

Moving Toward Web-Based Distributed Systems with XML

By Grigory Nudelman

A) Introduction

The idea of building effective Web-based distributed systems has been around for a long time. However, much of the efforts have been unsuccessful, largely due to the loose and fuzzy nature of the web architecture, and because HTML and proprietary formats, the two most common web communication mechanisms, do not lend themselves well to sharing data across time, space and communities. Recently, however, a fundamentally new standard of web communication, eXtensible Markup Language (XML), has come into being. XML is human and machine readable, self-validating, flexible and object-oriented. This paper highlights the challenges of adopting the HTML and proprietary formats for DS communication using a loan exchange B2B company and examines how XML solves these challenges making it a superior tool for Web DS integration.

XML is much better suited for communication between the distributed system components and it has enabled collaborative, flexible data sharing solutions on the web. Furthermore, due to XML's ability to validate web-based communications documents and discover and effortlessly translate information between various formats, XML-based communication protocols such as SOAP make possible real-world web-based Remote Procedure Calls of a true Distributed System. This is similar to a Java RMI, but is language and platform independent. Thus, XML through the power of SOAP, enables a true Distributed system architecture on the web: a new paradigm called Web Service, which allows a much higher level of automated, object/language/platform-independent integration possible.

B) Distributed System on the Web: Loan Exchange B2B Company

To highlight the challenges of distributed systems on the web, we will examine a sample loan exchange company which brings together brokers and lenders for the purpose of funding individual loans. Broker Loan object is fairly complex. It involves over 200 attributes, as well as complex borrower objects, credit report objects and other pertinent information which the loan package must contain. Most of the brokers use Loan Originating Software (LOS) to help collect this information from the borrowers and validate it against any errors.

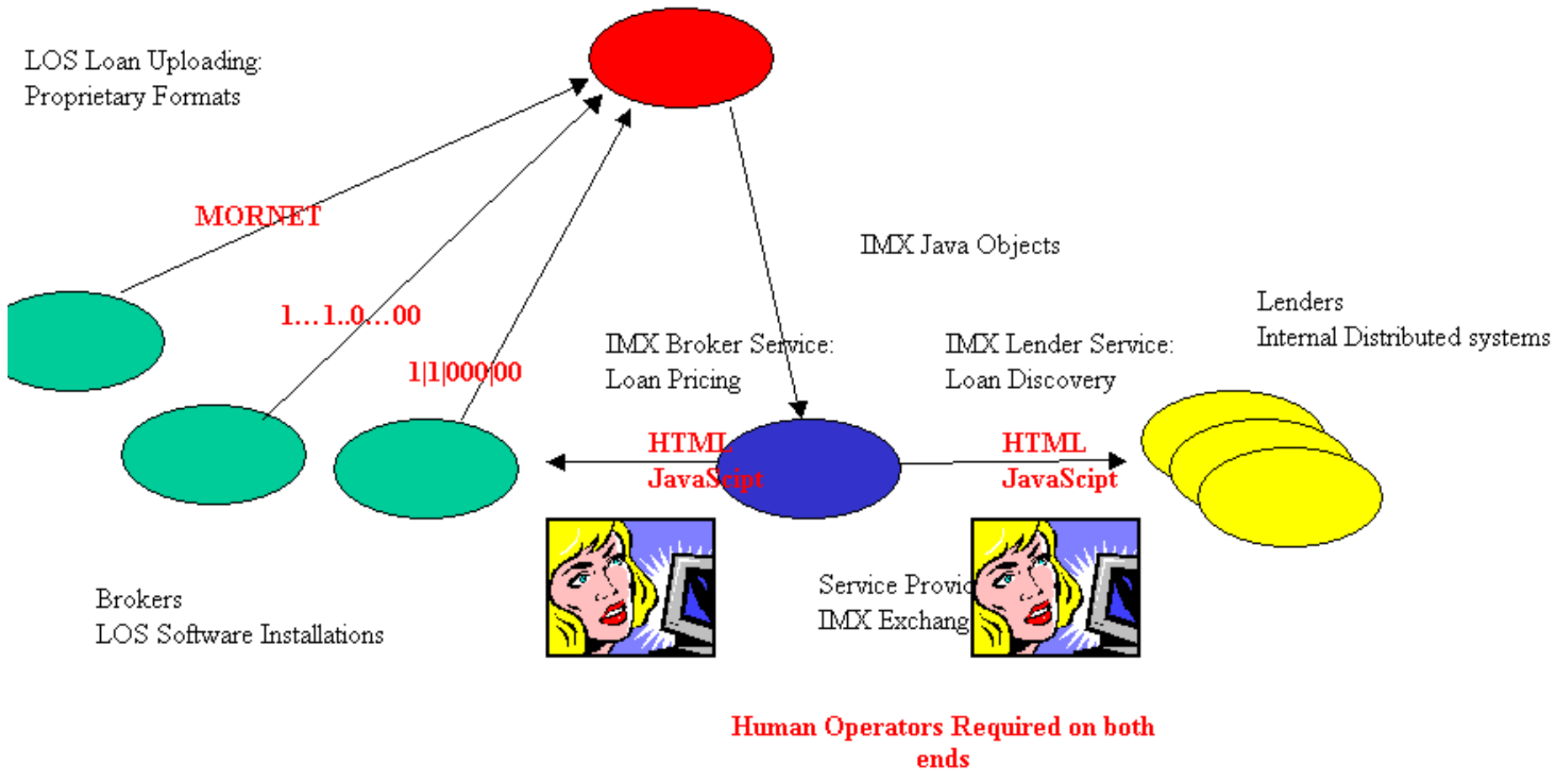
After the loan information has been entered into the LOS, the brokers have to manually search the lender program guidelines (published on the web as HTML) make note of exceptions and if the loan passes them, apply the lender's price adjustments to the published rate sheets to finally obtain the rate and points for the loan. After the loan price has been obtained using this tedious process, Broker uses the LOS software to print out the forms which make up the complex loan object. These forms are then faxed to the Lender whose program the loan qualifies for and the information contained in the form is again manually entered into the Lender's legacy system,

