb-dc/@transport-name eq $eb-tp/@name
              and $eb-de/@business-protocol-name eq "ebXML"
        return
        <ebxml-binding
              name="{$eb-dc/@name}"
              business-protocol-name="{$eb-de/@business-protocol-name}"
              business-protocol-version="{$eb-de/@protocol-version}"              \

              is-signature-required="{$eb-dc/@nonrepudiation-of-origin}"
              is-receipt-signature-required="{$eb-dc/@nonrepudiation-of-receipt}"

              signature-certificate-name="{$eb-de/EBXML-binding/@signature-certificate-n}"
              delivery-semantics="{$eb-de/EBXML-binding/@delivery-semantics}"
        {
              if(xf:empty($eb-de/EBXML-binding/@ttl))
              then()
              else attribute persist-duration
                    {concat(($eb-de/EBXML-binding/@ttl div 1000), " seconds")}

        }
```



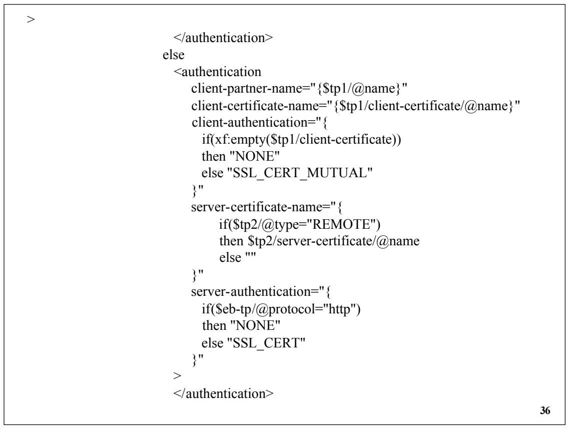
```
{
                if( xf:empty($eb-de/EBXML-binding/@retries))
                then ()
                else $eb-de/EBXML-binding/@retries
           }
           {
                if( xf:empty($eb-de/EBXML-binding/@retry-interval))
                then ()
                else attribute retry-interval
                     {concat(($eb-de/EBXML-binding/@retry-interval div 1000), " seconds")}
           }

            <transport
                protocol="{$eb-tp/@protocol}"
                protocol-version="{$eb-tp/@protocol-version}"
                endpoint="{$eb-tp/endpoint[1]/@uri}"
             >
                {
```



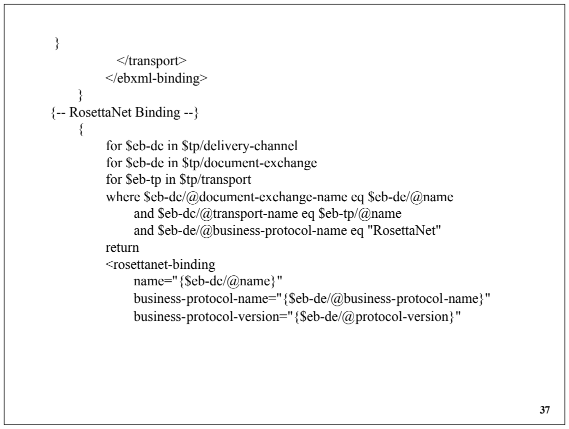
```
for $ca in $wlc/wlc/collaboration-agreement
                    for $p1 in $ca/party[1]
                    for $p2 in $ca/party[2]
                    for $tp1 in $wlc/wlc/trading-partner
                    for $tp2 in $wlc/wlc/trading-partner
                    where $p1/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name
                    or $p2/@delivery-channel-name eq $eb-dc/@name
                    and $tp1/@name eq $p1/@trading-partner-name
                    and $tp2/@name eq $p2/@trading-partner-name
```



```
    return
                          if ($p1/@trading-partner-name=$tp/@name)
                          then
                            <authentication
                               client-partner-name="{$tp2/@name}"
                               client-certificate-name="{$tp2/client-certificate/@name}"
                               client-authentication="{
                                 if(xf:empty($tp2/client-certificate))
                                 then "NONE"
                                 else "SSL_CERT_MUTUAL"
                               }"
                               server-certificate-name="{
                                   if($tp1/@type="REMOTE")
                                   then
                                     $tp1/server-certificate/@name
                                   else ""
                               }"
                               server-authentication="{
                                 if($eb-tp/@protocol="http")
                                 then "NONE"
                                 else "SSL_CERT"
                               }"
```

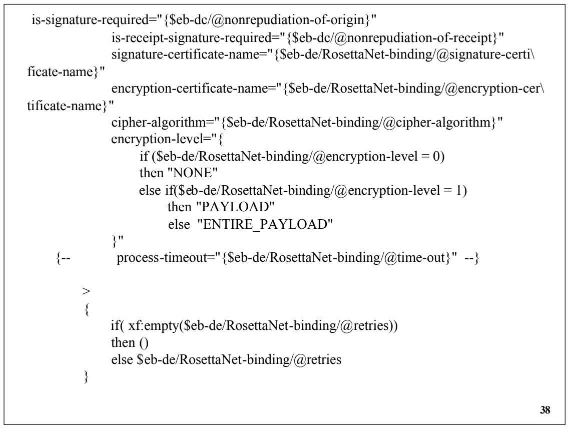


