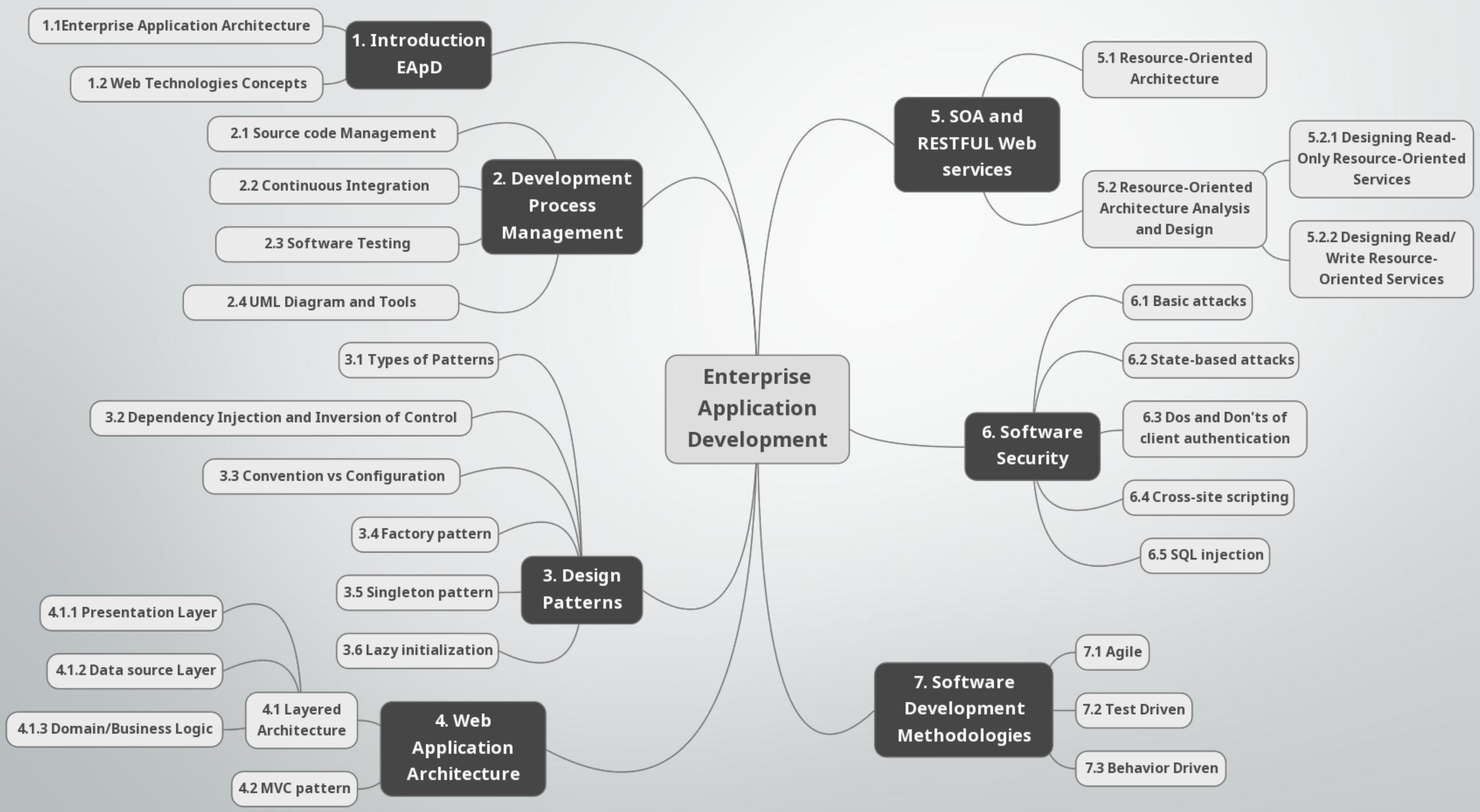
Enterprise Application Development

[BE SE-7th Semester]

Nepal College of Information Technology

POKHARA UNIVERSITY

Enterprise Application Development



Enterprise Application Development
1. Introduction EApD
1.1Enterprise Application Architecture
1.2 Web Technologies Concepts
2. Development Process Management
2.1 Source code Management
2.2 Continuous Integration
2.3 Software Testing
2.4 UML Diagram and Tools
3. Design Patterns
3.1 Types of Patterns
3.2 Dependency Injection and Inversion of Control
3.3 Convention vs Configuration
3.4 Factory pattern
3.5 Singleton pattern
3.6 Lazy initialization
4. Web Application Architecture
4.1 Layered Architecture
4.1.1 Presentation Layer
4.1.2 Data source Layer
4.1.3 Domain/Business Logic
4.2 MVC pattern
5. SOA and RESTFUL Web services
5.1 Resource-Oriented Architecture
5.2 Resource-Oriented Architecture Analysis and Design
5.2.1 Designing Read-Only Resource-Oriented Services
5.2.2 Designing Read/ Write Resource-Oriented Services
6. Software Security
6.1 Basic attacks
6.2 State-based attacks
6.3 Dos and Don'ts of client authentication
6.4 Cross-site scripting
6.5 SQL injection
7. Software Development Methodologies
7.1 Agile
7.2 Test Driven
7.3 Behavior Driven

Web Service definition

A simple definition:

“a Web Service is an application component accessible over open protocols”.

History

- Web services evolved from previous technologies that served the same purpose such as RPC, ORPC (DCOM, CORBA and JAVA RMI).
- Web Services were intended to solve three main problems:

1. Interoperability
2. Firewall traversal
3. Complexity

Interoperability

- Earlier distributed systems suffered from interoperability issues because each vendor implemented its own on-wire format for distributed object messaging.
- Development of DCOM apps strictly bound to Windows Operating system.
- Development of RMI bound to Java programming language.

Firewall traversal

- Collaboration across corporations was an issue because distributed systems such as CORBA and DCOM used non-standard ports.
- Web Services use HTTP as a transport protocol and most of the firewalls allow access though port 80 (HTTP), leading to easier and dynamic collaboration.

