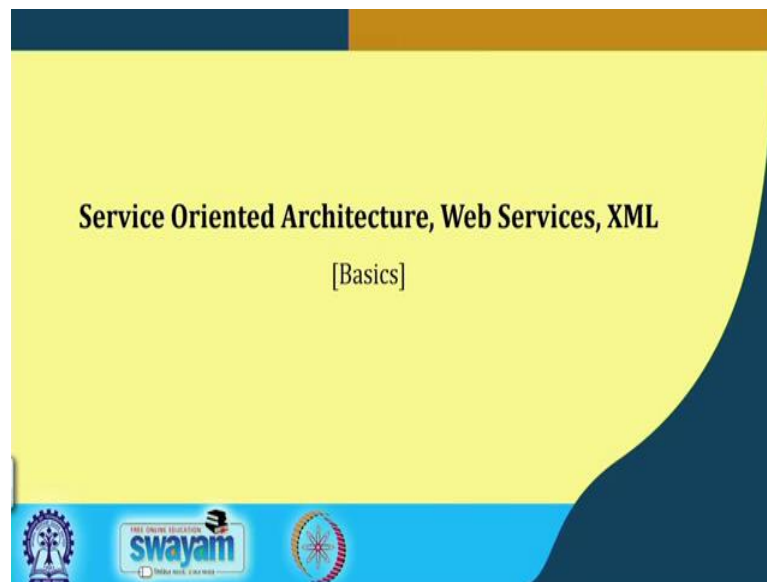


**Spatial Informatics**  
**Prof. Soumya K. Ghosh**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 07**  
**Spatial Web Services (2)**

Hi. So, we will continue our discussion on Spatial Informatics. In the last lecture, we started discussing on Spatial Web Services right. So, as we were discussing so, the spatial web services is a mechanism so to say to help in interoperate or interoperate between diverge data set or heterogeneous data sets right. Like other standard web services so this is a spatial extension to the things and as we were discussing with the several requirement. Rather to for those who were not a custom, those who are not much exposed with web services service oriented architecture XML in the last lecture, we were having a quick overview of these aspects. So, we will continue that in today's lecture; so, that same aspects.

(Refer Slide Time: 01:19)



So, we are discussing about Service Oriented Architecture, Web Service, XML, some of the slides are repetition from the time. So, that we will just I will skip those and go to the other things.

(Refer Slide Time: 01:28)

A presentation slide with a yellow background and a dark blue header. The header contains the text 'SOA' in white. The main body of the slide contains two paragraphs of text. At the bottom, there is a blue banner with two circular logos on the left. The first logo is a gear-like emblem with a figure inside, and the second is a circular emblem with a star-like pattern.

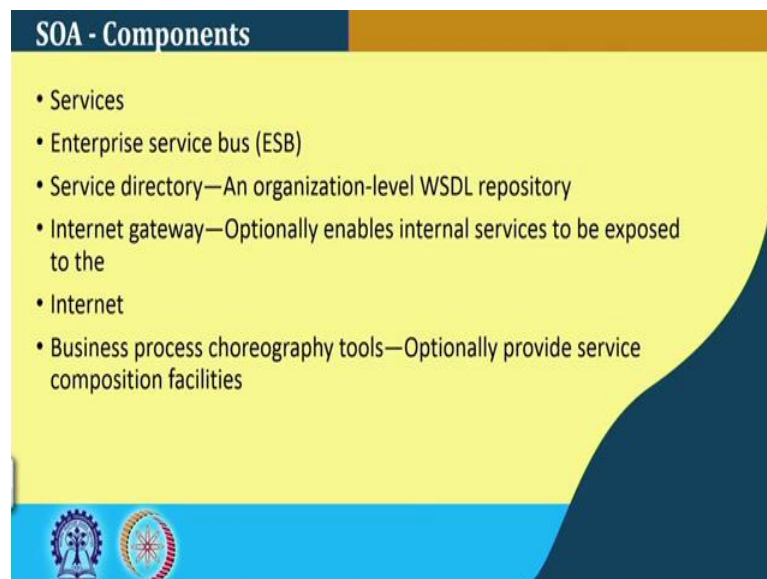
## SOA

A **Service-Oriented Architecture (SOA)** consists of a set of business-aligned services that collectively fulfill an organization's business process goals and objectives.

These services can be choreographed into composite applications and can be invoked through Internet-based open standards and protocols.

So, we discussed about Service-Oriented Architecture.

(Refer Slide Time: 01:32)

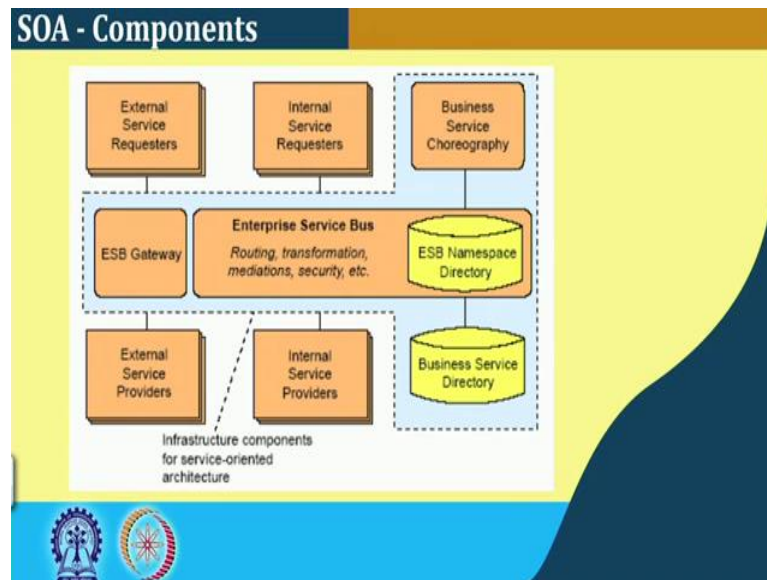
A presentation slide with a yellow background and a dark blue header. The header contains the text 'SOA - Components' in white. The main body of the slide contains a bulleted list of components. At the bottom, there is a blue banner with two circular logos on the left, identical to the ones in the previous slide.

## SOA - Components

- Services
- Enterprise service bus (ESB)
- Service directory—An organization-level WSDL repository
- Internet gateway—Optionally enables internal services to be exposed to the
- Internet
- Business process choreography tools—Optionally provide service composition facilities

And what are the major components of the service-oriented architecture.

(Refer Slide Time: 01:34)



This is a more generic platform; that means, this is true for any service oriented architecture including spatial services and will in this couple of lecture few lectures, we will see that how this spatial services can be realized, how it is useful for our for several deploying the services in different aspects and so on and so forth.