```
  >
                          </authentication>
                        else
                          <authentication
                             client-partner-name="{$tp1/@name}"
                             client-certificate-name="{$tp1/client-certificate/@name}"
                             client-authentication="{
                               if(xf:empty($tp1/client-certificate))
                               then "NONE"
                               else "SSL_CERT_MUTUAL"
                             }"
                             server-certificate-name="{
                                  if($tp2/@type="REMOTE")
                                  then $tp2/server-certificate/@name
                                  else ""
                             }"
                             server-authentication="{
                               if($eb-tp/@protocol="http")
                               then "NONE"
                               else "SSL_CERT"
                             }"
                          >
                          </authentication>
```



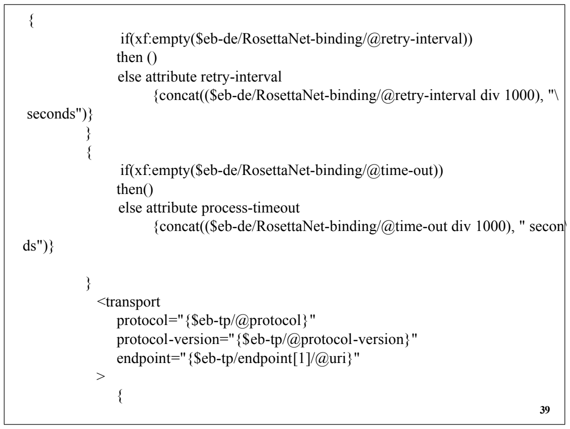
```
 }
                </transport>
              </ebxml-binding>
       }
    {-- RosettaNet Binding --}
       {
              for $eb-dc in $tp/delivery-channel
              for $eb-de in $tp/document-exchange
              for $eb-tp in $tp/transport
              where $eb-dc/@document-exchange-name eq $eb-de/@name
                    and $eb-dc/@transport-name eq $eb-tp/@name
                    and $eb-de/@business-protocol-name eq "RosettaNet"
              return
              <rosettanet-binding
                    name="{$eb-dc/@name}"
                    business-protocol-name="{$eb-de/@business-protocol-name}"
                    business-protocol-version="{$eb-de/@protocol-version}"
```

37



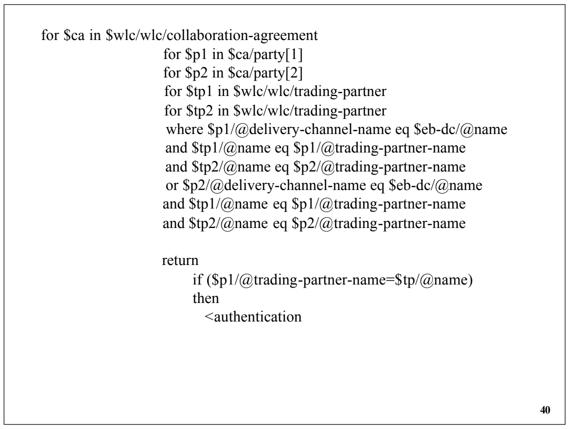
```
 is-signature-required="{$eb-dc/@nonrepudiation-of-origin}"
                   is-receipt-signature-required="{$eb-dc/@nonrepudiation-of-receipt}"
                   signature-certificate-name="{$eb-de/RosettaNet-binding/@signature-certi\
ficate-name}"
                   encryption-certificate-name="{$eb-de/RosettaNet-binding/@encryption-cer\
tificate-name}"
                   cipher-algorithm="{$eb-de/RosettaNet-binding/@cipher-algorithm}"
                   encryption-level="{
                         if ($eb-de/RosettaNet-binding/@encryption-level = 0)
                         then "NONE"
                         else if($eb-de/RosettaNet-binding/@encryption-level = 1)
                                then "PAYLOAD"
                                else  "ENTIRE_PAYLOAD"
                   }"
      {--          process-timeout="{$eb-de/RosettaNet-binding/@time-out}"  --}

            >
            {
                   if( xf:empty($eb-de/RosettaNet-binding/@retries))
                   then ()
                   else $eb-de/RosettaNet-binding/@retries
            }
```

38



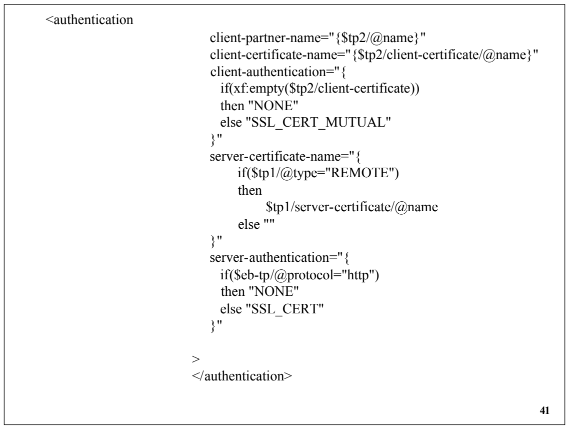
```
  {
                      if(xf:empty($eb-de/RosettaNet-binding/@retry-interval))
                     then ()
                     else attribute retry-interval
                            {concat(($eb-de/RosettaNet-binding/@retry-interval div 1000), "\
 seconds")}
              }
              {
                      if(xf:empty($eb-de/RosettaNet-binding/@time-out))
                     then()
                     else attribute process-timeout
                            {concat(($eb-de/RosettaNet-binding/@time-out div 1000), " secon\
ds")}

              }
                <transport
                     protocol="{$eb-tp/@protocol}"
                     protocol-version="{$eb-tp/@protocol-version}"
                     endpoint="{$eb-tp/endpoint[1]/@uri}"
                >
                     {
```

39



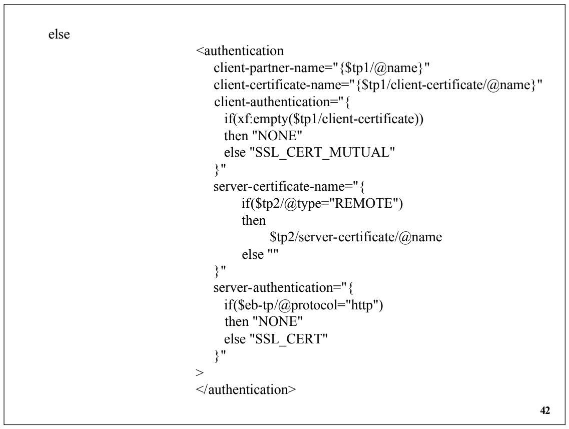
```
  for $ca in $wlc/wlc/collaboration-agreement
                          for $p1 in $ca/party[1]
                          for $p2 in $ca/party[2]
                          for $tp1 in $wlc/wlc/trading-partner
                          for $tp2 in $wlc/wlc/trading-partner
                           where $p1/@delivery-channel-name eq $eb-dc/@name
                           and $tp1/@name eq $p1/@trading-partner-name
                           and $tp2/@name eq $p2/@trading-partner-name
                           or $p2/@delivery-channel-name eq $eb-dc/@name
                          and $tp1/@name eq $p1/@trading-partner-name
                          and $tp2/@name eq $p2/@trading-partner-name