Complexity

- Web Services is a developer-friendly service system.
- Most of the above-mentioned technologies such as RMI, COM, and CORBA involve a whole learning curve.
- New technologies and languages have to be learnt to implement these services.

Web Service definition revisited

- A more precise definition:
  - an application component that:
    - Communicates via open protocols (HTTP, SMTP, etc.)
    - Processes XML messages framed using SOAP
    - Describes its messages using XML Schema
    - Provides an endpoint description using WSDL
    - Can be discovered using UDDI

Sockets

- Sockets, or ports, are a very low level software construct that allows computers to talk to one another
- When you send information from one computer to another, you send it to a port on the receiving computer
  - If the computer is “listening” on that port, it receives the information
  - In order for the computer to “make sense” of the information, it must know what protocol is being used
- Common port numbers are 80 (for web pages), 23 (for telnet) and 25 and 110 (for mail)
- Port numbers above 1024 are available for other kinds of communication between our programs

Protocols

- In order for computers to communicate with one another, they must agree on a set of rules for who says what, when they say it, and what format they say it in
- This set of rules is a protocol
- Different programs can use different protocols
- Protocols may be in ASCII (characters) or in binary
- Some common protocols are HTTP (for web pages), FTP (for file transfer), and SMTP (Simple Mail Transfer Protocol)

HTTP

- HTTP stands for Hypertext Transfer Protocol.
- It is the standard protocol for transferring web pages (and their content) across the Internet.
- You may have noticed that when you browse a web page, the URL is preceded by "HTTP://".
- This is telling the web browser to use HTTP to transfer the data.
- Most browsers will default to HTTP if you don't specify it.
- Test this by typing in say... www.google.com (instead of http://www.google.com").

HTTPS

- HTTPS stands for Hypertext Transfer Protocol over Secure Socket Layer.
- Think of it as a secure version of HTTP.
- When you browse a web page using HTTPS, you are using SSL (Secure Sockets Layer).
- For a website to use HTTPS it needs to have an SSL certificate installed on the server.
- These are usually issued by a trusted 3rd party, referred to as a Certificate Authority
- When you browse a web page using HTTPS, you can check the details of the SSL certificate.
- For example, you could check the validity of it. You could also check that the website does actually belong to the organization you think it does. You can usually do this by double clicking on the browser's padlock icon. The padlock icon only appears when you view a secure site.

FTP

- FTP stands for File Transfer Protocol. It is used to transfer files across the Internet.
- FTP is commonly used by web developers to publish updates to a website (i.e. to upload a new version of the website).
- Where HTTP is used for displaying the file in your browser, FTP is used simply to transfer the file from one computer to a specified location on another computer.
- You can use FTP to transfer the files from your computer to a remote computer (such as a web server), or to transfer from the remote computer to your local computer.

TCP/IP

- The Internet (and most other computer networks) are connected through TCP/IP networks
- TCP/IP is actually a combination of two protocols:
  - IP, Internet Protocol, is used to move packets (chunks) of data from one place to another
    - Places are specified by IP addresses: four single-byte (0..255) numbers separated by periods
    - Example: `192.168.1.1`
  - TCP, Transmission Control Protocol, ensures that all necessary packets are present, and puts them together in the correct order
- TCP/IP forms a “wrapper” around data of *any* kind
- The data uses its own protocol, for example, FTP

Hostnames and DNS servers

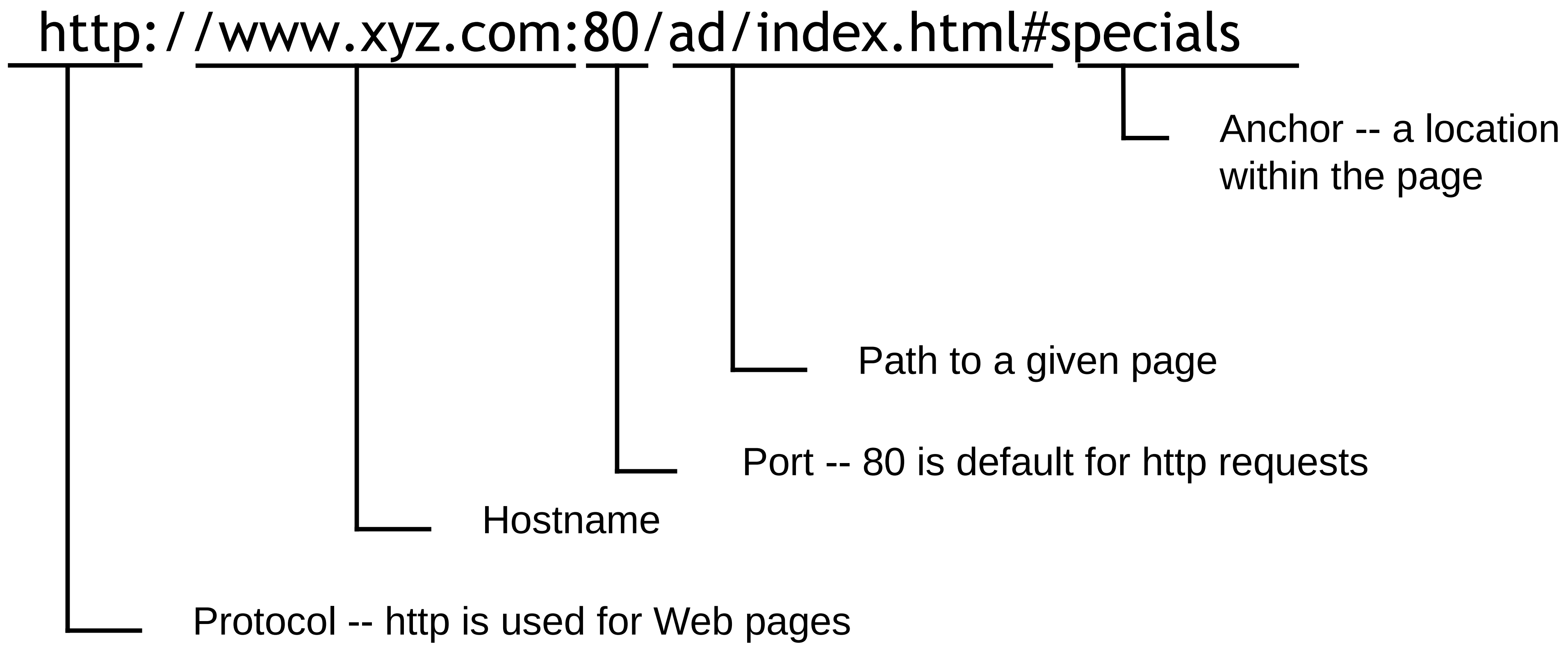
- The “real” name of a computer on the internet is its four-byte IP address
- People, however, don’t like to remember numbers, so we use hostnames instead
- For example, the hostname www.cis.upenn.edu is 158.130.12.9
- A DNS (Domain Name Server) is a computer that translates hostnames into IP addresses
  - Think of it as like a phone book--names to useful numbers
  - Of course, you have to know the IP address of the DNS in order to use it!
  - You usually get two DNS numbers from your Internet Service Provider (ISP)