and the pricing process is again repeated by a human operator to determine if the loan qualifies for the lender program and what price will the loan receive. After the price is determined the lock sheet is faxed back to the Broker and the process is thus completed.

It is obvious that many of these steps can be much improved upon by tying all of the components into a distributed system of some sort. IMX Exchange attempted to become one such system based on the web. (90) However, it soon became apparent that all of the LOS software packages save the loan files in their own proprietary formats that change with every release of the software. Thus the Brokers were forced to enter the loan twice: once into the LOS and once into the IMX's own web-based forms.

After the loan is thus posted, a price can be obtained immediately, based on the lender guidelines that are stored on the IMX's database. The Brokers save a lot of time not having to hunt for the best price, and that is an obvious improvement. However, once the price is found and the loan is locked, the pricing sheets from IMX have to be printed or saved as HTML and need to be faxed or emailed over to the lender's office, where they are again entered into the lender's legacy system. So, even though IMX provides a valuable pricing service, the service is incomplete, because it consists of 3 separate systems that have no easy way to share the data with each other:

Un-maintainable IMX Adapter Kludge



Having to change one of the loan attributes is also a nightmare: LOS, IMX and Lender's files need again be brought to mutual correctness by means of tedious triple data-entry and a flurry of printing and faxing. It would be significantly more convenient to be able to propagate the changes over the distributed system from a single

terminal. For this, a reliable middle-layer web format must be found to share the data across time, space and communities.

To share the data across time, this middle-layer format must be flexible enough to accommodate future changes and improvements, the addition and deletion of attributes and changes of schema in the objects. In addition and to share the data effectively across the space of many different systems, this format must be self-validating, and readable by both machines and humans. And finally, to be able to speak a very different language of the lender's and broker's systems, the format must be self-describing and organically developed between interested parties (communities) with minimum involvement of any central governing body.

Until recently, the only middle-layer connection protocols available to a web-based exchange like IMX were HTML and proprietary text and binary formats. Both of these have been used in the past, but the results were discouraging, because both formats are either not human or not machine readable, non-validating, brittle and non-future proof. However, the new XML standard is free of these shortcomings, and can be used to extend this rudimentary human-enabled framework into a true, automated Web-Base Distributed System (WBDS) as a Web Service with the use of SOAP XML protocol.

B) Current Web communication formats

Before the coming of XML and even now, most of the web-based systems operate on 2 major standards, HTML and the proprietary text and binary standards. Here we will examine them in an effort to discover their weaknesses as Distributed Systems communication standards.

1) HTML

HTML format carries a mixture of tags describing contents and tags determining the page's display properties. Although HTML is human-readable and flexible in presentation, the absence of any meaningful content meta-data in the markup (tags) of the page make it non-validating and not computer-readable. Freezing a small set of tags limits the HTML's usefulness as the middle tier communication format in several important respects, the chief among them extensibility, structure and validation. (24)

One is the most familiar web representation format, Hyper Text Markup Language (HTML), suffers from many weaknesses when it is examined in the light of the communication standard. HTML was initially designed by Tim Berners-Lee as a markup standard for sharing scientific papers. To mark up the document was to describe its structure using metadata in the tag format. Thus, the original HTML tags were designed to describe contents of the document, not to control the presentation of the page to the user. (1)

Unfortunately for HTML, when "Marc Andreessen came up with the idea of the tag, and the [HTML-based] Web was both born and destroyed at that moment." (1) because the structure and content become mixed in

an HTML page. From that point on, the early web designers like David Siegel began using the 1-pixel spacers and <TABLE> tags to align the pages for presentation and increase the pages visual appeal, irregardless of the content. Thus mixing structure with presentation resulted in irretrievable loss of the original a well-structured document format.(2) From that time, HTML has evolved mostly for delivering pages that present information to the human user on the User Agent (web browser like Netscape or Internet Explorer).