                          return
                                 if ($p1/@trading-partner-name=$tp/@name)
                                 then
                                   <authentication
```

40



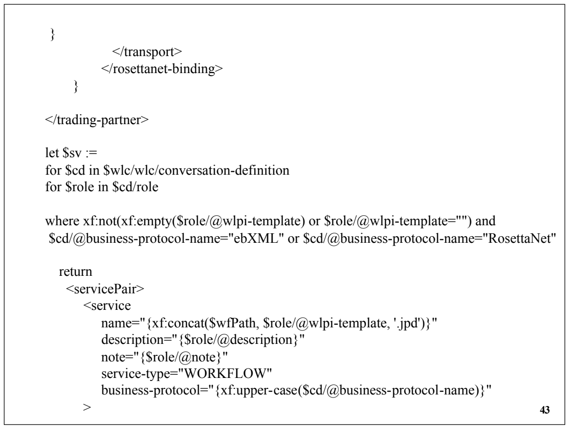
```
   <authentication
                                  client-partner-name="{$tp2/@name}"
                                  client-certificate-name="{$tp2/client-certificate/@name}"
                                  client-authentication="{
                                     if(xf:empty($tp2/client-certificate))
                                     then "NONE"
                                     else "SSL_CERT_MUTUAL"
                                  }"
                                  server-certificate-name="{
                                         if($tp1/@type="REMOTE")
                                         then
                                                $tp1/server-certificate/@name
                                         else ""
                                  }"
                                  server-authentication="{
                                     if($eb-tp/@protocol="http")
                                     then "NONE"
                                     else "SSL_CERT"
                                  }"

                              >
                              </authentication>
```

41



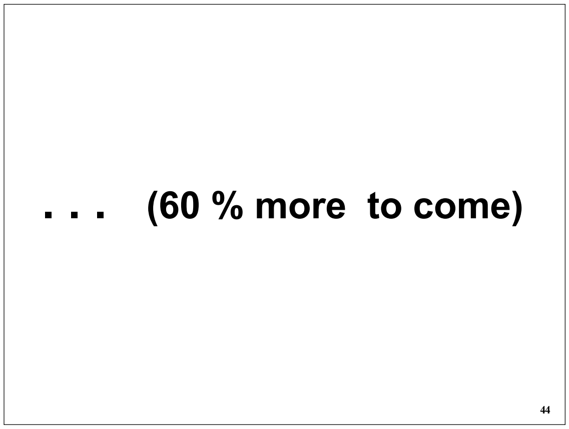
```
  else
                       <authentication
                           client-partner-name="{$tp1/@name}"
                           client-certificate-name="{$tp1/client-certificate/@name}"
                           client-authentication="{
                              if(xf:empty($tp1/client-certificate))
                              then "NONE"
                              else "SSL_CERT_MUTUAL"
                           }"
                           server-certificate-name="{
                                  if($tp2/@type="REMOTE")
                                  then
                                         $tp2/server-certificate/@name
                                  else ""
                           }"
                           server-authentication="{
                              if($eb-tp/@protocol="http")
                              then "NONE"
                              else "SSL_CERT"
                           }"
                       >
                       </authentication>
```

42



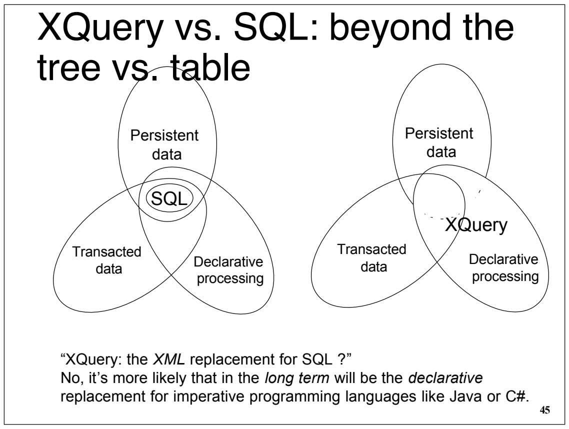
```
  }
                </transport>
            </rosettanet-binding>
        }

</trading-partner>

let $sv :=
for $cd in $wlc/wlc/conversation-definition
for $role in $cd/role

where xf:not(xf:empty($role/@wlpi-template) or $role/@wlpi-template="") and
 $cd/@business-protocol-name="ebXML" or $cd/@business-protocol-name="RosettaNet"