DHCP

- If you have a web site, it must be hosted on a computer that is “permanently” on the Web
  - This computer must have a permanent IP address
  - There aren’t enough IP addresses for the number of computers there are these days
- If you have no permanent web site, you can be given a *temporary* (dynamically allocated) IP address each time you connect to the Web
- Similarly, if you have a home or office network, only one computer needs a permanent IP address
  - The rest of the computers can be assigned *internal,* permanent IP addresses (not known to the rest of the world)
  - They can also be assigned internal IP addresses dynamically
- DHCP (Dynamic Host Configuration Protocol) is a way of assigning temporary IP addresses as needed

URLs

- A URL, Uniform Resource Locater, defines a location on the Web
- A URL has up to five parts:


http://www.xyz.com:80/ad/index.html#specials
Anchor -- a location within the page
Path to a given page
Port -- 80 is default for http requests
Hostname
Protocol -- http is used for Web pages

Data Exchange Format

XML

JSON

RDF

Introducing: RDF

- Improve on PICS, HTML, and XML
- Machine *understandable* metadata
- Support structured values
- Support metadata bureaux
- Encourage authenticated metadata
- Base for a variety of descriptions:
  - cataloging, privacy, accessibility, IPR, ...

Data Integration

- Example:
  - “The author of a document is Paul”
  - “Paul is the author of a document”
  - “A document is authored by Paul”
  - “The **author** of a **document** is **Paul**”
- Representation(s) in XML:

```
<document
        href = “http://doc_url”
        author = “Paul”
        />
```

Data Integration

- Complexity of querying XML documents
  - N ways of mapping XML to logical structure
  - Requires the normalization of all possible representations for effective query
- Mean the same thing to a person
- Mean very different things to a machine
- RDF much less flexible
  - less flexible == more interoperable!
  - consistent way of representing statements

RDF Components

- Formal data model
- Syntax for interchange of data
- Schema Type system (schema model)
- Syntax for machine-understandable schemas
- Query and profile protocols

RDF Data Model

- Imposes structural constraints on the expression of application data models
  - for consistent <u>encoding</u>, <u>exchange</u> and <u>processing</u> of metadata
- Enables resource description communities to define their own semantics
- Provides for structural interoperability

What is JSON

- JSON is a data interchange format
- Interactive Web 2.0 applications, no more use page replacement. Data transfer without refreshing a page.
- The most important aspects of data transfer are simplicity, extensibility, interoperability, openness and human readability
- Key idea in AJAX – Asynchronous Java Script and XML.

Topics to cover

- Overview of the working of JSON
- Properties of JSON as a data format
- JSON with AJAX
- Advantages of using JSON with AJAX
- Security Concerns in using JSON
- Where does it fit the best

How does JSON work?

- JSON is a subset of Java Script. JSON can be parsed by a Java Script parser.
- It can represent either complex or simple data as it has data types
- They are Strings, Number, Boolean, Objects and Arrays
- E.g. of Object:
- { "name": "Jack Nimble", "format": { "type": "rect", "width": 120, "interlace": false}}

How does JSON work?

- An array can be shown as
- ["Sunday", "Monday", "Tuesday", "Wednesday“]
- All data types are intuitive and similar to other programming languages
- Also compatible with other languages like C, C++, C#, ColdFusion, Python and many more.

Properties of JSON

- It's simultaneously human- and machine-readable format.
- It has support for Unicode, allowing almost any information in any human language to be communicated;
- The self-documenting format that describes structure and field names as well as specific values.
- The strict syntax and parsing requirements that allow the necessary parsing algorithms to remain simple, efficient, and consistent;
- The ability to represent the most general computer science data structures: records, lists and trees.

JSON in AJAX

- JSON can be used in AJAX as follows:
- Include it in HTML directly
- <html>... <script>
  var data = *JSONdata*;
  </script>... </html>
- JSON is used with XMLHttpRequest and can be converted into a JavaScript structure
- responseData = eval('(' + responseText + ')');

JSON in AJAX

- Another approach is to use an invisible <iframe> for data communication.
- The server sends JSON text embedded in a script in a document.
- <html><head><script> document.domain = ‘something.com’; parent.deliver(*JSONtext*); </script></head></html>
- deliver is passed the incoming data structure.

Why is JSON better suited for AJAX?

JSON is widely used in AJAX. The X in AJAX stands for XML.