HTML pages that are "too" human readable and non-validating, resulted in fuzziness of format that a computer can not reliably understand. (24) For example, if the user can compare 2 movie reviews from the different websites with an apparent ease, even the most flexible and powerful search engine software application with thousands of lines of code (like AltaVista or Google) can not perform this simple task reliably, because the description of the page (metadata) can not be easily inferred from the presentation-mixed format. HTML does not support data schemas or hierarchies, thus the best most engines can hope for is stripping all the HTML tags (and thus losing all the metadata) and trying to infer the meaning of the page from the resulting text file without any metadata.(24)

To make the matters worse, the two major makers of User Agents, Microsoft and Netscape, have declared a virtual tag war on each other. Between the two companies' products only 80% of the tags are shared, whereas the rest give results unpredictable at best.(3) This incompatibility is often called the 80/20 rule, which forces some sophisticated website pages to have to "snoop" the user agent and deliver the HTML code customized for a particular browser. In this sense, many web pages have become flashy, semi-proprietary creations of individual companies which would cause problems when displayed on any other browser version except the one they are specifically targeted for.(30) Needless to say this wrecks havoc if we try to deploy the HTML as the middle layer in this mish-mash.

In contrast, XML has always been intended to allow the easy and rapid construction of custom tag-sets specific to corporation, scientific discipline or a web community. While every individual can choose to define their own tag sets, a particular strength of XML is the sharing of these tag sets and vocabularies, like Chemical Markup Language (CML). (31) XML vocabularies provide a more easily searchable documents and databases and a way to exchange information easily between many different organizations and computer applications.(30)

HTML by itself is a bad enough situation, however, to make things worse, it can incorporate sophisticated scripting capabilities as well, with the use of JavaScript and VBScript. This allows the designers to add tremendous power and interactivity to the web pages, but it make the job of tying these pages as a middle layer incredibly difficult, as much of the content and metadata itself is "document written" using scripting languages, making it impossible for the parsers to divine the meaning from this gibberish. Thus, by using HTML for the middle-tier communication between the web distributed system components, we restrict our server code to use with human operators sitting in front of the browser. (28)

In the case of IMX, the loan information is contained in IMX's HTML pages and forms. Given the modern nature of the web, and the growing nature of IMX's enterprise, these forms have a very short life, in the order of weeks. If the Lender were to invest capital into building the bridge between IMX's HTML pages and Lender's legacy systems, that would hardly be the capital well spent, as All the spacing of the HTML tags and the markup itself

would change forcing us to discard the old standard. Clearly, HTML is non-validating, non-machine readable and not future proof. It leaves a lot to be desired as a middle-layer format for Web Distributed Systems.

2) Proprietary Binary and Text-Serialized Formats

Binary, text-serialized with delimited special characters (~,^,|) , and text-serialized by character spacing (MORNET (91) format for loans) also suffer from being non-validating and are very hard to debug as the format is often entirely non-human readable. They are also incredibly brittle: adding a single attribute to the special-character delimited stream of 300-400 attributes is almost guaranteed to break the system and read in gibberish unless all of the systems components are upgraded at the same time.(15)

The same problem as HTML, complete lack of content-describing metadata, will cause wrong strings to be read at runtime. Java Applets and ActiveX runtime parsing errors are hard to debug, producing nightmarish black-box legacy systems without any documentation. Legacy systems often communicate using these formats, because they were developed using the in-house tools and expertise, maintained by programmers that invested a lifetime of format patches, hacks and bug-fixes into the system, making no two exactly alike and being unable to provide a reliable bridging mechanism between these systems.

For a distributed system, it would be more helpful if the information is in the form that can be reused in many different ways. (13) For this to be possible, every party that wishes to use the proprietary format, they must build very expensive, slow and brittle software adapters, which are only good until the next release of the format by the party that invented that standard. The control of this standard resides in private hands, and thus can be changed or even withdrawn without notice.(13)

In the case of IMX, the binary and text-delimited standard make it impossible for IMX to extract information reliably from the LOS software's proprietary storage formats as these change with every new release of the software and IMX has no way to know which version is currently running on which of our several thousand Broker customers. This brittle and non-validating nature of the proprietary standard make it necessary for the Brokers to enter, validate and maintain two copies of the same information.

In addition, proprietary formats are often protected by a slew of licenses, copyrights, patents and intellectual property rights restrictions. (12) This makes them hard to share, especially among competition. XML on the other hand is completely free and unencumbered by these restrictions.

The challenge of interoperability has grown exponentially with the number of systems. The most challenging aspect of this problem is the ability to exchange information services.(14) The system's language defines how it is constructed and used, and different proprietary standards can not connect with one another easily. This "proprietary patented standard" strategy may have worked well in the past, as a strategy for locking your customers into your format, but in this widely distributed world, exclusivity is hardly a goal one can still attempt to pursue. To be effective in today's distributed environment a wide variety of dissimilar information systems, developed independently by different software engineering groups across countries, time and

communities, must share timely, accurate, complete information in a form that is mutually understandable.(14)
This goal is clearly not met by the use of any proprietary standard.

Thus, the legacy binary and text formats suffer from being brittle from any lack of schema and validation, and also from the difficulty of being non-human readable and thus difficult to debug. They are not at all future-proof against upgrades and handle any changes very poorly, resulting in unpredictable system behavior. Thus any proprietary standard is a bad choice for a middle-layer of a Web-based DS.