    return
      <servicePair>
        <service
            name="{xf:concat($wfPath, $role/@wlpi-template, '.jpd')}"
            description="{$role/@description}"
            note="{$role/@note}"
            service-type="WORKFLOW"
            business-protocol="{xf:upper-case($cd/@business-protocol-name)}"
        >
```

43



. . .   (60 % more to come)

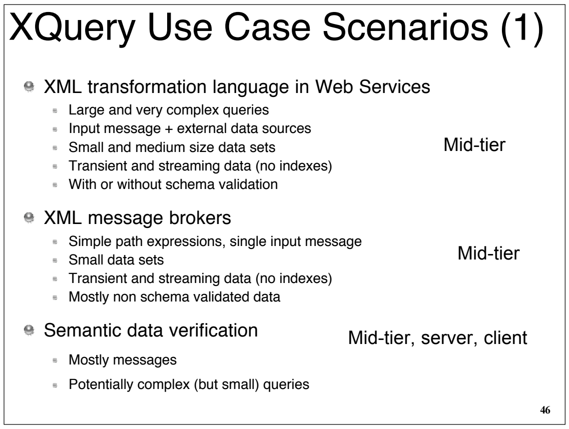
44



XQuery vs. SQL: beyond the tree vs. table

"XQuery: the *XML* replacement for SQL ?"
No, it's more likely that in the *long term* will be the *declarative* replacement for imperative programming languages like Java or C#.

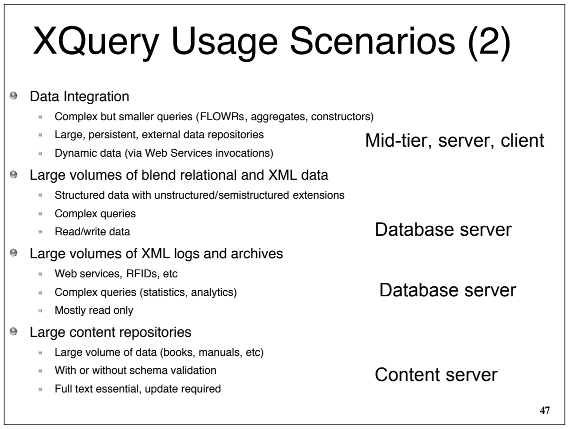
45



XQuery Use Case Scenarios (1)

- XML transformation language in Web Services — Mid-tier
  - Large and very complex queries
  - Input message + external data sources
  - Small and medium size data sets
  - Transient and streaming data (no indexes)
  - With or without schema validation
- XML message brokers — Mid-tier
  - Simple path expressions, single input message
  - Small data sets
  - Transient and streaming data (no indexes)
  - Mostly non schema validated data
- Semantic data verification — Mid-tier, server, client
  - Mostly messages
  - Potentially complex (but small) queries

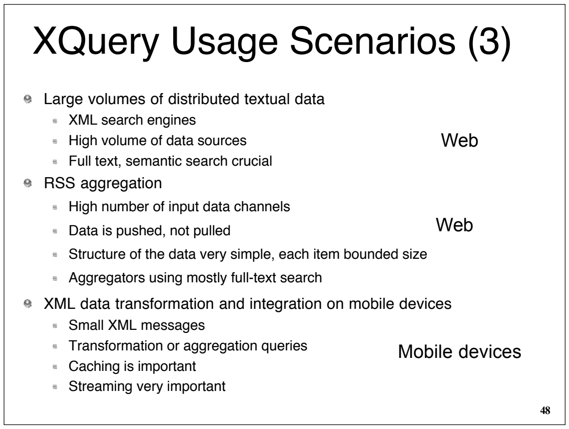
46



XQuery Usage Scenarios (2)

- Data Integration — Mid-tier, server, client
  - Complex but smaller queries (FLOWRs, aggregates, constructors)
  - Large, persistent, external data repositories
  - Dynamic data (via Web Services invocations)
- Large volumes of blend relational and XML data — Database server
  - Structured data with unstructured/semistructured extensions
  - Complex queries
  - Read/write data
- Large volumes of XML logs and archives — Database server
  - Web services, RFIDs, etc
  - Complex queries (statistics, analytics)
  - Mostly read only
- Large content repositories — Content server
  - Large volume of data (books, manuals, etc)
  - With or without schema validation
  - Full text essential, update required

47



XQuery Usage Scenarios (3)

- Large volumes of distributed textual data — Web
  - XML search engines
  - High volume of data sources
  - Full text, semantic search crucial
- RSS aggregation — Web
  - High number of input data channels
  - Data is pushed, not pulled
  - Structure of the data very simple, each item bounded size
  - Aggregators using mostly full-text search
- XML data transformation and integration on mobile devices — Mobile devices
  - Small XML messages
  - Transformation or aggregation queries
  - Caching is important
  - Streaming very important

48

“Where do XML and XQuery fit in my Web 2.0 application architecture ?”

In theory everywhere.

In practice, my most feared question.

My honest answer: “nowhere without you paying a large price.”

The Russian old man and his beard.

Significant changes will happen as result of this question:

- either XML and/or XQuery will *fail* or
- the existing architectures will have to *change* (***)

What is missing from a complete XML picture ?

- Missing technical pieces:
  - At the XML data model/representation level
  - At the XML Schema level
  - At the XML processing level (XQuery)
  - At the protocol level
- Requires changing the overall architecture

Changes in the XML Data Model

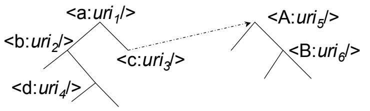
- Make the XQuery Data Model an *XML first class citizen* (must)
  - APIs in various programming languages
  - Support in Binary XML
- Make XML be a *graph*, not a tree (*) (must)
  - XML cannot be the primary information model until this happens
- Integrate the XML Data Model with RDF
- Deprecate the document nodes (nothing but calories)

XML: graph, not tree

- HealthCare Level 7 lesson. XBRL lesson. Etc.
- Any information model needs an E/R model
- An E/R model is by definition cyclic
- We need *native references* in XML
- “*Hack*” solutions; no *global* and *standard* solution
- Possible (simplistic) solution:
  1) Define *xs:ref* be a subtype of *xs:anyUri* (XML Schema)
  2) Nodes have node identifiers in XML Data Model; make those node *identifiers* be exposed externally as values of *xs:ref* (XML Data Model)
  3) Support *ref()* and *deref()* operations (XQuery and XSLT)

XML and RDF

- *“When should I use XML and when should I use RDF ?”* -- does it sound familiar !!?
- *“XML is for syntax; RDF is for semantics.”*
- *“XML is for data; RDF is for the metadata.”*
- Our community must work harder at the integration of XML and RDF.