E.g.

```
{
"fullname": "Jeevan Thapa",
"org": "NCIT",
}

<?xml version='1.0‘ encoding='UTF-8'?>
<element>
<fullname>Jeevan Thapa</fullname>
<org>NCIT</org>
</element>
```

JSON vs XML

```
{
        "id": 3,
        "bbox": [
            83.9187728578849,
            26.9191431698797,
            86.5727626547517,
            28.3862845807188
        ],
        "centroid": {
            "type": "Point",
            "coordinates": [
                85.47056392270287,
                27.67865196906648
            ]
        },
        "title": "Bagmati",
        "title_en": "Bagmati",
        "title_ne": "बाग्मती",
        "code": "bagmati",
        "order": 1
    }
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <id>3</id>
  <bbox>83.9187728578849</bbox>
  <bbox>26.9191431698797</bbox>
  <bbox>86.5727626547517</bbox>
  <bbox>28.3862845807188</bbox>
  <centroid>
    <type>Point</type>

<coordinates>85.47056392270287</coordinates>

<coordinates>27.67865196906648</coordinates>
  </centroid>
  <title>Bagmati</title>
  <title_en>Bagmati</title_en>
  <title_ne>बाग्मती</title_ne>
  <code>bagmati</code>
  <order>1</order>
</root>
```

Why is JSON better suited for AJAX?

- JSON response at client side is:
- *var name = eval('(' + req.responseText + ')').fullname.value;*
- To access a composite element
- *eval('(' + req.responseText + ')').xyz.abc.value;*
- Thus, any level deep elements can be easily accessed.

Why is JSON better suited for AJAX?

- XML response at client side is:
- *var root = req.responseXML;*
- *var name = root.getElementsByTagName(‘fullname’);*
- To access a composite element
- *root.getElementsByTagName(‘xyz’)[0].firstChild*
- To access deeper levels we need more overhead.
- Reduced extensibility in XML

Security Concerns

- Same Origin Policy - JavaScript to access the contents of a Webpage, both the JavaScript and the Web page must originate from the same domain.
- Malicious website could serve up JavaScript that loads sensitive information from other websites using a client's credentials and communicates it back to the attacker.

Security Concerns

- Although the malicious JavaScript can’t directly manipulate the contents, it can view the execution of the JavaScript and store the results it returns.
- This problem gets aggravated with JSON as the JSON arrays are themselves JavaScript objects and any malicious user can view them directly.

Web Services Components

- **XML** – eXtensible Markup Language – A uniform data representation and exchange mechanism.
- **SOAP** – Simple Object Access Protocol – A standard way for communication.
- **UDDI** – Universal Description, Discovery and Integration specification – A mechanism to register and locate WS based application.
- **WSDL** – Web Services Description Language – A standard meta language to described the services offered.

Example – A simple Web Service

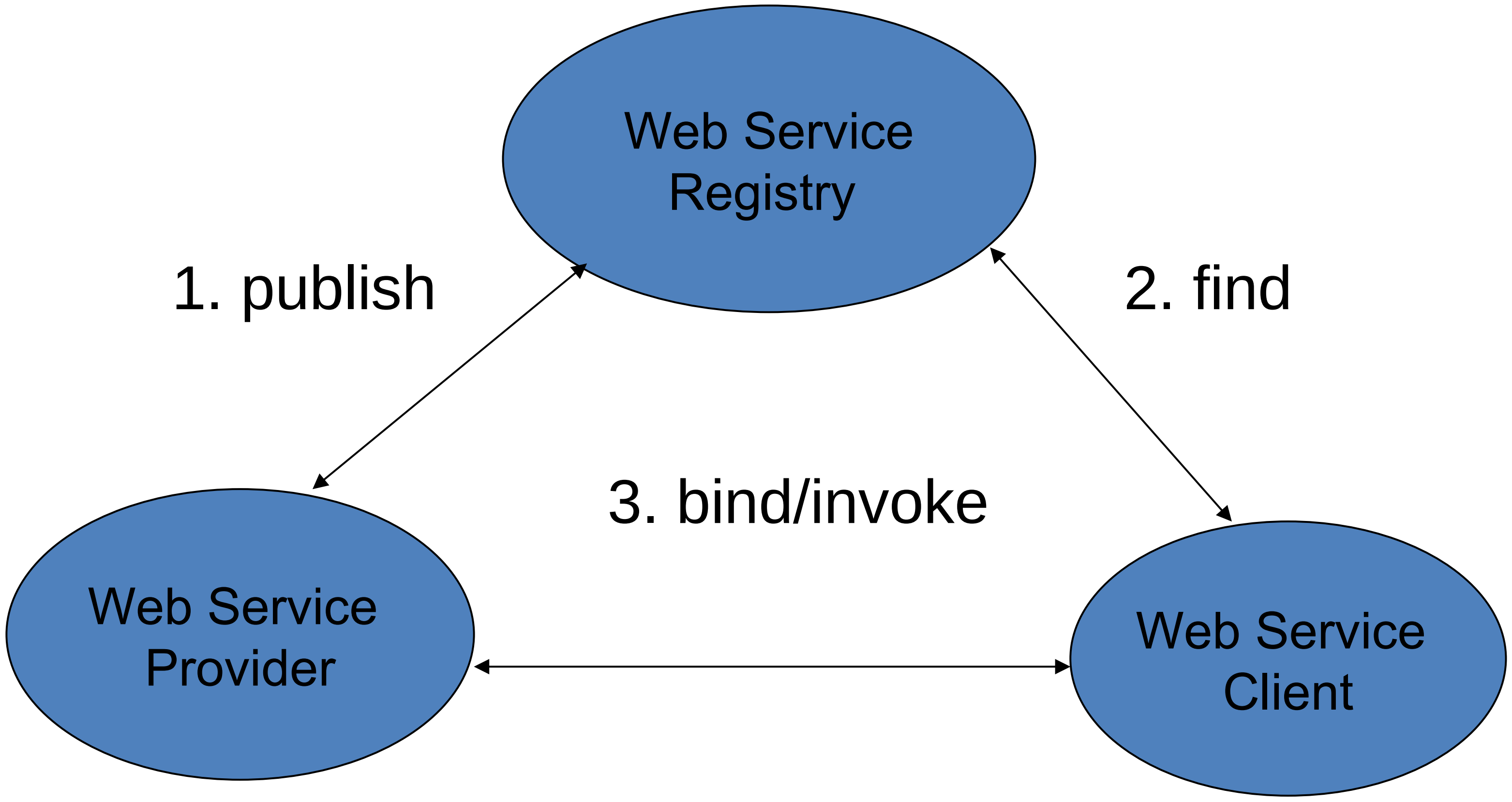
- A buyer (which might be a simple client) is ordering goods from a seller service.
- The buyer finds the seller service by searching the UDDI directory.
- The seller service is a Web Service whose interface is defined using Web Services Description Language (WSDL).
- The buyer is invoking the order method on the seller service using Simple Object Access Protocol (SOAP) and the WSDL definition for the seller service.
- The buyer knows what to expect in the SOAP reply message because this is defined in the WSDL definition for the seller service.