C) The new emerging XML standard

In recent years, a new standard, XML, has been developed by the W3C to "address the requirements of commercial web publishing and to enable further expansion of web technology into new domains of distributed document processing." (24) XML holds the promise to improve the communication between the distributed systems by removing the shortcomings of HTML and proprietary formats, that we have so far discussed. XML provides a powerful information technology. More specifically, it provides an "interpreted, flexible means of transmitting not only data but also its semantics and structure" (35) XML aims to have as many as possible of the following attributes:

Human-readable,

Machine-readable,

Object-oriented, hierarchical, and meta-document object model,

Self-validating,

Must allow for easy query of the document elements and for easy transformation,

Extensible and graceful to upgrade,

Future-proof and organically evolving. (14) (10)

XML has all these superior qualities, making it a natural choice as the next communication standard of choice. More importantly, the attribute of being easily machine readable in a standardized way (using SOAP) allows XML to make direct RPC calls to a Web Service, providing true integration for the WBDS.

1) XML has superior qualities as communication standard of choice.

For a format to be both machine- and human-readable is quite a challenge. Human logic is often fussy and

language has many permutations, so format must allow for this. It must also maintain a rigid structure to be understood by computer parsers and generators. Fortunately, XML takes care of both of these seemingly conflicting requirements gracefully. The key is self-description by use of generic meta-data. Like HTML, XML is a mark up language which relies on nested tags and tag attributes to represent data and data structure. For example, the property address of a loan in our B2B exchange can be represented as follows:

```
<loan>
  <property>
    <address>
      <street>
        10 Maple Drive
      </street>
      <city>
        Springfield
      </city>
      <state>
        IL
      </state>
    </address>
  </property>
</loan>
```

As you can see, this is quite human-readable, we did not have to tell you it is a loan's property address, as this information is already included in the document. This format is also machine readable through two major types of parsers: even-driven parser (SAX) and tree-based parser (DOM). Because the tags in the well-formed document explain their contents so precisely, this information can be used by the machine to discover the document's meaning and use it elsewhere.

This format is also object oriented, hierarchical, and adheres to meta-document object model. Object orientation is enforced through the hierarchy of nested meta-date tags:

We know for example that loan is an object, of which property is an attribute. Property itself is a composite object: it contains an address object, where city, street and state are simple objects or text attributes (see second example below).

We also mentioned that XML allows for some of the human fussiness, which imparts it additional flexibility. Compare previous document's XML format with this, also perfectly valid representation of the same data:

```
<loan>
  <property>
    <address street="10 Maple Drive" city="Springfield" state="IL" />
  </property>
</loan>
```

The same data which was previously contained in several simple objects, like <street> has now become a direct part of the attributes of the address object. Both XML documents are perfectly valid from human standpoint, but a change like that would mean a complete disaster for a machine translators if it were HTML or a proprietary standard. However, because the data is in XML, DOM and SAX parsers would have no problem interpreting this new data format, because that tag metadata still imparts firm, specific structural meaning to the document's contents. But how do we know that this new format is valid for our application? How do we provide this fussiness and flexibility to our format? The answer is, of course, automatic validation.

Validation specification is "patterns expressed in some formal way, in particular for use by the program... the pre- and post-conditions we want to assert about the document's structures ... it clarifies a programmers tasks and capabilities and nature of data." (17) Validation is the key to usefulness of XML, because it imparts meaning to a particular piece of data and rigorously enforces the format for the purpose of sharing the data with other systems. (18) Currently there are two major validation formats, Document Type Definition (DTD) and XML Schema. Only DTD is accepted by the W3C.(22)

DTD is truly a validation specification -- it describes the number and the nature of attributes and elements and their nesting pattern. However, DTDs have many shortcomings: they are not re-usable or extensible. They must be created purely by hand and can not be automatically inferred from the related XML. Also DTD is a stand-

alone entity, in a format all its own, and thus must be treated and validated differently from the XML data. For large and complicated documents, DTDs get quite large and they can not be split into pieces which would describe portions of the main document. (19) Also, very importantly, it does not impart data type meaning (such as integer, double, text, month, day, 4-digit year, etc.) to the elements, merely providing us with a convenient notation.(19)

XML Schema, on the other hand, is a still emerging format, supported by several major software vendors including Microsoft.(100) Schema is a collection of rules about a document's structures, which also includes validation and information about meaning of elements. (17) Schema has the ability to specify: "this element is day; this element is month; this element is a 4-digit year" whereas a validation language like DTD concentrates on lexical/structural issues: "this element must conform to regular expression /nnnn-nn-nn". So schema contains somewhat different validation information which also carries additional meta-data about the element's data type. As such schema can provide a more rigorous validation than the DTD, but more importantly, it can be used to both constrain and explain the document. (20)

The current draft of the W3C XML Schema proposal states: "The purpose of a ... schema is to define and describe a class of XML documents by using these [markup] constructs to constrain and document the meaning, usage and relationships of their constituent parts: datatypes, elements, and their content, attributes and their content, entities and their contents and notations. Schema constructs may also provide for the specification of additional information such as default values. Schemas are intended to document their own meaning, usage, and function through a common documentation vocabulary. Thus, XML Schema Structures can be used to define, describe, and catalogue XML vocabularies for classes of XML documents.(22)

XML Schema is also free of other DTD restrictions, it is written in the original XML format and is thus XML compliant, which gives users a lot of freedom in creating automated Schemas and breaking up and re-using larger Schemas as a set of smaller specific validation rules.(21)

Very importantly, both DTD and XML Schema allows for easy versioning and graceful future upgrades in the data formats. This can be accomplished by either sending the schema or DTD inline, along with every XML document, or better yet, providing a set of validation templates available on the web, instantly accessible and versioned, so both a version of the format and rules themselves can be gracefully changed as the needs of the format evolve organically within a community.