```
<a xml:ref="uri1">
    <c>uri5</c/>
…..
</a>
<A xml:ref="uri5">
….
</A>
```

Supporting RDF links directly the XML Data Model.

Extensions to XML Schemas

- Integrate references into XML Schemas (must)
- Add integrity constraints and flexible structures definitions (must)
  - The “*flexible schemas*” community is waiting
- Integrate with XML Forms
  - Same here
- Embed code/behavior into schemas (must)
  - The “*metadata*” community is waiting
- Automatic support for “historical” data
- Deprecate xsi:nil (another bunch of calories)

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

Extending XML processing capabilities (I.e. XQuery)

- More processing power and user facilities (group-by, outer-joins, etc)
- Full text and updates (work already in progress)
- Error handling: try/catch
- Assertions (*)
- Continuous queries (*)
- Better integration with XSLT (*)
- Integration with Web Services (*)
- Integration with Semantic Search and ontologies (*)
- Procedural logic (*)
- Metadata extraction functions
- *eval*(XML-code)
- Second order functions

XQuery Full Text

- XML is primarily about text and markup
- XQuery Full Text extension provides search capabilities
- Use case example: RSS/blogs filtering
- FTSelections: special kind of Boolean predicates
  - Operators
    - words, and, or, not, mild not, order, scope, distance, window, times)
  - Match options
    - Case, diacritics, stemming, thesauri, stop words, language, wildcards
- Scoring

XQuery Full Text Example

```
for $book in
  doc("http://bstore1.example.com/full-
  text.xml")/books/book
for $section score $s := $book/section[.
  ftcontains "improving" && "usability"
  distance at most 2 words ordered at
  start]
where count($section/subsection)>0
return $section/title
```

XML Update facility

- NEW: XML Update Facility Working Draft
- Ability to do side effects (e.g. modify nodes in an XDM instance) in a declarative fashion
- Primitive update operations
  - insert <age>24</age> into $person[name=“Jim”]
  - delete $book[@year<2000]
  - rename $article as “publication”
  - replace ($books/book)[1] with <book>….</book>
  - replace value of $title with “New Title”

XQuery Update Facility (2)

- Conditional updates

  if($book/year<2000)

  then delete $book/year

  else rename $book/year as “publicationTime”
- Collection-oriented updates

  for $x in $book

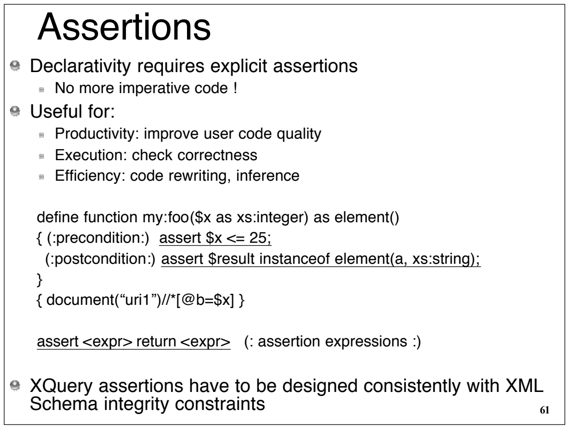
  where $x/year<200

  do rename $x as “oldBook”
- XML transformations using the update syntax

Extending XML processing capabilities (I.e. XQuery)

- More processing power and user facilities (group-by, outer-joins, etc)
- Full text and updates (work already in progress)
- Error handling: try/catch
- Assertions (*)
- Continuous queries (*)
- Better integration with XSLT (*)
- Integration with Web Services (*)
- Integration with Semantic Search and ontologies (*)
- Procedural logic (*)
- Metadata extraction functions
- *eval*(XML-code)
- Second order functions

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

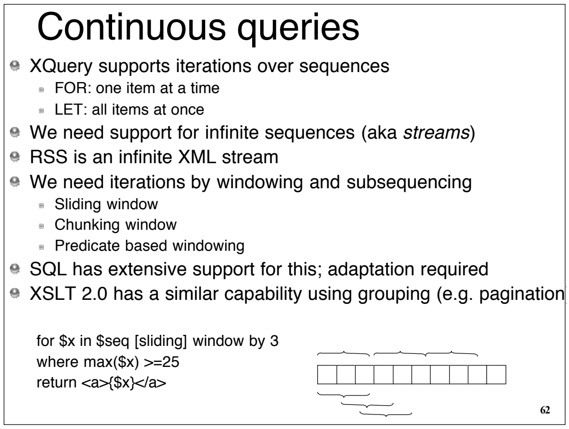


Assertions

- Declarativity requires explicit assertions
  - No more imperative code !
- Useful for:
  - Productivity: improve user code quality
  - Execution: check correctness
  - Efficiency: code rewriting, inference

```
define function my:foo($x as xs:integer) as element()
{ (:precondition:)  assert $x <= 25;
  (:postcondition:) assert $result instanceof element(a, xs:string);
}
{ document("uri1")//*[@b=$x] }

assert <expr> return <expr>   (: assertion expressions :)
```

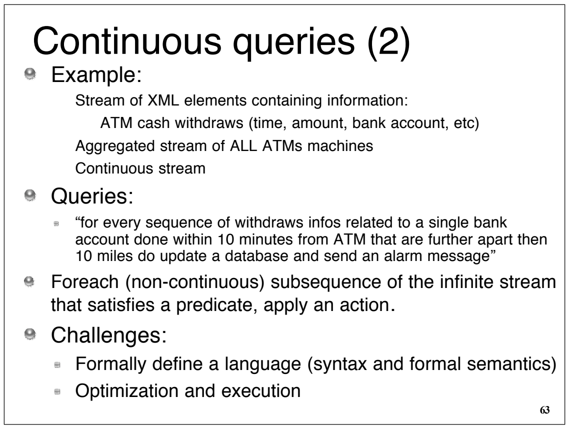
- XQuery assertions have to be designed consistently with XML Schema integrity constraints



Continuous queries

- XQuery supports iterations over sequences
  - FOR: one item at a time
  - LET: all items at once
- We need support for infinite sequences (aka *streams*)
- RSS is an infinite XML stream
- We need iterations by windowing and subsequencing
  - Sliding window
  - Chunking window
  - Predicate based windowing
- SQL has extensive support for this; adaptation required
- XSLT 2.0 has a similar capability using grouping (e.g. pagination