(Refer Slide Time: 02:10)

**WS & SOA**

Web services technology is an ideal technology choice for implementing a service-oriented architecture (SOA):

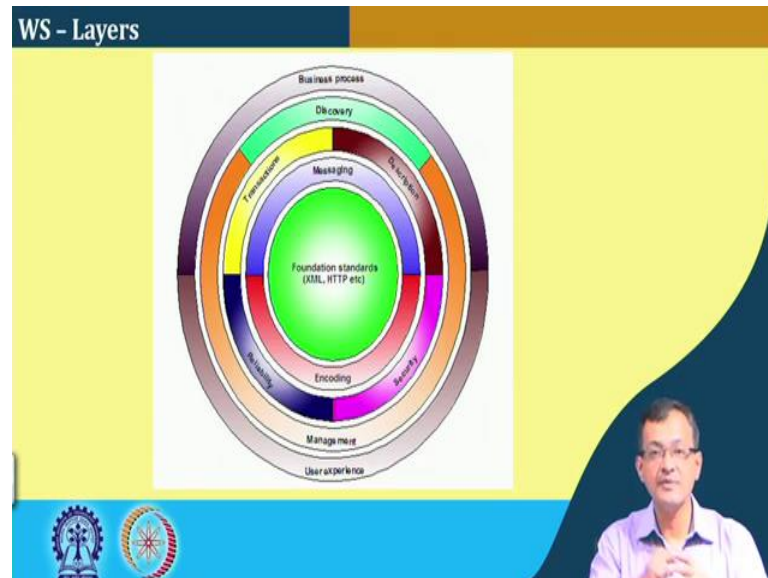
- Web services are standards based. Interoperability is a key business advantage within the enterprise and is crucial in B2B scenarios.
- Web services are widely supported across the industry. All major vendors are recognizing and providing support for Web services.
- Web services are platform and language independent—there is no bias for or against a particular hardware or software platform. Web services can be implemented in any programming language or toolset. This is important because there will be continued industry support for the development of standards and interoperability between vendor implementations.
- This technology provides a migration path to gradually enable existing business functions as Web services are needed.
- This technology supports synchronous and asynchronous, RPC-based, and complex message-oriented exchange patterns.

The slide also features a video inset of a man speaking in the bottom right corner.

Also we talked about web services. Again this is a generic as we are as I am telling that this is a general overview of the service oriented architecture, web services and XML and we will be going deep into the after that again we will come back to the spatial web

services. So, this is what we have seen that this web services technology is an technology for implementing service oriented architecture and so, this services can be realized.

(Refer Slide Time: 02:42)

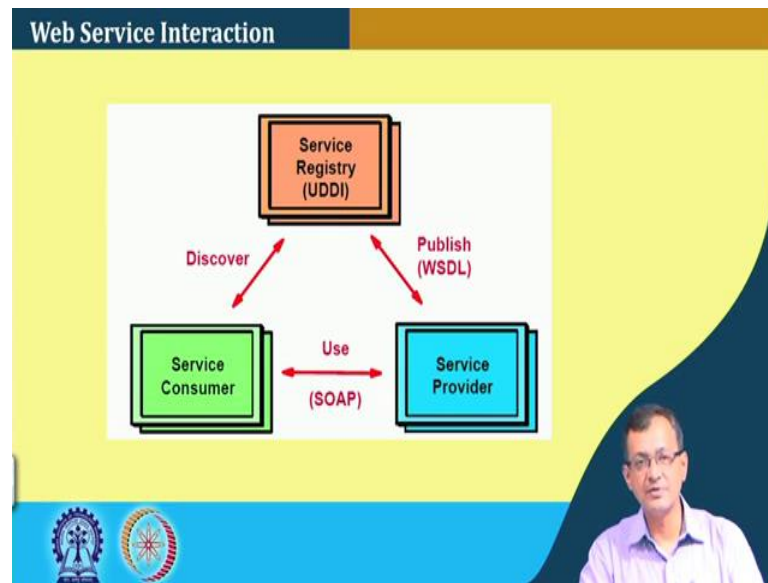


And also we have seen that generic layering or that onion of this web services that core of these things is XML. So, all encoding everything is an XML form as we as many of you know, XML is a data transformation language where as HTML is typically a data representation language right.

So, a browser purchase the HTML data and display on the things where the XML will helps you in transforming transmit transforming one form of data to the other; in other sense, it helps in interoperate right. So, XML needs a carrier sometimes call what we say transport mechanism. Again I am telling do not mix up with the transport mechanism what we are having in case of network layer.

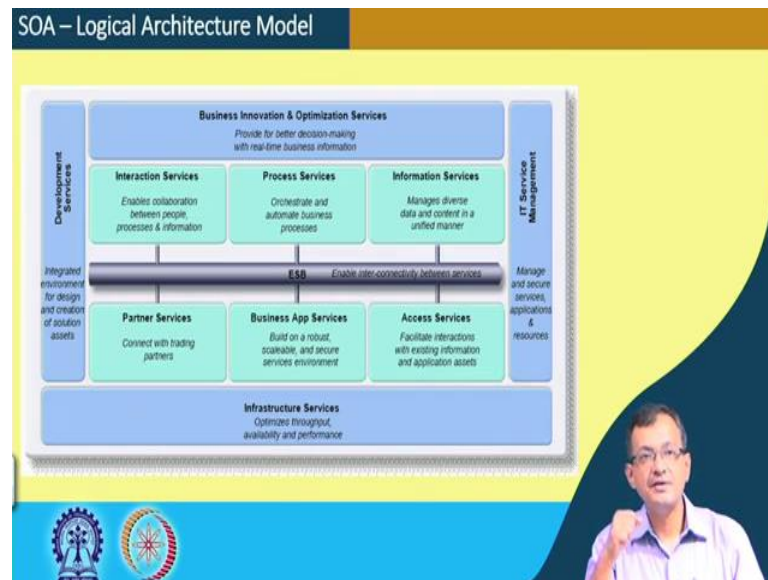
So, and the most popular one is HTTP. So, it piggyback on HTTP and the things HTTP and of course, there are other standards like SMTP etcetera also can be used. Now if you see the XML so, it requires some sort of a messaging and you need to describe that what this message content and also if you are looking for a service, there will be a way to find out that where the services are there. So, those all those mechanisms are in XML and as XML is a standard W3C standard so, it is it can be easily realized in any system right.

(Refer Slide Time: 04:16)



So, and this is the famous triangle you see in several book or several things. So, one is the service repository or registry where the services are kept and housed and there are service consumer and service provider. So, service provider when new services come, it rides to the registry. So, registry is popularly that UDDI registry and can be written or published in the registry by WSDL, Web Service Description Language and the consumer search for the services. And ones that desired services is there so, it understand that who is providing the service and where the service, at which port the service is available, what is the service port and then it binds or with the service provider. So, this is the way it works.

(Refer Slide Time: 05:04)



And this is a more generic view from the other thing what we have seen with that with the other IT management services, development services, infrastructural and overall that business innovation and optimization services.

(Refer Slide Time: 05:21)

### Web Services

*Web Services can convert an application into a Web-application, which can publish its function or message to the rest of the world.*

- Web services are application components
- Web services communicate using open protocols
- Web services are self-contained and self-describing
- Web services can be discovered using UDDI
- Web services can be used by other applications
- XML is the basis for Web services

The slide includes a presenter's video in the bottom right corner and institutional logos in the bottom left.

So, that is the big picture sort of thing and also we see that web services can convert an application or what we say a sort of a coat un coat proprietary application is way well enabled application right which can be published its function and messages to the rest of the world. That is the beauty of the web service philosophy. So, it can be published in the

thing. So, it is web services are application components. Web service communicate via open standard so, anybody can connect that; web service a self contained and self describing.