Such ease of upgrade allows for the format to begin very simply and evolve quickly along with the needs of the people who use it. In the past, to insure the industry's use of a specific format, an industry association had to be involved in some way to appease the conflicting parties and come up with a format that meets the need of everyone involved. This lengthy process often takes years, and any upgrade to the format that has been finally developed takes months and cooperation of the entire industry to approve. These are difficult propositions, and XML can be used effectively to shorten the time for developing and maintaining a standard, entirely without a ruling body of any kind.

Several industries have chosen to form consortiums that define the new XML standards that everyone in that industry should abide by. One of these standards, Solution Exchange Standard (SES), markup language standard supported by the consortium of over 60 hardware, software and communication companies to facilitate the exchange of technical information on the web, is described with this announcement:

The standard has been designed to be flexible. It is independent of any platform, vendor or application, so it can be used to exchange solution information without regard to the system it is coming from or going to. [...] Additionally, the standard has been designed to have a long lifetime. SGML offers room for growth and extensibility, so the standard can easily accommodate rapidly changing support environments.(40)

In relevance to our Mortgage Industry B2B we must mention MISMO, the new XML standard developed by major Broker and Lender companies to replace the old brittle proprietary MORNET standard developed by Fannie Mae.(99) MISMO was developed by the Mortgage Industry Standards Maintenance Organization in association with Mortgage Bankers Association of America. (101)

Although XML-based MISMO is a much more flexible standard than MORNET, it still suffers from intransigency of being developed and approved by the central party, and thus time to develop, approve and release any changes to MISMO is still quite long. MISMO work began in 1998, with the release of the W3C XML 1.0 standard. Yet, it took till June 15, 2000 to get the MISMO 1.0 approved. (102) This is scarcely the way to go in a fast web-based environment in which B2B companies like IMX have to work. Thus, even though it is exciting and gratifying to see many industry consortiums come up with XML-based standards like MISMO and SES, some parts of which may be reusable, true power of XML-based communication comes from Web Services integration model, not from the top-down format standardization by an industry consul. In a Web Services model, XML standard is flexible, and can be changed on a day-to-day basis as different businesses open up their internal core mission-critical systems to collaboration in a SOAP-based XML format to form a true flexible WBDS.

Thus, in a Web Services model, interested companies and communities can come together purely on the need-to-basis and create XML standards that answer the exact needs of that group in just a matter of days. Software can also be created quickly to parse, translate and validate this XML data. If any part of the standard needs to change, this again can be accomplished quite quickly and painlessly, and even several versions of the same standard can exist without customary confusion. This capability allows not only the companies within a particular industry to exchange the format, but also a variety of heterogeneous agencies that can exchange their data in a "natural" way. "Let a thousand DTDs bloom" on the web is the goal of such an organic collaboration, because as business conditions and requirements change so can new XML standards be adopted quickly to meet these challenges.(10)

One of the important XML capabilities that makes this kind of community data sharing possible, is the ability to easily transform XML into HTML for viewing or into another XML document of a different format to be sent to another part of the distributed system. (17) This language is called eXtensible Stylesheet Language or XSL. With XSL, any XML document can be transformed into a different format in just several hundred lines of code, that can be written in just a few hours. Compare this to building and validating a C++ code adapter to a

proprietary data format, or an HTML search engine data parser!

XSL is one more step in insuring that the XML format is future proof. If a new data format is demanded by a business need or a change in the community, XSL can be easily deployed to transform the older XML format into the newer version. This version can also be sent in the request for the Web Service, thus completing the information cycle. Client asks for a specific format and the server can choose among several XSLs to return the document in an exact format the client has requested. If the version needed is not available, the client can deploy it's own XSL to transform the data into the exact format required for passing the data down the line to the next distributed system component. Thus XSL is an essential component in a web distributed system, something that HTML and proprietary formats can not provide with any degree of reliability without much time and expense.

Thus, we have shown that XML has evolved to overcome the shortcomings of the older web formats. XML has several important characteristics. It is:

- a) Human-readable: allows for fussy logic and growth.
- b) Machine-readable, accurate object presentation.
- c) Self-validating with DTD and newly developing XML Schema. Validation rules can be made available on the web for easy versioning.
- d) Extensible, for graceful upgrades in schema.
- e) Transformable, for easy change in formats and organically generated adapters.
- f) Future-proof, with organically evolving standards which change as the community grows and data requirements change.

2) XML-Based Web Services Using SOAP.