```
for $x in $seq [sliding] window by 3
where max($x) >=25
return <a>{$x}</a>
```



Continuous queries (2)

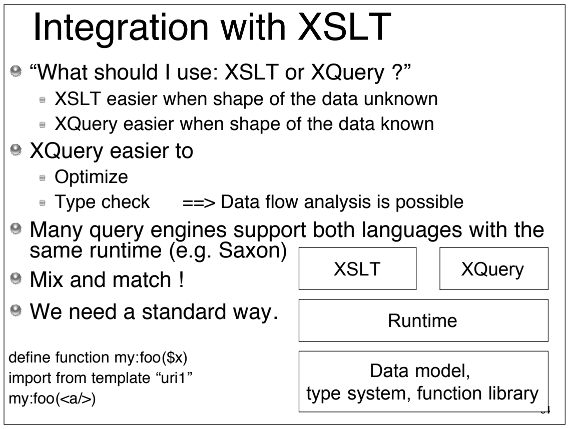
- Example:

  Stream of XML elements containing information:

  ATM cash withdraws (time, amount, bank account, etc)

  Aggregated stream of ALL ATMs machines

  Continuous stream
- Queries:
  - "for every sequence of withdraws infos related to a single bank account done within 10 minutes from ATM that are further apart then 10 miles do update a database and send an alarm message"
- Foreach (non-continuous) subsequence of the infinite stream that satisfies a predicate, apply an action.
- Challenges:
  - Formally define a language (syntax and formal semantics)
  - Optimization and execution

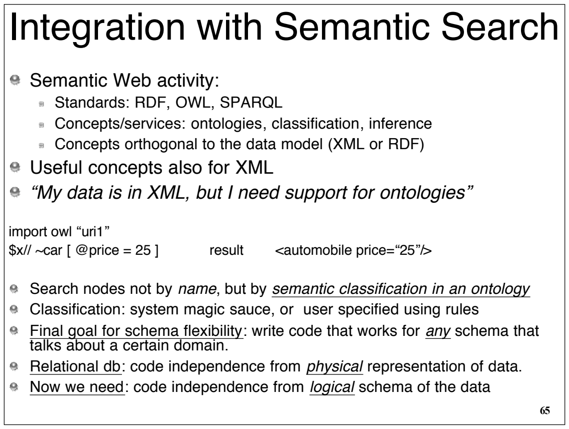


Integration with XSLT

- "What should I use: XSLT or XQuery ?"
  - XSLT easier when shape of the data unknown
  - XQuery easier when shape of the data known
- XQuery easier to
  - Optimize
  - Type check   ==> Data flow analysis is possible
- Many query engines support both languages with the same runtime (e.g. Saxon)
- Mix and match !
- We need a standard way.

XSLT
XQuery
Runtime
Data model,
type system, function library

```
define function my:foo($x)
import from template "uri1"
my:foo(<a/>)
```

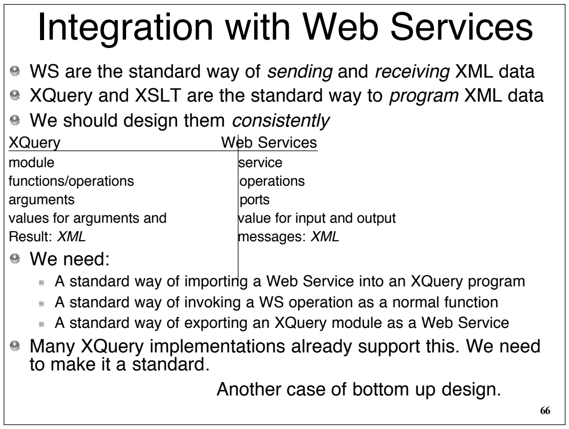


Integration with Semantic Search

- Semantic Web activity:
  - Standards: RDF, OWL, SPARQL
  - Concepts/services: ontologies, classification, inference
  - Concepts orthogonal to the data model (XML or RDF)
- Useful concepts also for XML
- *"My data is in XML, but I need support for ontologies"*