So, whatever is there, it is self contained and self describing. Web service can be discovered through UDDI will see you one or two one slide on that maybe. The web service can be used for by other application and XML is the basis for all service. So, it is the base for XML is the base operator.

Similarly as we look at the spatial web services all these things are true and instead of a variant of XML, I know I should not say instead a variant of XML called GML is there in the things, Geographic Markup Language.

So, all properties of XML with some geometric property or geographic property or geometry property are in build in to this GML and that is the de facto language for talking from two spatial repository talk to each other. Those who are more interested in web services or quickly want to learn about web services, you may look into the W3 schools that tutorials which are very helpful and quickly you can look at the things right.

(Refer Slide Time: 07:05)



**Web Services - Working**

Basic Web services platform is XML + HTTP

- XML provides a language which can be used between different platforms and programming languages and still express complex messages and functions.
- HTTP protocol is the most used Internet protocol.

Web services platform elements:

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

So, if you look at the web service working so, it is XML plus HTTP. XML is the encoding part and the HTTP is the carrier of the things. So, XML provides the language which can be used between different platforms and programming language and still

express complex mechanisms messages and functions and HTTP protocol is the mostly used internet protocol right. So, that is the mostly as we know that HTTP is mostly used internet protocol not only that, HTTP has the typically most of the routers across the world allow HTTP to pass through protocol. It allows HTTP protocols; that means, that is that is the most what we say desirable or favorite carrier protocol for XML, but there can be an other things.

And web service if you look at the platform, elements one is that SOAP that is the messaging things Simple Object Access Protocol, UDDI for the registry service that Universal Description Discovery and Integration and WSDL, Web Service Description Language right; how do isa. So, these are the forms there are other forms also. So, this is the very basics of web services.

(Refer Slide Time: 08:24)



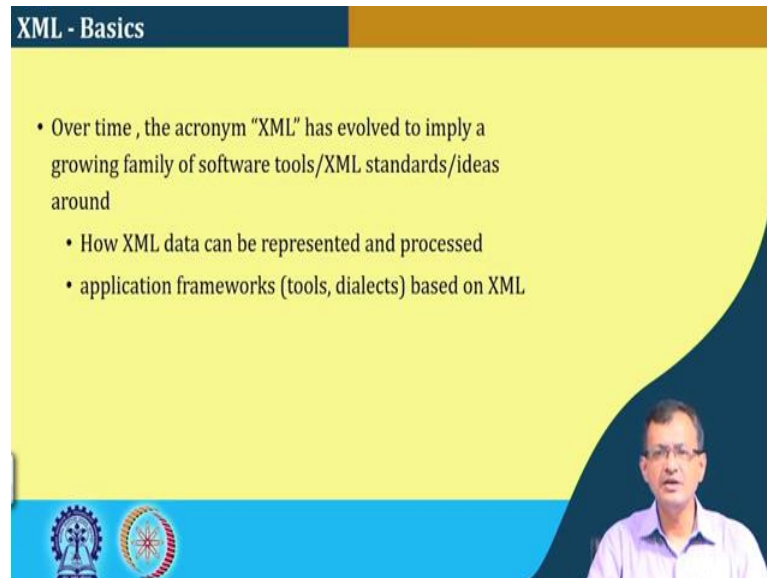
**Web Services - Need/ Uses**

- Interoperability has High Priority
- Web Services take Web-applications to the Next Level
- Reusable application-components.
- Connect existing software

So, need and uses are obvious; interoperability is the high priority it promotes. Web service take web application to the next level that means it is as we have seeing that from your desktop or in house application can be made to a application on the web. Reusable application component as it is the mechanisms allow you to reuse you different application component different context and it allows you to connect to your existing software or sometimes what we say proprietary software's or some software which is legacy software which you may be doing something. So, it allows you to connect this with the rest of the other things right. It is not only external world even within your own

organizations some of the proprietary software is working something this web service allow you to the way it connects to the rest of the world.

(Refer Slide Time: 09:20)



**XML - Basics**

- Over time, the acronym "XML" has evolved to imply a growing family of software tools/XML standards/ideas around
  - How XML data can be represented and processed
  - application frameworks (tools, dialects) based on XML.

Now, we look at the very basics of XML; again it is a known thing. So, just to quickly browse to it or otherwise understanding other aspects of spatial web service will be little difficult. So, over the time, this XML has evolved simply a growing family of software tool, XML standard, ideas around, how XML can be represented and processed; application frameworks based on XML. So, it is a as your it in a it is a extensible markup language. It is a more of a data transformation or data translating type of language than the transformation language rather than data representation so; that means, if XML data cannot be directly represented.

So, for representing XML data we require some style sheet. So, XML contain the data along with the there is a mechanism to keep the description of the data. And there is the beauty right, I form a data of my in proprietary other things. The portion of the data I want to expose to the external world, I can create a XML document and the data description as something what we say popularly XSD or XML schema definition by which I can communicate right. And that is the core of any different business processes going around right like, you we book a train ticket, flight ticket through some travel portal and the portal is neither having flight or something. So, what it is exchanging is

the XML data across the things; not only that, we can easily pay from different mechanisms and so on and so forth right.

(Refer Slide Time: 11:00)

The slide is titled "What is XML?" and features a yellow background with a blue header and footer. The header contains the title. The main content area lists four bullet points: "A syntax for 'encoding' text-based data (words, phrases, numbers, ...)", "A text-based syntax. XML is written using *printable Unicode* characters (no explicit binary data; character encoding issues)", "Extensible. XML lets you define your own *elements* (essentially *data types*), within the constraints of the syntax rules", and "Universal format. The syntax rules ensure that all XML processing software *MUST* identically handle a given piece of XML data." Below the list is a pink box with the text "If you can read and process it, so can anybody else" and a red arrow pointing to it. The footer contains two logos on the left and a portrait of a man on the right.

**What is XML?**

- *A syntax* for "encoding" text-based data (words, phrases, numbers, ...)
- *A text-based syntax*. XML is written using *printable Unicode* characters (no explicit binary data; character encoding issues)
- *Extensible*. XML lets you define your own *elements* (essentially *data types*), within the constraints of the syntax rules
- *Universal format*. The syntax rules ensure that all XML processing software *MUST* identically handle a given piece of XML data.

If you can read and process it, so can anybody else

So, what is XML? A syntax for encoding text based data. So, it is a text based data; word, phrases, numbers. A text based syntax: XML is written in printable texts or simple printable unicode character, no explicit binary data, character encoding issues right. So, it is a printable character set. Extensible: XML can let you define your own element; that means, data types like HTML mostly all of you might have used.

So, HTML is the data tags etcetera are defined; you cannot use your own tags. Here you it is extensible, you can use your tag let us say tremendous flexibility. It based on your domain, you can define your own tag; obviously, when you define the own tag then the to know that other person how did he, how the other systems or person can know. So, there should be that way of defining those tags right and those things has to be there.

And it is a universal format. The syntax rules ensure that all XMLs processing software must identically handle a given XML data in the same way right; that means, whatever your XML parser, the understanding for the theme is should be same. Otherwise looking at the XML data, you interpret in some way I interpret in some way that the whole purpose of interpretability interoperability has gone right. So, this is the core of the things. So, that things; that means, if you can read and write at as they say if you can read and read and process it right so, can anybody else; so can anybody else.