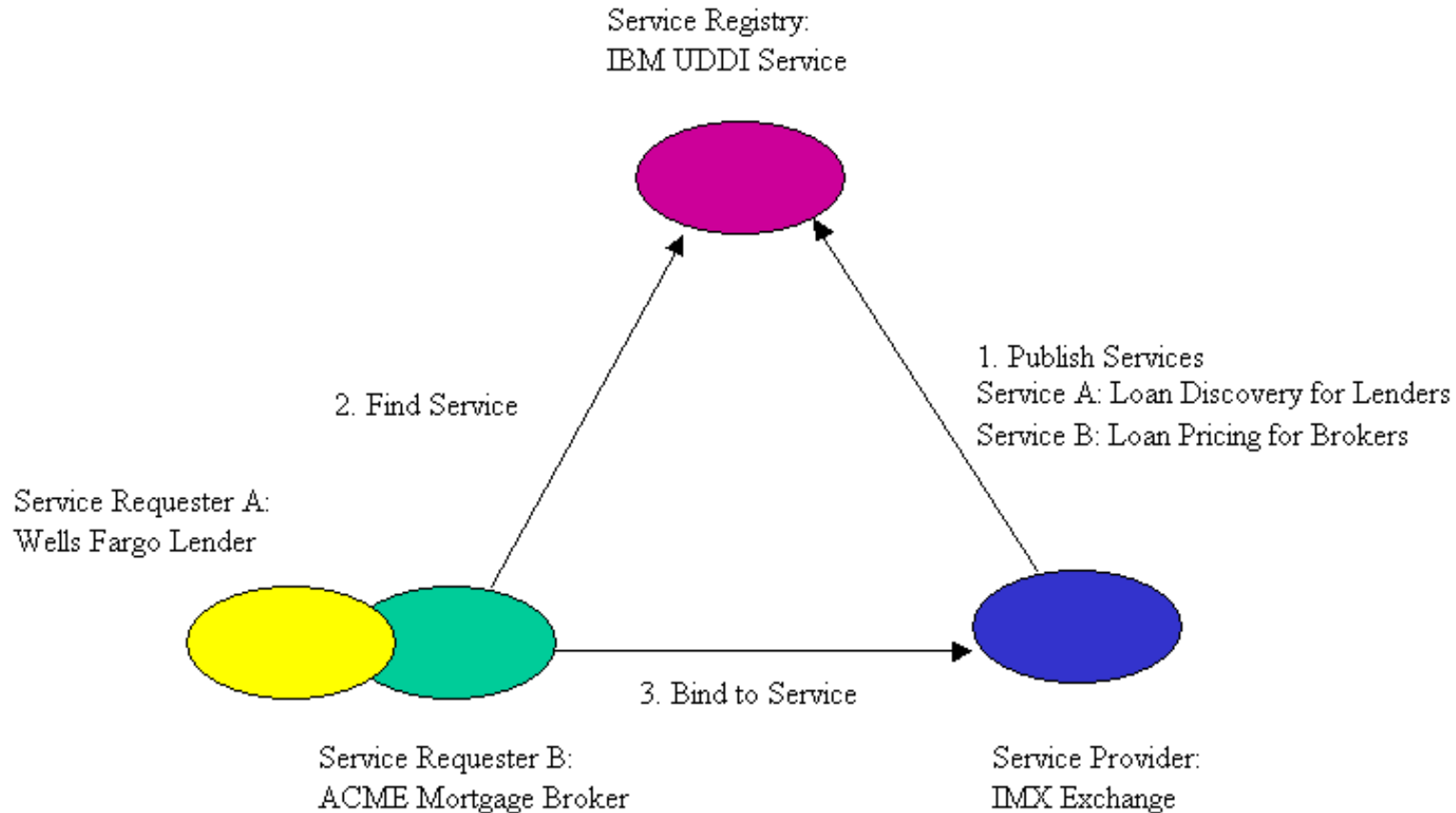
Web Service is any program that is callable by another program across the Web in a platform/language/object model neutral fashion using standardized Web Services protocols. (103) The Web Services protocols include Simple Object Access Protocol (SOAP); Universal Description Discovery and Integration (UDDI); and Web Services Description Language (WSDL); All these protocols are XML-based. Thus a Web Service to an application is what Web page is to a person, it allows one machine to automatically discover the format and information that the other machine has and allows it to get and utilize this information in a purely automatic fashion, without any need for human intervention.

SOAP, is at it's simplest level an XML-Based RPC Methodology. SOAP behaves in a similar fashion to the HTTP headers, the only difference is that SOAP describes the functionality that must occur as a result of the exchange. WSDL serves as a mechanism for creating endpoints, or destinations for the SOAP requests. WSDL

files define what a request should contain and the data types of each element in the request. WSDL file can be considered a stub, which defines the way application generating the stub works.(117)

UDDI is the equivalent of the Web Yellow Pages. Businesses such as IMX can register their web services in the UDDI business registry and, in doing so, let the partners and anyone else who knows and uses UDDI easily find the Web Services they need. (118) UDDI provides a standard place and a standard interface for Web Services query and storage.

The process for a typical Web Service is described below. It follows the basic publish-find-bind cycle. (121) In this example, the service provider publishes its service interface (a WSDL document) to a service registry (a UDDI Provider). A service requester searches the service registry and finds a web service that suits its needs. (Catalogs of reusable business collaboration definitions and forms is also available through this registry.) It then binds to the service provider dynamically at runtime in order to execute the service (using SOAP) (104):



One of the main benefits of a Web Service is the ease of integration. Instead of defining a proprietary standard and building adapters to transform them between different enterprise systems, through SOAP we have access to standardized exchange protocol, similar to an RMI remote procedure call but XML-based, and thus platform and language independent. Thus a Web Service can be used “out of the box” as part of the Web-Based Distributed System. Thus, a web service model provides a mechanism for solving the two key problems in distributed computing today: Open Interoperability, the requirement that heterogeneous systems communicate without a predefined infrastructure coupling; and Dynamic Computing, the requirement that once a technology is deployed, new opportunities that arise can be seamlessly engaged without reengineering. (115)