```
import owl "uri1"
$x// ~car [ @price = 25 ]          result       <automobile price="25"/>
```

- Search nodes not by *name*, but by *semantic classification in an ontology*
- Classification: system magic sauce, or user specified using rules
- Final goal for schema flexibility: write code that works for *any* schema that talks about a certain domain.
- Relational db: code independence from *physical* representation of data.
- Now we need: code independence from *logical* schema of the data



Integration with Web Services

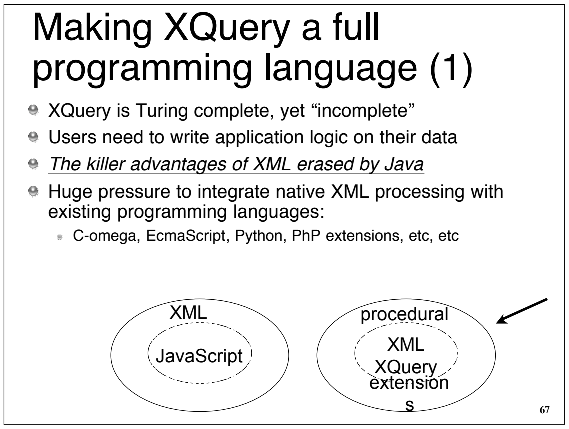
- WS are the standard way of *sending* and *receiving* XML data
- XQuery and XSLT are the standard way to *program* XML data
- We should design them *consistently*

XQuery	Web Services
module	service
functions/operations	operations
arguments	ports
values for arguments and Result: *XML*	value for input and output messages: *XML*

- We need:
  - A standard way of importing a Web Service into an XQuery program
  - A standard way of invoking a WS operation as a normal function
  - A standard way of exporting an XQuery module as a Web Service
- Many XQuery implementations already support this. We need to make it a standard.

Another case of bottom up design.

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.

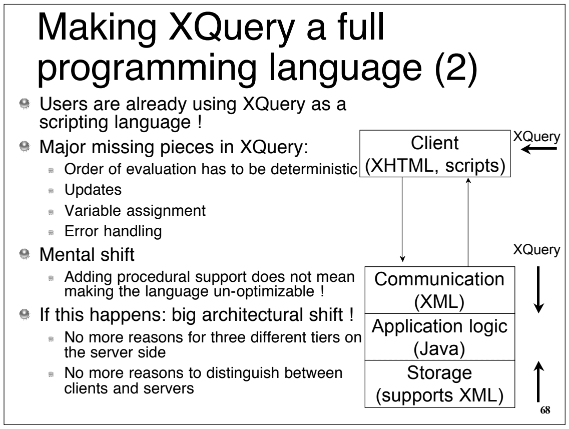


Making XQuery a full programming language (1)

- XQuery is Turing complete, yet "incomplete"
- Users need to write application logic on their data
- *The killer advantages of XML erased by Java*
- Huge pressure to integrate native XML processing with existing programming languages:
  - C-omega, EcmaScript, Python, PhP extensions, etc, etc

XML
JavaScript
procedural
XML
XQuery
extension
s

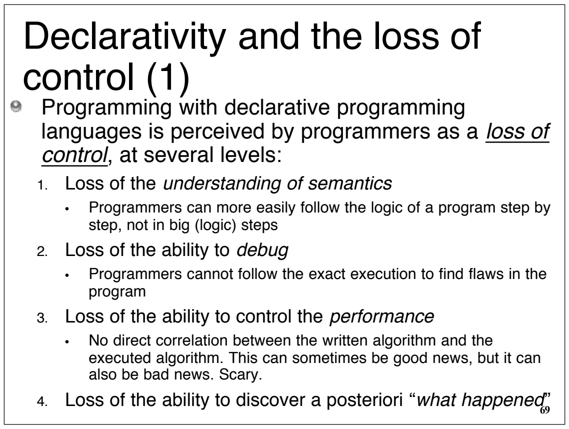
67



Making XQuery a full programming language (2)

- Users are already using XQuery as a scripting language !
- Major missing pieces in XQuery:
  - Order of evaluation has to be deterministic
  - Updates
  - Variable assignment
  - Error handling
- Mental shift
  - Adding procedural support does not mean making the language un-optimizable !
- If this happens: big architectural shift !
  - No more reasons for three different tiers on the server side
  - No more reasons to distinguish between clients and servers

68

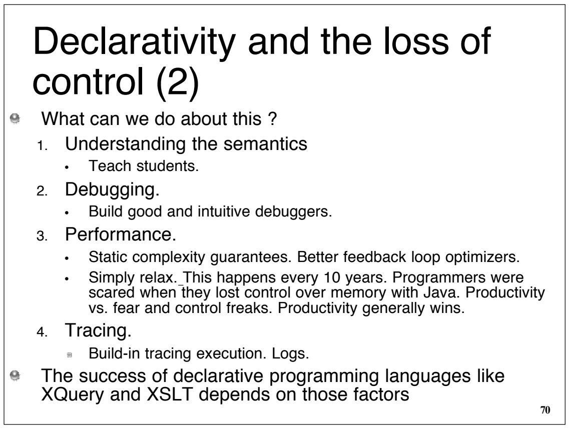


Declarativity and the loss of control (1)

- Programming with declarative programming languages is perceived by programmers as a *loss of control*, at several levels:

1. Loss of the *understanding of semantics*
   - Programmers can more easily follow the logic of a program step by step, not in big (logic) steps
2. Loss of the ability to *debug*
   - Programmers cannot follow the exact execution to find flaws in the program
3. Loss of the ability to control the *performance*
   - No direct correlation between the written algorithm and the executed algorithm. This can sometimes be good news, but it can also be bad news. Scary.
4. Loss of the ability to discover a posteriori "*what happened*"

69



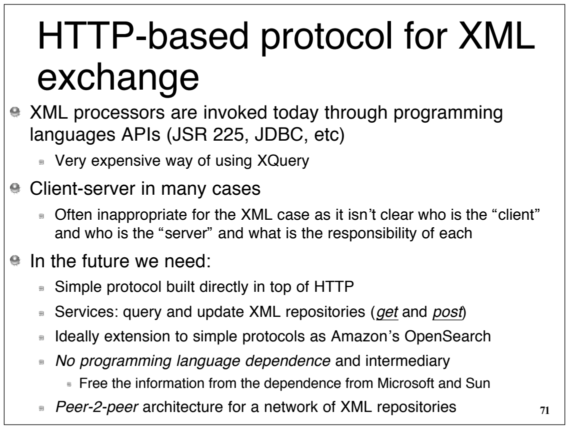
Declarativity and the loss of control (2)

- What can we do about this ?

1. Understanding the semantics
   - Teach students.
2. Debugging.
   - Build good and intuitive debuggers.
3. Performance.
   - Static complexity guarantees. Better feedback loop optimizers.
   - Simply relax. This happens every 10 years. Programmers were scared when they lost control over memory with Java. Productivity vs. fear and control freaks. Productivity generally wins.
4. Tracing.
   - Build-in tracing execution. Logs.

- The success of declarative programming languages like XQuery and XSLT depends on those factors