That means if you can transform to XML data your data which you want to exposed or shared with the rest of the world, then the processing of the data is trivial by and any XML parser should be there.

(Refer Slide Time: 12:48)

What is XML: A Simple Example

XML Declaration ("this is XML")

Binary encoding used in file

```
<?xml version="1.0" encoding="iso-8859-1"?>
<partorders
  xmlns="http://myco.org/Spec/partorders">
  <order ref="x23-2112-2342"
    date="25aug1999-12:34:23h">
    <desc>Gold sprockel grommets,
      with matching hamster
    </desc>
    <part number="23-23221-a12" />
    <quantity units="gross"> 12 </quantity>
    <deliveryDate date="27aug1999-12:00h" />
  </order>
  <order ref="x23-2112-2342"
    date="25aug1999-12:34:23h">
    ... Order something else ...
  </order>
</partorders>
```

swayam

And this also allows you to what the system using what sort of things are at the other end. It is a absolutely not botheration on the other thing. So, if the provider and consumer when they talk to each other, talking in XML terms means they do not bothered there is no there is full liberty of using your proprietary software tools etcetera and other end.

So, if you look at a simple example so, this is the way it is defined right. First that declaration, we say it is a XML file and also the what is the encoding scheme and then there are see there are several tax like parts of order, description. These are a which are defined for this purpose right only right.

So, if I want to defined a student schema, then I can defined I think the student name, student roll number whatever way I want to do. Only thing I need to be careful that other if I want to other people to read that data so, that I need to see other schema or structure of that XML right. So, there is a this is the general structure. So, these are the different tags all this XML tags has a end tags. So, it starts with this and ends with this slash partorder type of things.

There is a thing call xmlns, we will come to that XML namespace. So, that there is no clash of the things right. So, we will come to that aspects.

So, this is pretty handy right, you define your own tags and things like that though processing may be little complex because your tags the other party should understand the schema definition etcetera. But otherwise, you see that whatever you want to define on your own you can this. Other part whatever you want to share your data you share it like suppose the it is you do not want to share some data, you do not provide that right. So, that it is not contained in the XML data and it is you see it is printable human readable per say and type of things.

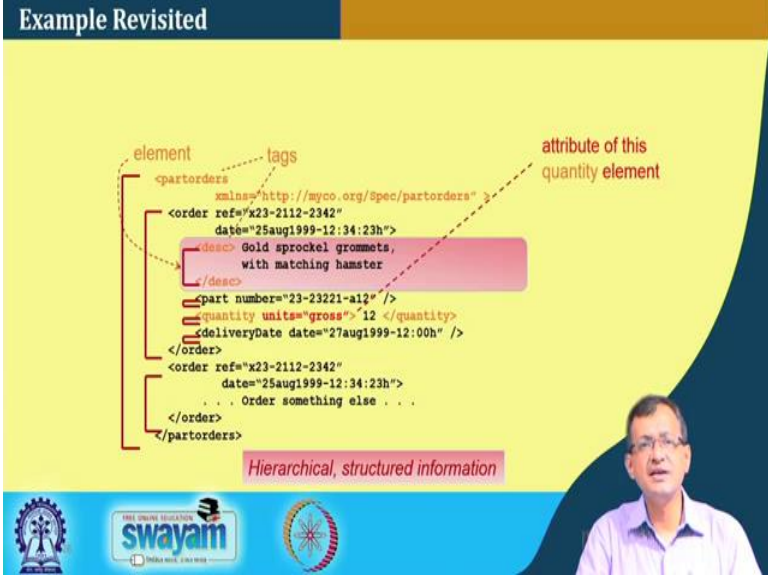
(Refer Slide Time: 14:46)

**Example Revisited**

```
<partorders
  xmlns="http://myco.org/spec/partorders"
  <order ref="x23-2112-2342"
    date="25aug1999-12:34:23h">
      <desc> Gold sprockel grommets,
        with matching hamster
      </desc>
      <part number="23-23221-a12" />
      <quantity units="gross">12 </quantity>
      <deliveryDate date="27aug1999-12:00h" />
    </order>
    <order ref="x23-2112-2342"
      date="25aug1999-12:34:23h">
      . . . Order something else . . .
    </order>
  </partorders>
```

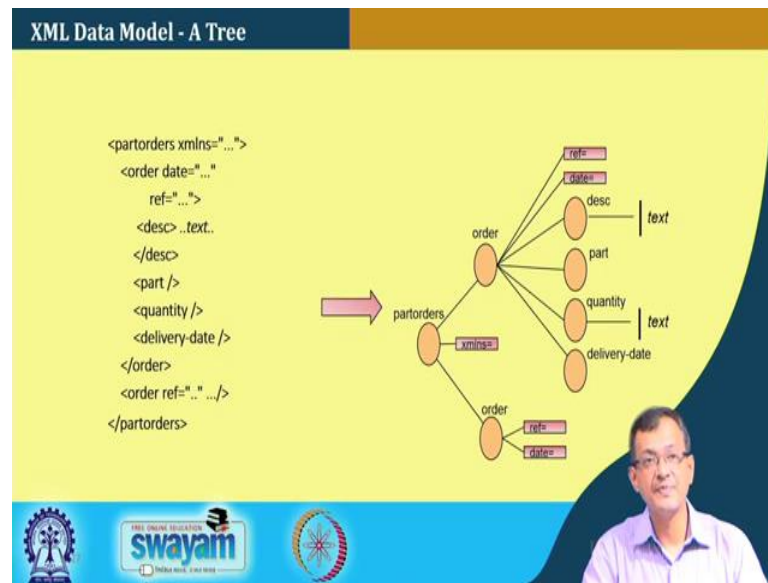
element tags attribute of this quantity element

*Hierarchical, structured information*



So, what we say this is a XML tags, the same thing and you see, there is a inherent hierarchical structure. There is a root and the tree grows like this right and the tags are having maybe having different attribute like attributes like in this case, it is showing that units is gross and type of things any different attributes and it is very flexible in defining.

(Refer Slide Time: 15:14)



So, see if you look at this, it is some sort of a sort of a tree is there right. So, part order which has two orders and then, you have the orders have different type of other references. So, one order is this to this other order is on this and it has different other this order is at different description. So, it is like a hierarchical tree.

So, now if we can understand this XML tree structure so, I can parse this tree right or I can write a parser exploiting this thing right. And again I am repeating so when we talk about spatial web services which is primarily based on XML or what we will say as a GML and then it all follow up of all this processing. Now we see that if I want to interoperate, I exchange the XML or GML data which can be easily parse at the consumer end and can be processed at the thing.

(Refer Slide Time: 16:19)



**XML**

- **Simple** (like HTML -- but not quite so simple)
  - Strict **syntax** rules, to eliminate syntax errors
  - syntax **defines** structure (hierarchically), and **names** structural parts (element names) -- it is **self-describing data**
- **Extensible** (unlike HTML; vocabulary is not fixed)
  - Can create your own **language** of tags/elements
  - **Strict syntax** ensures that such markup can be reliably processed
- Designed for a **distributed environment** (like HTML)
  - Can have data all over the place; can retrieve and use it reliably
- Can **mix** different data types together (unlike HTML)
  - Can mix one set of tags with another set; resulting data can still be reliably processed

So, XML it is simple like HTML not quite simple, but so simple, but it is simple. So, strict syntax rule no eliminate of syntax errors so that should be whatever syntactically, it should be correct.