The Web Service Model

- The Web Services architecture is based upon the interactions between three roles:
  - Service provider
  - Service registry
  - Service requestor
- The interactions involve the:
  - Publish operations
  - Find operation
  - Bind operations.

The Web Service Model

The Web Services model follows the *publish*, *find*, and *bind* paradigm.



XML

- XML stands for E**X**tensible **M**arkup **L**anguage.
- XML is a **markup language** much like HTML.
- XML was designed to **describe data**.
- XML tags are not predefined.
- You must **define your own tags.**
- The prefect choice for enabling cross-platform data communication in Web Services.

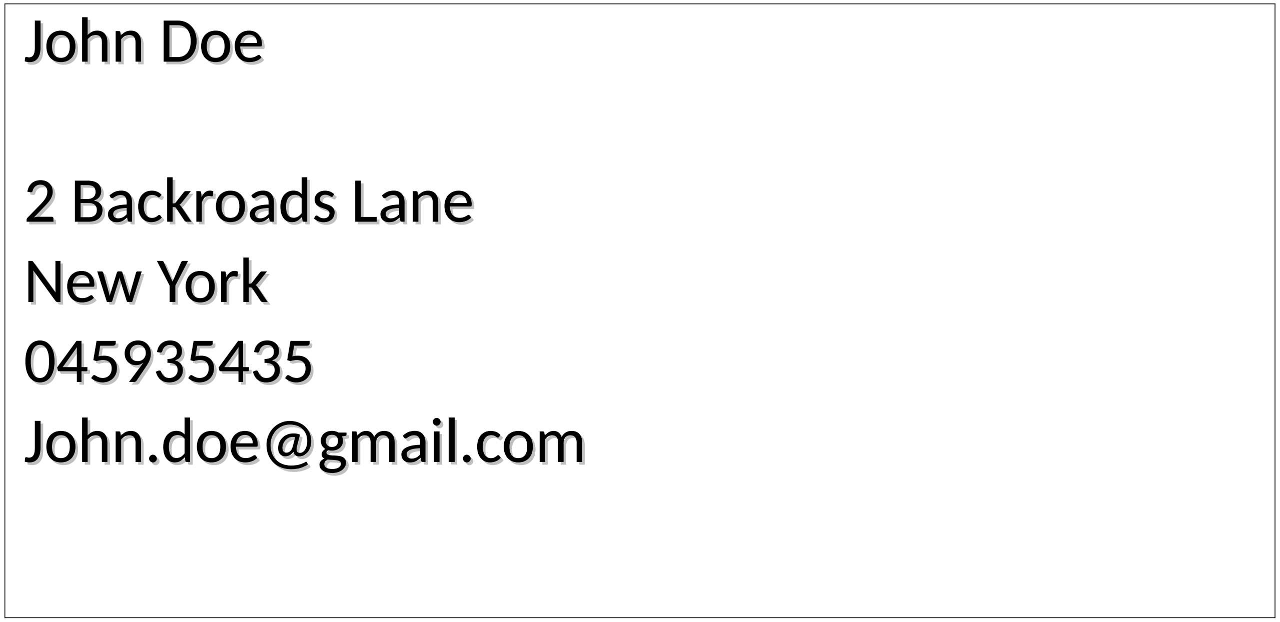
XML vs HTML

An HTML example:

```
<html>
<body>
    <h2>John Doe</h2>
    <p>2 Backroads Lane<br>
          New York<br>
          045935435<br>
          john.doe@gmail.com<br>
          </p>
</body>
</html>
```

XML vs HTML

- This will be displayed as:



John Doe

2 Backroads Lane

New York

045935435

John.doe@gmail.com

- HTML specifies how the document is to be displayed, and not what information is contained in the document.
- Hard for machine to extract the embedded information. Relatively easy for human.

XML vs HTML

- Now look at the following:

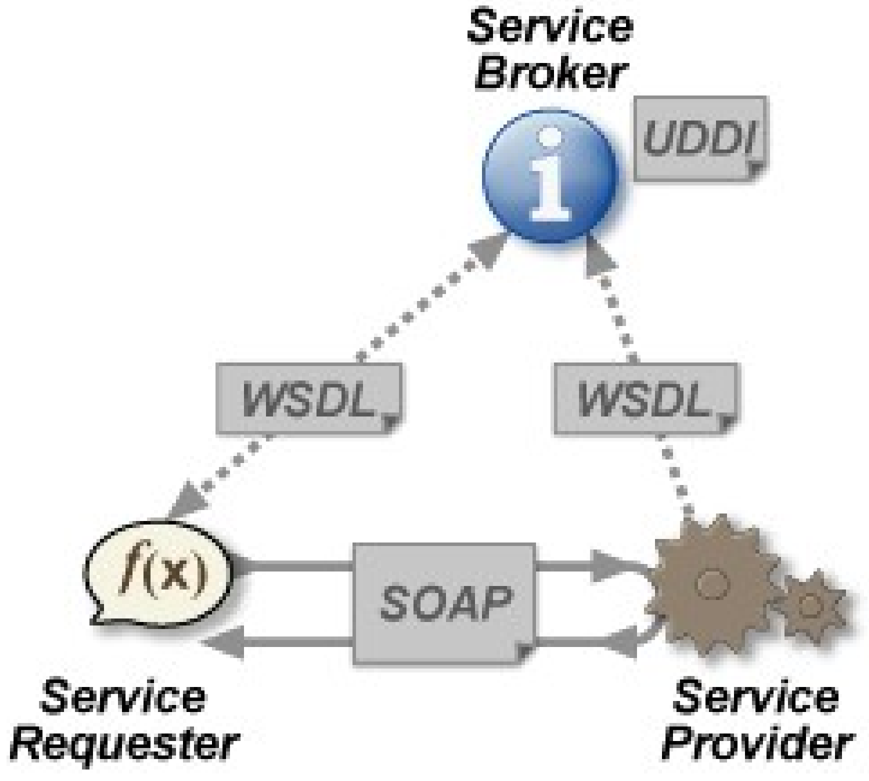
```
<?xml version=1.0?>
<contact>
  <name>John Doe</name>
  <address>2 Backroads Lane</address>
  <country>New York</country>
  <phone>045935435</phone>
  <email>john.doe@gmail.com</email>
</contact>
```