But, you can say, is using WSDL and SOAP not exactly the same as using IDL and CORBA? Technically, it is the same idea: XML Web Services as a technology is just an RPC mechanism that uses XML over HTTP to call specific methods and objects while using the UDDI registry for service lookup. The difference is that CORBA requires too much to be in place for displaying the core business functions over the web. SOAP on the other hand uses the easy to read standard that can penetrate the company’s firewall, can be understood by any hardware platform and very easy to deploy. (119) Thus Web Services, using the same paradigm, really deliver the promise of CORBA for today’s web-based distributed computing systems. (116)

Web Service is simply a piece of software that provides functionality over the web and relying on standard protocols such as SOAP to identify, call and receive results from that functionality. The promise of the Web Services foundations is that all clients written in all languages will have the same APIs to achieve the same end results. Once a web service has been deployed using WSDL, all the clients can implement this functionality. XML/SOAP’s ability to define a precise but flexible automated communication standard that can be altered for a payload of any kind, with sequencing of payloads in complex communications, supporting comprehensive security mechanism, make XML a key component to “distributed system ready” Web Services.

3) XML as a Middle-tier Web DS standard: B2B Example.

"The web can grow significantly in the power and scope if it is extended to support communication between applications, from one program to the other."(41) XML is a foundation for a web services framework within which automated, decentralized services can be defined, deployed, manipulated and evolved in an automated fashion.(42)

If we now examine how our B2B loan exchange can benefit from using XML as the middle tier in the web distributed system. First, we can effectively replace the brittle proprietary LOS standards. LOS companies and IMX exchange can sit down and come up with a quick presentation format in XML. If any discrepancies arise between the different LOS companies, IMX can easily adopt their parsing to that format change -- all they would require is an XSL style sheet that can be easily adopted and changed as the standard evolves. For example if

one LOS company has a tag <UnitsOccupied> and the other <PropertyUnits> a 1-line XSL stylesheet is all one would need to convert any of these documents into the <Units> tag used by IMX. Thus any changes to the organically-evolving standard are quite acceptable and easily handled by our distributed system paradigm.

This presents a stark contrast with the use of a proprietary LOS format in the middle tier between the LOS and IMX. With the proprietary formats, any upgrades, like bug fixes or new features added for the new release of the software, present uncoordinated changes to data organization on the LOS end are certain to break the client, often causing unpredictable "black box" behavior, because client's code is responding to the data it assumes it is receiving, not the data it is actually receiving. (26) These "occasional" breakages are the most difficult bugs to track down.

On the other hand, by using XML in the middle tier, we can be assured that the data coming in is validated against the set of iron-clad rules. Any additions to the format that the client does not understand (a new attribute for example) can be ignored by the client or directly signaled to the programmers, who can quickly devise a solution, rather than keep trying to guess how a brittle, proprietary black-box transformation is breaking their system. (26) Data formats always evolve over time, so we want to be able to describe how the format has changed. That way, when a client accesses a service that offers data in the form slightly different from what originally requested, client can still extract some useful information from the exchange. (27)

As a provider of a SOAP-based Web Service, IMX B2B Model is even more flexible, as it's core pricing process can be taken out of its proprietary shell and made available as an Remote Procedure Call to a variety of users. Thus this pricing service becomes a repeatable, portable business process, a part of a true Web-Based Distributed System capable of uniting supply chains across different industries (like finding a house and getting the best price in a mortgage for it -- done through 2 Web Service RPC calls).

Once the loan information is sent to IMX in an XML format, in response to a SOAP request, IMX can perform the matching and pricing and send back to the broker a result set of data, also in XML, indicating loan pricing and the format of the XML response. This information can then be displayed in the browser on the broker's machine, using the presentation XSL or become a part of the LOS interface, through the different set of XSLs. This same data can also be transformed into another XML that is sent to the lender company for storage and uploading into their legacy systems, again using the SOAP protocol. Any changes in this format can be gracefully handled using SOAP to communicate the changes and XSL stylesheets to transform the information into the format necessary for the next step in a Web Distributed System processing. If we really want to re-use the loan data, we must be prepared that the applications and people do not use the same standard as we do. SOAP/XML/XSL combination provides a graceful way to handle that. (23)

In the case of HTML which is displayed by the browser, a human being needs to facilitate the flow of information from one part of the distributed system to the next, because HTML is a presentation language and does not allow automatic, intelligent data transfer. (24) On the other hand, the entire process of uploading, validating, pricing, and locking the loan can be completely automated by using XML. Using SOAP/XML in the middle tier provides a flexible and strong bridge between the different components of the WDS. With the help

of XML, a B2B web application can perform as a re-usable, plug-able Web Service which can be easily extended and augmented as the business needs dictate.(25)

D) Conclusion

As we have seen on the example of a B2B loan matching company, XML is a superior standard for exchange the system state in a web-based distributed system. XML is superior to the existing web communication standards like HTML or proprietary standards like MORNET, because it is human and machine readable, self-validating, organically extensible, easy to transform. XML supports longevity of data, because it is future-proof against format updates and is self-documenting with the use of DTD and XML Schema. Integration aids like Simple Object Access Protocol (SOAP), extend the XML standard's flexibility even further, by automating the exchange of the information and the standard in which the information is contained. SOAP allows core legacy systems buried behind the firewalls be exposed simply and securely, in a platform and language independent Remote Procedure Call to a part of the Distributed System. Thanks to an open standard that Web Services support, they communicate with any module running on any platform. The main benefit of such flexibility is that you can extend existing distributed systems to expose the functionality as Web Services. Regardless of the platform, internal implementation, and vendor, all the Web Services share an unprecedented ability to work together as pieces of one system. Thus, by using XML as the middle tier systems communication tool, we enable real-world viable Web-Based Distributed Systems composed of robust re-usable Web Information Services.