So, defines syntax defines structure or the hierarchy of the thing and the name structural element or it should be self describing stuff. It is extensible unlike HTML, vocab is not fixed. So, you can use your own vocab. Design for a distributed environment like HTML also is distributed can have data all over place, can retrieve use it reliably, can mix different data types together unlike html right.

So, I can have different data type and create another XML file with all those data types. So, I take a student academic data, student personal data, student hall related data and mix them and create a data set which is required for something like if I want to create a data set for training placement and then taking all those data, then I create another XML file which can be parsable by different companies etcetera right.

Can mix different data types that is a best part. So, in you see this is inherently allow you to interoperate with multiple depository and create a data which can be used to this right. So, that is also exploited even when we look about this spatial web services.

(Refer Slide Time: 17:51)

### XML Processing

```
<?xml version="1.0" encoding="utf-8" ?>
<transfers>
  <fundsTransfer date="20010923T12:34:34Z">
    <from type="intrabank">
      <amount currency="USD"> 1332.32 </amount>
      <transitID> 3211 </transitID>
      <accountID> 4321332 </accountID>
      <acknowledgeReceipt> yes </acknowledgeReceipt>
    </from>
    <to account="132212412321" />
  </fundsTransfer>
  <fundsTransfer date="20010923T12:35:12Z">
    <from type="internal">
      <amount currency="CDN" >1432.12 </amount>
      <accountID> 543211 </accountID>
      <acknowledgeReceipt> yes </acknowledgeReceipt>
    </from>
    <to account="65123222" />
  </fundsTransfer>
</transfers>
```

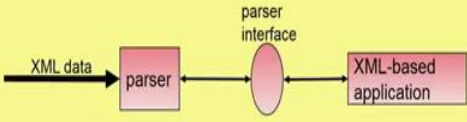
xml-simple.xml



So, processing it is again if you look at the simple XML data so, it is a there are tags, there are definition, there are attributes of the tags and so and so forth.



(Refer Slide Time: 18:03)

### XML Parser Processing Model



```
graph LR
    XMLdata[XML data] --> parser[parser]
    parser <--> interface((parser interface))
    interface <--> app[XML-based application]
```

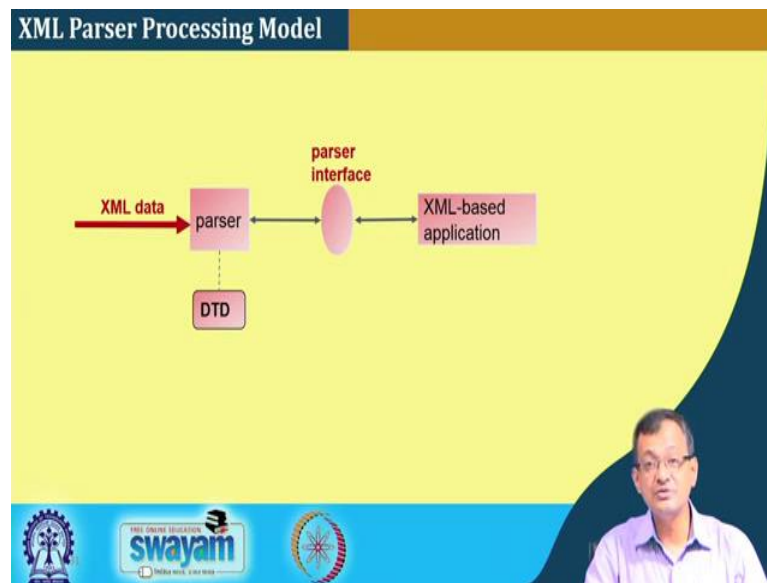
- The parser must verify that the XML data is syntactically correct.
- Such data is said to be **well-formed**
  - The minimal requirement to “be” XML
- A parser **MUST** stop processing if the data isn't well-formed
  - E.g., stop processing and “throw an exception” to the XML-based application. The XML 1.0 spec **requires** this behaviour



And so; that means, if there is a tree structure, then I can if I have a parser the XML data can be parsed and then can be used by the XML application XML based application. So, it parse the data and use the data to process the thing. So, the parser must verify the XML syntactically correct or not right.

So, to look at the XML strict syntax rules etcetera, you need to follow some tutorial or books. It is beyond the scope of this particular course, but you see it is a very strict on the syntax. So, it should be syntactically correct. So, syntactically correct XML are we called well formed XML such data is said to be well formed. So, a parser must stop processing if the data is it not well formed. As this is used for heterogeneous distributed environment so, we need to be well formed data right. So, the XML one of the requirement of this things is the XML data is the well formed data.

(Refer Slide Time: 19:02)



So, the XML data, this parser and there is a concept or DTD we will come to that. So, that it allows to see that whether it is following some syntax right like I will come to that.

(Refer Slide Time: 19:22)

**Two types of XML parsers**

- **Validating parser**
  - **Must** retrieve all entities and must process **all** DTD content. Will stop processing and indicate a failure if it cannot
  - There is also the implication that it will test for compatibility with other things in the DTD -- instructions that define syntactic rules for the document (allowed elements, attributes, etc.). We'll talk about these parts in the next section.
- **Non-validating parser**
  - Will try to retrieve all entities defined in the DTD, but will **cease processing the DTD** content at the first entity it can't find. But this is not an error -- the parser simply makes available the XML data (and the names of any unresolved entities) to the application.

Application behavior will depend on **parser type**

The slide features a yellow background with a blue header and footer. The footer includes logos for Swamyam and other educational institutions, along with a small video inset of a man in a blue shirt.

So, So, see two things are there. One it should be syntactically correct right; another is semantically correct right like I say, I want to transmit student data which consist of student name, roll number, phone number, call of residence, department, year of registration and what category like UG, PG or whatever. This is my objective or consider a I want to transmit a data recording road network, then I want to find out that what is the name of the road, whether there is a idea of the road, type of the road, whether it is a metal nonmetallic road. And also maybe if there is a international coding etcetera with that road and type of things, I can add lot of other things around the road. Now this is the structure this is the semantic I want to transmit.

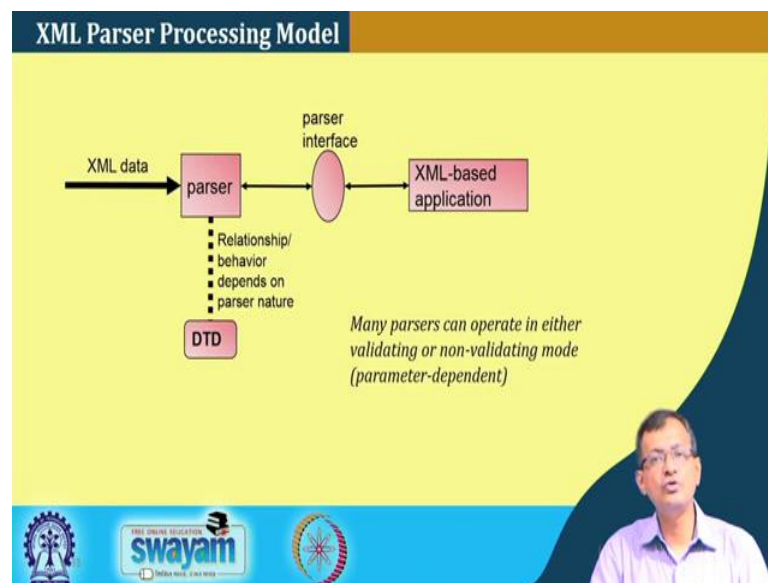
Now, like XML data is of the form. Now all those or student data is of the all the of the form. Now all the tags etcetera, I have defined based on those right. Now if I want these tags; now if I want the other and parsers to understand this so, I need to define this structure right. Think about the databases. So, database structure need to be defined before the data is populated or transmitted so that need to be defined.