- In this case:
  - The information contained is being marked, but not for displaying.
  - Readable by both human and machines.

Architectures for Web Services

Overall Architecture

- UDDI
  - Information on available web service
- WSDL
  - A description of how to communicate using the web service
- SOAP
  - Protocol for exchanging messages



Universal Description, Discovery, and Integration (UDDI)

- Platform-independent XML registry
- Allows businesses to list services they provide
- Registration consists of:
- While pages info – real address and contact information
- Yellow pages info – industrial categorization
- Green pages info – technical information on exposed services

Web Services Description Language (WSDL)

- XML format for describing public interface of web services
  - Services are collection of abstract endpoints called “ports”
  - Each port has a protocol (“binding”) and address
  - Each port has a type that defines valid “operations”
  - An operation consists of messages and data formats

WSDL document describes:

- Data formats
- Valid messages
- Ports types with their supported operations
- Binding of ports to types and addresses
- Services in terms of ports they provide and documentation

Simple Object Access Protocol (SOAP)

- Lightweight protocol for message exchange
  - Enable “access” to objects, including RPCs
  - Defines formats for requests, responses, errors
- XML based, runs on top of HTTP
- Optional header with information on
  - Security requirements
  - Routing
  - Transactions
- Body contains actual data

Simple SOAP example

- Request:

  -
    ```
    <soap:Envelope
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
       <soap:Body>
         <getProdDetails xmlns="http://warehouse.example.com/ws">
           <productId>827635</productId>
         </getProdDetails>
       </soap:Body>
    </soap:Envelope>
    ```
- Response:

  -
    ```
    <soap:Envelope
     xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Body>
    <getProdDetailsResponse xmlns="http://warehouse.example.com/ws">
       <getProductDetailsResult>
            <productName>Toptimate 3-Piece Set</productName>
            <productId>827635</productId>
            <description>3-Piece luggage set Polyester</description>
            <price>96.50</price>
            <inStock>true</inStock>
        </getProductDetailsResult> </getProductDetailsResponse>
    </soap:Body>
    </soap:Envelope>
    ```

SOAP Characteristics

- SOAP has three major characteristics:
  - Extensibility – security and WS-routing are among the extensions under development.
  - Neutrality - SOAP can be used over any transport protocol such as HTTP, SMTP or even TCP.
  - Independent - SOAP allows for any programming model .

SOAP Building Blocks

A SOAP message is an ordinary XML document containing the following elements:

- A required Envelope element that identifies the XML document as a SOAP message.
- An optional Header element that contains header information.
- A required Body element that contains call and response information.
- An optional Fault element that provides information about errors that occurred while processing the message.

SOAP Request

```
POST /InStock HTTP/1.1
Host: www.stock.org
Content-Type: application/soap+xml; charset=utf-8 Content-Length: 150

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle=http://www.w3.org/2001/12/soap-encoding”>

   <soap:Body xmlns:m="http://www.stock.org/stock">
         <m:GetStockPrice>
              <m:StockName>IBM</m:StockName>
         </m:GetStockPrice>
   </soap:Body>
</soap:Envelope>
```

SOAP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap; charset=utf-8
Content-Length: 126

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Body xmlns:m="http://www.stock.org/stock">
        <m:GetStockPriceResponse>
                <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>
</soap:Envelope>
```

SOAP Security

- SOAP uses HTTP as a transport protocol and hence can use HTTP security mainly HTTP over SSL.
- But, since SOAP can run over a number of application protocols (such as SMTP) security had to be considered.
- The WS-Security specification defines a complete encryption system.

WSDL

- WSDL stands for Web Services Description Language.
- WSDL is an XML vocabulary for describing Web services.
- It allows developers to describe Web Services and their capabilities, in a standard manner.
- WSDL specifies what a request message must contain and what the response message will look like in unambiguous notation.
- In other words, it is a contract between the XML Web service and the client who wishes to use this service.
- In addition to describing message contents, WSDL defines where the service is available and what communications protocol is used to talk to the service.

The WSDL Document Structure

- A WSDL document is just a simple XML document.
- It defines a web service using these major elements:
  - **port type** - The operations performed by the web service.
  - **message -** The messages used by the web service.
  - **types -** The data types used by the web service.
  - **binding -** The communication protocols used by the web service.

WSDL Document

```
<message name="GetStockPriceRequest">
    <part name="stock" type="xs:string"/>
 </message>
<message name="GetStockPriceResponse">
    <part name="value" type="xs:string"/>
 </message>