References

- 1) David Siegel. The Web is ruined and I ruined it., in Web Review , Number 4, 1997.
- 2) David Siegel. Creating Killer Web Sites, 2nd edition (September 18, 1997) Hayden Books; ISBN: 1568304331 (page 12)
- 3) Vincent Flanders, et al., Web pages that suck. 1996 Sybex; (page 24)
- 22) w3c XML Schema specification. <http://www.w3.org/XML/Schema>
- 31) CML. <http://www.xml-cml.org>
- 41) W3C XML Protocol Activity. <http://www.w3.org/2000/xp/Activity>
- 42) Web Services Framework, IBM, Microsoft W3C Workshop on Web Services 11-12 April 2001, San Jose, CA USA.

- 10) Rohit Khare and Adam Rifkin. Capturing the State of Distributed Systems with XML. In the World Wide Web Journal Special Issue on XML, Volume 2, Number 4, Fall 1997, Pages 207-218.
- 17) Rick Jelliffe. Using XSL as a Validation Language. Academica Sinica, Taipei, Taiwan. <http://www.ascc.net/xml/en/utf-8/XSLvalidation.html>
- 35) Kevin Brennan. An XML Approach to Network Element Management MS Dissertation Projects, 1999/2000 Department of Computer Science, Trinity College of Dublin.
- 14) Dr. Marry Ann Malloy. Experiences Designing Query Languages for Hierarchically Structured Text Documents. MITRE Corporation. <http://www.w3c.org/TandS/QL/QL98/pp/mitre.html>
- 13) Peter Flynn. The XML FAQ. <http://www.ucc.ie/xml/>
- 12) The Open Healthcare Group. XML and Healthcare. <http://www.openhealth.org/xmlhealth.htm>
- 40) Solution Exchange Standard <http://www.dmtf.org/spec/stan.html>
- 24) Jon Bosak. XML, Java, and the future of the Web. Sun Microsystems Press 1997.
- 30) Didier Martin, et al. Professional XML. WROX August 2000. (page 16,17)
- 30) Didier Martin, et al. Professional XML. WROX August 2000. (page 15)
- 15) Didier Martin, et al. Professional XML. WROX August 2000. (page 70)
- 18) Didier Martin, et al. Professional XML. WROX August 2000. (page 71)
- 19) Didier Martin, et al. Professional XML. WROX August 2000. (page 91)
- 20) Didier Martin, et al. Professional XML. WROX August 2000. (page 132)
- 23) Didier Martin, et al. Professional XML. WROX August 2000. (page 369)
- 21) Didier Martin, et al. Professional XML. WROX August 2000. (page 259)
- 25) Didier Martin, et al. Professional XML. WROX August 2000. (page 800)
- 26) Didier Martin, et al. Professional XML. WROX August 2000. (page 803)

- 27) Didier Martin, et al. Professional XML. WROX August 2000. (page 805)
- 28) Didier Martin, et al. Professional XML. WROX August 2000. (page 806)
- 90) IMX: <http://www.imx.com>
- 91) Mornet Standard: <http://www.efanniemae.com/singlefamily/businesscenter.jhtml>
- 102) MISMO Engineering Guidelines: http://www.mismo.org/mismo/docs/mismo_dn_standard.htm
- 120) Hugh Grant. Using XML with Enterprise Component Systems, XML Journal, 2001 Volume 1, Issue 3.
- 119) Dino Esposito. At Your Service On the Web, Windows 2000 Journal, April 24, 2001
- 121) Simeon Simeonov. Framework for Using Web Services, XML Journal, 2001 Volume 2, Issue 6.
- 118) Dino Esposito. Discovering Web Services, Windows 2000 Journal, June 25, 2001
- 117) Brady Gaster. Gaining Client Interoperability by Using SOAP and WSDL, ASP Today, 07/13/2001
- 104) JP Morgenthal. Where are Web Services Going? Web Services Journal, Preview Issue, June 2001
- 101) MISMO: <http://www.mismo.org/>
- 103) Andy McCright. Writing Your First Web Service, Web Services Journal, Preview Issue, June 2001
- 21) Tony Patton, XML Schemas in Action, XML Journal, 2001 Volume 2, Issue 6.
- 116) Steve Benfield. Web Services: the Power to Change the World? Web Services Journal, Preview Issue, June 2001
- 100) Microsoft XML Schema: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptools/html/cpconxmlschemadefinitiontoolxsdexe.asp>
- 115) David Russel. Electronic Business XML, Web Services Journal, Preview Issue, June 2001