Now if your XML data is not confirming so, one is syntactically that representation is correct that is well formed. But if you are XML data is not confirming to your definition, then it is what we say whether it is a validating or non-validating.

So, in case of a validating parser, the XML must retrieve all entity that must be in the retrieve document. So, that the in the definition document right. We will stop processing

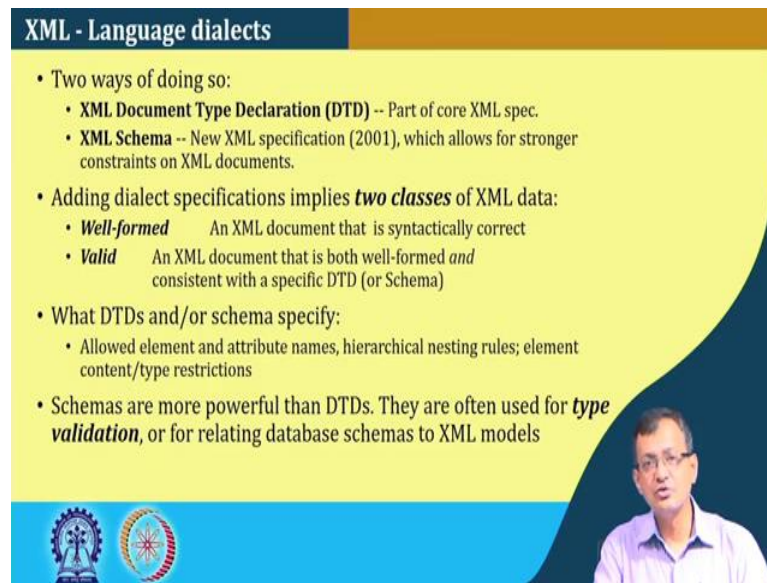
the things it, it is can be well formed, but it is not validating. In case of non validating parser, it does not it is only checking at the whether it is a well form or not it is like, we will try to retrieve all entries define in the DTD, but we will sees processing the DTD content at the parser you can find, but it is not an error. The parser simply makes the available XML data, the name of the unresolved entity to the application. Now that let the application take a call. So, well formed is syntactic error and if we say validating, it is semantically correct data.

(Refer Slide Time: 21:58)



So, what we see that the parser, we will also it will by default take the wells well formed things along with that relationship behaviour depends on the parser nature by the DTD data. Say that whether it is semantically correct or not; that meaning that what sort of data, I want to transmit there it is there or not right. And then we have that parser interface and then the XML based application. So, many parser can operate either validating or non validating mode based on the application and the parameter dependents.

(Refer Slide Time: 22:34)



**XML - Language dialects**

- Two ways of doing so:
  - **XML Document Type Declaration (DTD)** -- Part of core XML spec.
  - **XML Schema** -- New XML specification (2001), which allows for stronger constraints on XML documents.
- Adding dialect specifications implies **two classes** of XML data:
  - **Well-formed** An XML document that is syntactically correct
  - **Valid** An XML document that is both well-formed *and* consistent with a specific DTD (or Schema)
- What DTDs and/or schema specify:
  - Allowed element and attribute names, hierarchical nesting rules; element content/type restrictions
- Schemas are more powerful than DTDs. They are often used for **type validation**, or for relating database schemas to XML models

The slide features a blue header with the title 'XML - Language dialects'. The main content area is yellow with a blue curved border on the right. A video inset in the bottom right corner shows a man with glasses speaking. The bottom of the slide has a blue bar with two circular logos on the left.

So, if you look at that how to define this structure etcetera as we are looking at the DTD or document type declaration. So, it is a part of the core XML, but spec right. The, but the more popular today is the XML schema right or so, new XML schema specification which are two thousand on onwards was defined which allows for stronger constraint of XML documents right. So, adding dialects specification of two classes of XML well formed and XML document that is syntactically correct and validate valid and XML document that is both well formed and consistent with a specific DTD or schema right.

So; that means, well formed as we are taking the syntactically and this is confirming to this DTD or the schema definition or popularly like DTD what we say XSD or XML schema definition things right. Incidentally DTD is not retained in that style of XML whereas the XSD is retained in the same style as the XML; it is a XML file. That means, the parsing of this because you require those parsing also, this XML XSDS much with the same type of parsing parser, we will do. So, what this DTD and XML schema or XSD specify? Allow elements, attribute, names, hierarchical nested rules; element content etcetera, type restriction and all those things that is more of a structural constraint. Schemas are more powerful than DTD.


They often use for type validation or relating database to XML model etcetera. So, ideally what happened that you have the data then you make a XML then make a schema

make the database schema based on that and then from the data, you push it to the database because querying etcetera on the database are much optimized right. Think about the spatial database with large data sets will be like that.

(Refer Slide Time: 24:23)

### XML Schemas

- A new specification (2001) for specifying validation rules for XML.  
Specs: <http://www.w3.org/XML/Schema>  
Best-practice: <http://www.xfront.com/BestPracticesHomepage.html>
- Uses **pure XML** (no special DTD grammar) to do this.
- Schemas are more powerful than DTDs - can specify things like integer types, date strings, real numbers in a given range, etc.
- They are often used for **type validation**, or for relating database schemas to XML models
- They don't, however, let you declare entities -- those can only be done in DTDs.




So, we the same thing that XML schema which is defined in that XML slash schema definitions.