<portType name=“StocksRates">
      <operation name=“GetStockPrice">
            <input message=“GetStockPriceRequest"/>
            <output message=“GetStockPriceResponse"/>
      </operation>
</portType>
```

What is ebXML?

- ebXML = Electronic Business XML
- Global Standard for electronic business
- ebXML enables anyone, anywhere to do business with anyone else over the Internet
- Specifically designed to support SME
- Complementary to existing B2B initiatives

  (UDDI, RosettaNet, TradeXchange, etc.)

An end-to-end B2B XML Framework

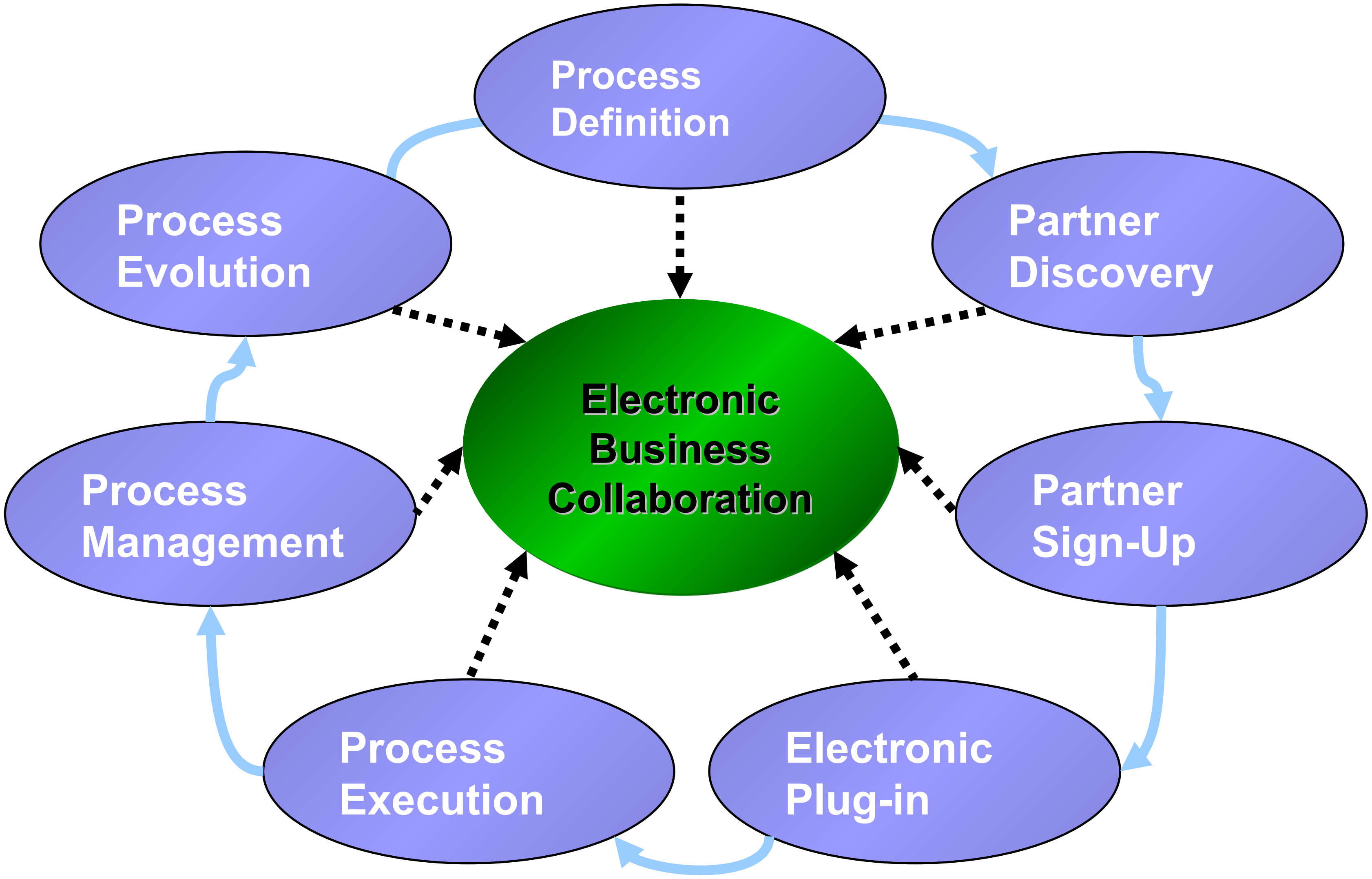
ebXML Vision

- A global electronic market place where enterprises of any size, anywhere can:
  - Find each other electronically