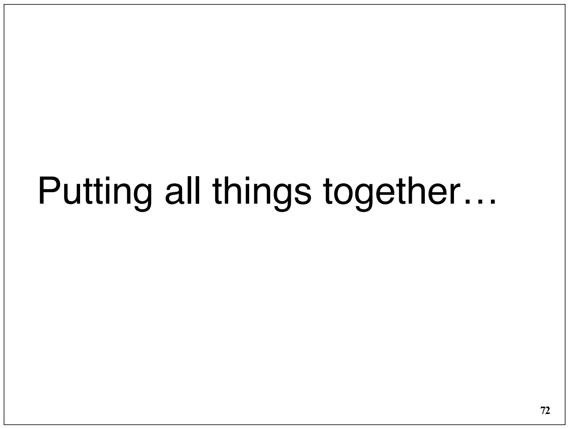
70



HTTP-based protocol for XML exchange

- XML processors are invoked today through programming languages APIs (JSR 225, JDBC, etc)
  - Very expensive way of using XQuery
- Client-server in many cases
  - Often inappropriate for the XML case as it isn't clear who is the "client" and who is the "server" and what is the responsibility of each
- In the future we need:
  - Simple protocol built directly in top of HTTP
  - Services: query and update XML repositories (*get* and *post*)
  - Ideally extension to simple protocols as Amazon's OpenSearch
  - *No programming language dependence* and intermediary
    - Free the information from the dependence from Microsoft and Sun
  - *Peer-2-peer* architecture for a network of XML repositories

71

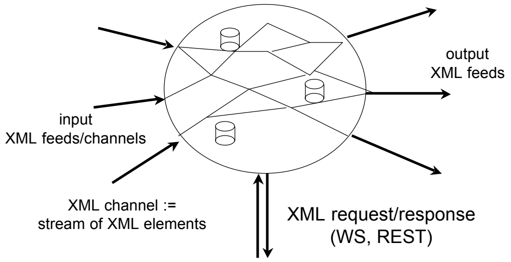


Putting all things together…

72

The XML information hub

Dataflow architecture: *channels* and *actors*
Information modeled only as XML through the hub
Declarative specification of actors, rules



XML information hub: the data

- Channels
  - Contain infinite streams of XML nodes
  - Append only
  - Communication with the external world, but also internal
- Stores
  - Contain finite collections of XML nodes
  - Updatable (XQuery updates)
- Both:
  - Identified by a URI
  - Can be constructed or deleted programmatically
  - Can be constraint by XML schema and/or integrity constraints
  - All changes are automatically archived (100% logging)
  - Modifications (appends, updates) are not necessarily transacted

XML information hub: the code

- Actors
  - Snippets of XQuery code (extended with all the goodies: updates, XSLT, ontologies, full text, procedural logic, etc)
  - Independent of each other
  - Can access all channels and stores (modulo security)
  - Describe "How to react when a certain event happens."
  - Get invoked when an event happens:
    - A new entry in a channel
    - A state change of interest in a certain store
- Allow for organic growth of the code
- Declarative in nature
- No global orchestration
- Raising errors => sending an entry on an error channe

XML information hub: the constraints

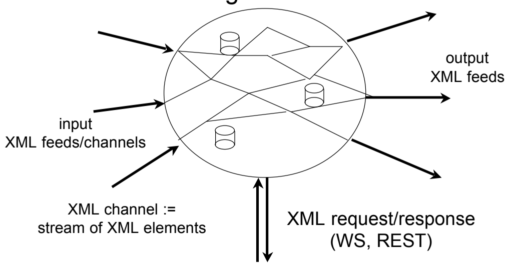
- Assertions:
  - Linked to a channel or store
  - Global
  - Snippets of XQuery code (boolean)
- Guaranteed by the system to be satisfied
- Declarative specification

XML information hub

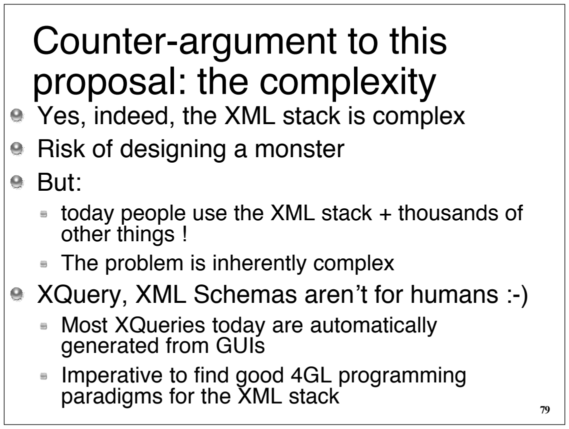
- Channels
- Stores
- Actors
- Constraints
- All code is XQuery++

The XML information hub: the execution

- How to execute it in a cluster of machines?
- Automatic scaling ?



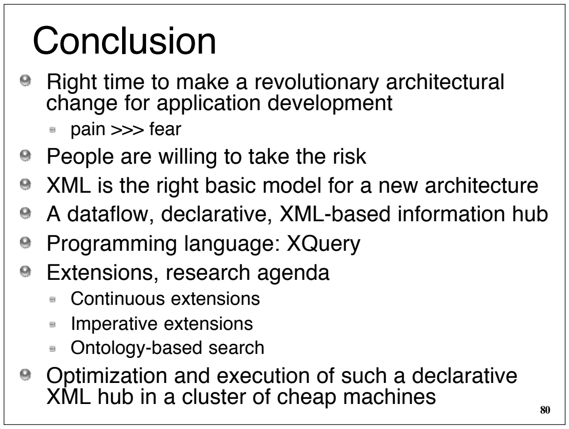
This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.



Counter-argument to this proposal: the complexity

- Yes, indeed, the XML stack is complex
- Risk of designing a monster
- But:
  - today people use the XML stack + thousands of other things !
  - The problem is inherently complex
- XQuery, XML Schemas aren’t for humans :-)
  - Most XQueries today are automatically generated from GUIs
  - Imperative to find good 4GL programming paradigms for the XML stack

79



Conclusion

- Right time to make a revolutionary architectural change for application development
  - pain >>> fear
- People are willing to take the risk
- XML is the right basic model for a new architecture
- A dataflow, declarative, XML-based information hub
- Programming language: XQuery
- Extensions, research agenda
  - Continuous extensions
  - Imperative extensions
  - Ontology-based search
- Optimization and execution of such a declarative XML hub in a cluster of cheap machines

80

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.