(Refer Slide Time: 24:33)

### XML Schema version - Example

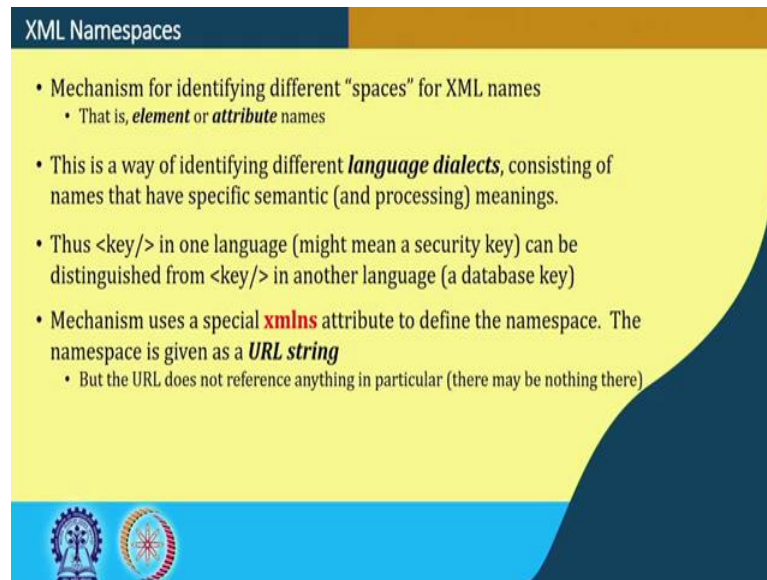
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="accountID" type="xs:string"/>
  <xs:element name="acknowledgeReceipt" type="xs:string"/>
  <xs:complexType name="amountType">
    <xs:simpleContent>
      <xs:restriction base="xs:string">
        <xs:attribute name="currency" use="required"/>
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="USD"/>
            ... {some stuff omitted} ...
          </xs:restriction>
        </xs:simpleType>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="fromType">
    <xs:sequence>
      <xs:element name="amount" type="amountType"/>
      <xs:element ref="transitID" minOccurs="0"/>
      <xs:element ref="accountID"/>
      <xs:element ref="acknowledgeReceipt"/>
    </xs:sequence>
  </xs:complexType>
  ...
</xs:schema>
```

simple.xsd



And the assemble of a XML schema XSDs like this if you look at the compare with the DTD the same. So, you will see here the it is more of the definitions are there. It is not the content of the data, but information or the about the data out here.

(Refer Slide Time: 24:51)



**XML Namespaces**

- Mechanism for identifying different “spaces” for XML names
  - That is, *element* or *attribute* names
- This is a way of identifying different *language dialects*, consisting of names that have specific semantic (and processing) meanings.
- Thus <key/> in one language (might mean a security key) can be distinguished from <key/> in another language (a database key)
- Mechanism uses a special **xmlns** attribute to define the namespace. The namespace is given as a **URL string**
  - But the URL does not reference anything in particular (there may be nothing there)

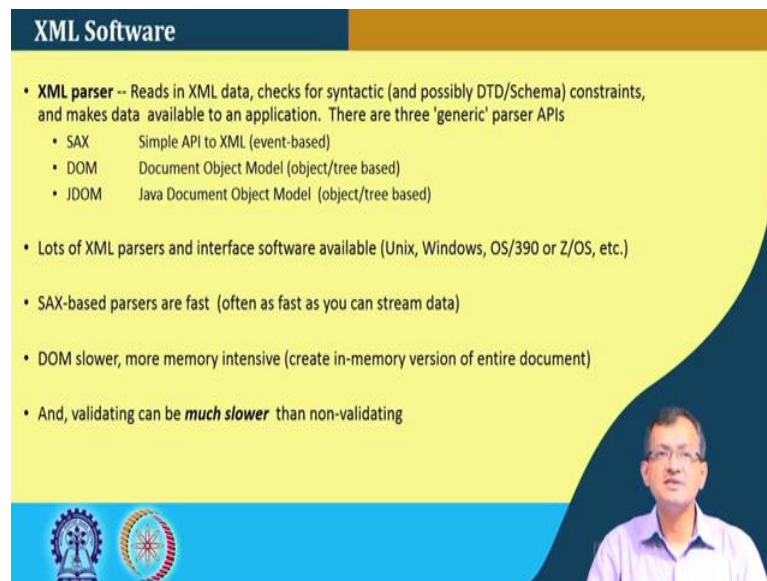
There is a another concept as if we have seen xmlns. So, next xmlns namespace mechanism for identifying different spaces for XML schema they are element, attribute, names etcetera right. Like the popular example they say that if I if I have a name of a particular things, it may have different meaning for different type of things. Like for example, say if I have table like, table can be a furniture table can be a database; table or spreadsheet table. So, which you are referring is important to understand because which table.

So, that is the namespace. If my namespace is furniture, then it is referring to this sort of table. If the namespace is something say spreadsheet or type of thing, then it is something some other table right. So, if my elements or entities or attributes is based on the things. In other sense, I can use different type of namespace. You see that the beauty of the things say, I say that I my student definition is there.

Then I if I am say communicating with some other party or say I am a banking arrangement with the banks which by which students pay registration etcetera with the bank I say that my this is the data, you check my name space for the definition of this data right. So, that is what we see that that identifying different language dialect

consisting of names that has specific semantics and processing meaning the. Thus key is one language, this might mean a security key as you are telling key or can be in a distinguished the key; another thing which is a database key right. So, as we are telling table and type of things. So, this mechanism uses a special xmlns attribute to define the namespace. The namespace is given by URL string right.

(Refer Slide Time: 26:50)



**XML Software**

- **XML parser** -- Reads in XML data, checks for syntactic (and possibly DTD/Schema) constraints, and makes data available to an application. There are three 'generic' parser APIs
  - SAX Simple API to XML (event-based)
  - DOM Document Object Model (object/tree based)
  - JDOM Java Document Object Model (object/tree based)
- Lots of XML parsers and interface software available (Unix, Windows, OS/390 or Z/OS, etc.)
- SAX-based parsers are fast (often as fast as you can stream data)
- DOM slower, more memory intensive (create in-memory version of entire document)
- And, validating can be **much slower** than non-validating

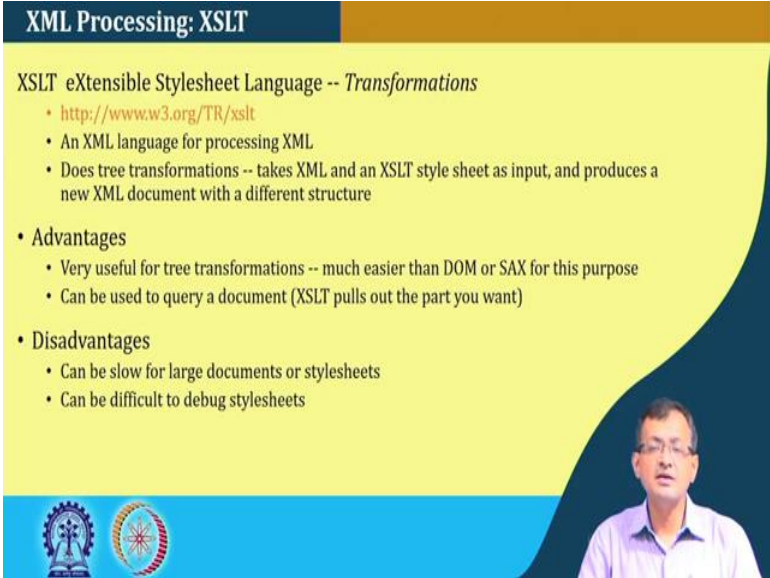
The slide features a yellow background with a blue header and footer. In the bottom right corner, there is a small video inset of a man with glasses speaking. The footer also contains two circular logos on the left.

And we have different XML software or parser. So, XML parser reads a XML data, checks for syntactically correct; it should be well formed and possibly with DTD and schema for that meaning or structural or semantic correctness of the constant and makes the data available to the application. So, there are three type of typical generic parser, one is SAX, Simple API to XML even best mostly available with most of the open source system. DOM Document Object Model object or tree based structure and JDOM, Java Document Object Model; this is also object or tree based. So, these are the popular things; more if you want you need to consult some book or material.

So, lots of XML parser interface software like UNIX software different OS systems are available and as you understand, if we if you can under good in coding, you can write your own parser to do that. DOM a SAX parser is fast often as fast as your data stream, but that not that very versatile. Whereas DOM is slower more memory intensive, create in memory things, but it has a more versatile thing can easily handle this schema and other definition. And of course, the validating can be much slower than non validating.

So, validating parser means validating with the schema or DTD, mostly another schema DTD; hardly used these days DTD. So, and it takes time because validating, we need to validate and process the thing.

(Refer Slide Time: 28:23)



**XML Processing: XSLT**

XSLT eXtensible Stylesheet Language -- *Transformations*

- <http://www.w3.org/TR/xslt>
- An XML language for processing XML
- Does tree transformations -- takes XML and an XSLT style sheet as input, and produces a new XML document with a different structure

• **Advantages**

- Very useful for tree transformations -- much easier than DOM or SAX for this purpose
- Can be used to query a document (XSLT pulls out the part you want)

• **Disadvantages**

- Can be slow for large documents or stylesheets
- Can be difficult to debug stylesheets

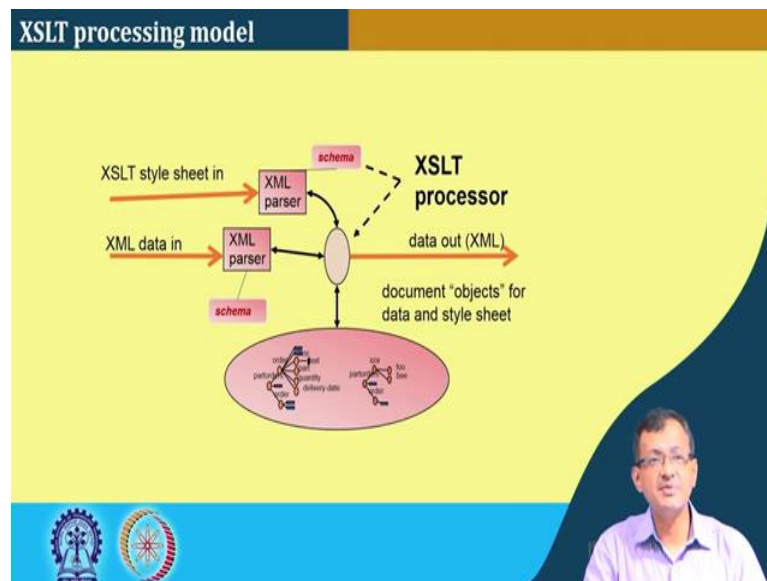
There is a another concept call what we say student XSLT eXtensible Style Sheet Language for transformation right. So, what we say style sheeting of the thing right. If I have the data, I want to display that thing data. So, how do I do? I need to have the stylesheet of the things right. So, what style need to be displayed right. So, if I suppose I have the road network of different category of road national highway street etcetera, I want to have different colors etcetera for the things. So, where will I get? I need to have a style sheeting right of the data even I suppose I have a road network want to wants to display only the national highways, then I other stylesheet I can basically switch off.

Similarly, for any type of data structure so, this allows you to in some sort of a processing purpose of the XML. So, it is a the reference you will get in the W3 consortium. So, an XML language for processing XML does tree transformation takes XML as an XML stylesheet is input and produce another new XML right. So, based on those structure, you get a defined XML which is a another type of structure.

Advantage: very useful tree transformation much easier than DOM, SAX and other things. So, it is on the fly you can do the transfer can be used for query a document. So, XSLT pulls out the part you want so, I want to query and extract only those things out of

the things. So, I can have those type of. So, it sort of a filtering mechanisms what you say. Disadvantage in some cases, it can be slow for large document or stylesheets can be difficult to debug the stylesheets right. If there is a problem in the stylesheeting, then it will be a problem in some times for large data's processing and type of things XML processing; it may be a difficult to debug the stylesheeting.

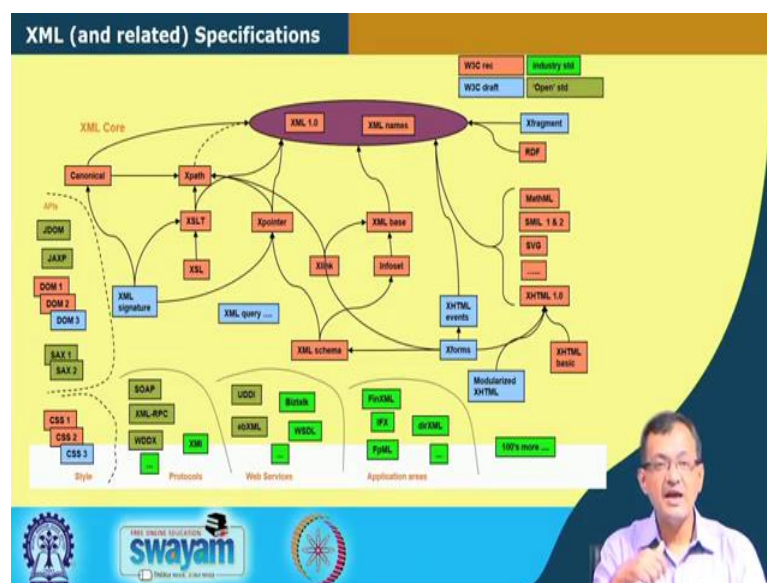
(Refer Slide Time: 30:11)



So, if we look at so, we have the XML file XML data in the things with the parser and the schema definition along with we have this XML stylesheet which can see this type of this is little difficult to read. But nevertheless I have a tree and based on my XSLT that the refined tree is there and based on that that data is get a output. So, you see there is a lot of mechanism that what you what you generate out the data and then what you output that is a lot of mechanism to filter it right.

I can also filter it by inputting I only want this data for this application, this data for other application and type of things. And there is there is a if you see that there is a lot of flexibility into the things right with the data. Now easily can be transferred transform into the different form. This helps indirectly in interoperability of data which will be useful for our this mechanism of spatial web services where this data transformation heterogeneous data sources need to be coded trade and type of things.

(Refer Slide Time: 31:15)



And this is again a picture taken from internet sources. And so, this is a big XML family rather it may not be all exhaustive. So, you see that the type of things which are having. We are looking at different protocols like SOAP and other etcetera. There are different web services which are available WSDL, UDDI; there are different application areas where it can be do and there are we have this different APIs which allows this parsing of the data etcetera. There are different stylesheeting mechanisms and so and so forth.

So, this is the full family of or not full family this is a bigger family of XML that may be several other things into the things. Now and this allows me to manage the data in a appropriate way. Now what will see with this knowledge of the XML and you can always encourage to read some of the things specially the W3 schools tutorial.

So, what we can we will use these thing for our spatial web services. So, as we mentioned at the beginning for the last lecture that we are working on we will be looking at a variant of XML call GML, Geographic Markup Language which is again a OGC standard which is followed by all participating all the other all countries and vendors and etcetera.

So, if you can encode in a data in a GML format so, it is transformable to easily by the other; in the other sense, it helps in interoperating. So, with these things, let us stop today. We will continue our discussion on spatial web services in the subsequent lecture.

Thank you.