  - And conduct business
    - Using XML messages
    - According to standard business process sequences
    - With clear business semantics
    - According to standard or mutually agreed trading partner protocol agreements
    - Using off the shelf purchased business applications

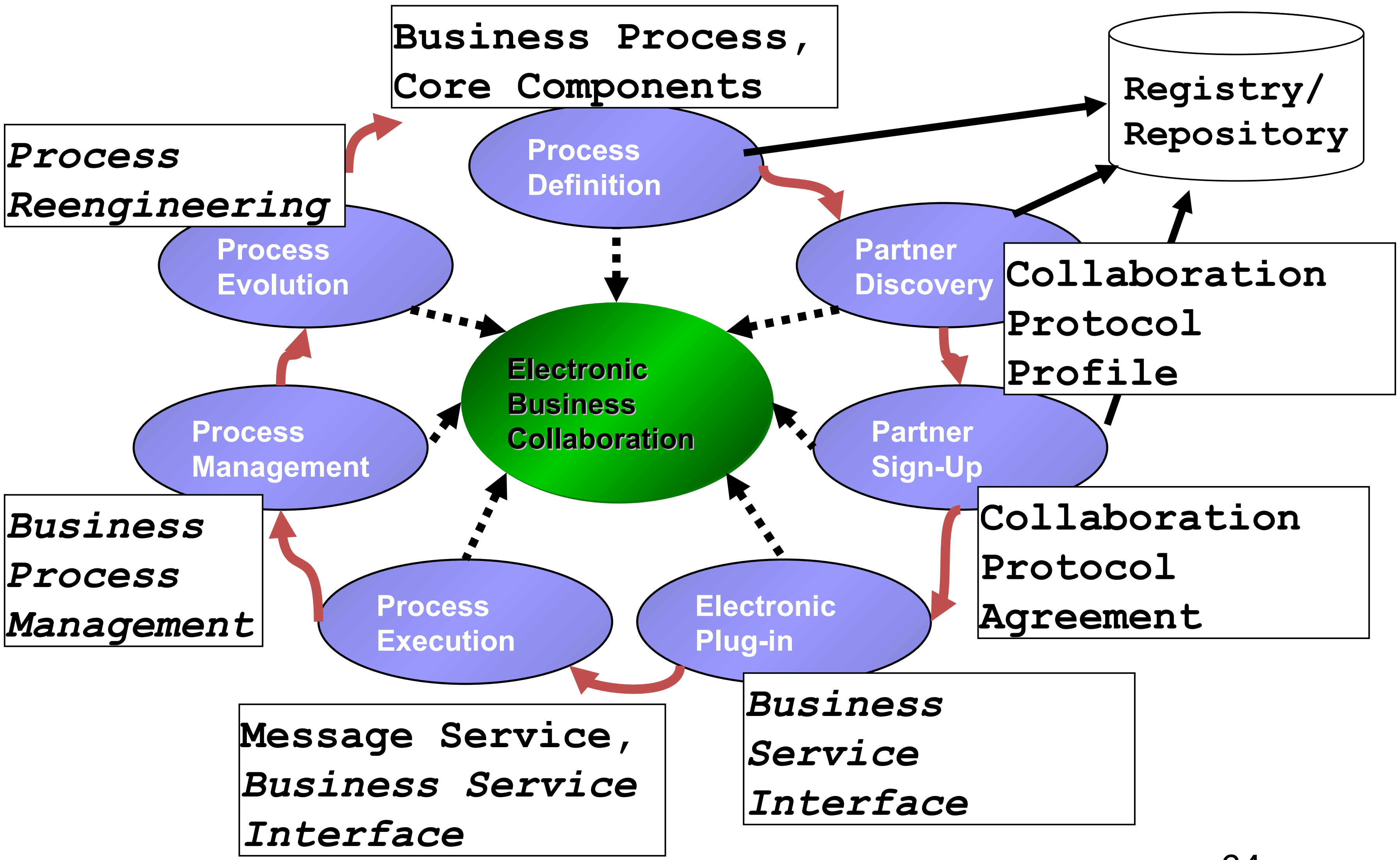
B2B Collaboration

- B2B collaboration requires more than just an XML protocol and a service registry
- You have to deal with
  - Business semantics
  - Negotiating terms and conditions
  - Interoperability
  - Security and Privacy
  - Reliability
- ebXML provides concrete specifications to enable dynamic B2B collaborations

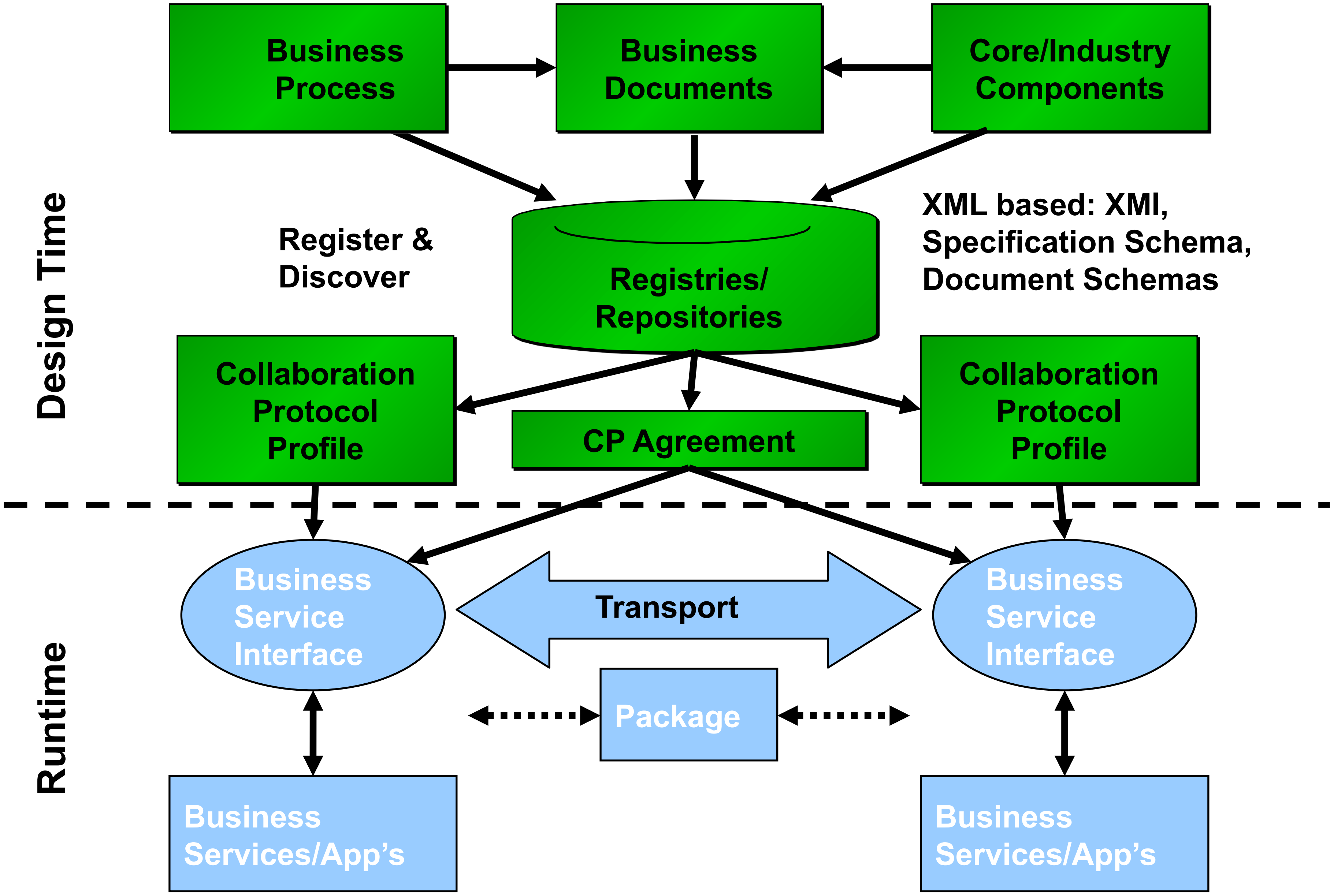
B2B Collaboration Process



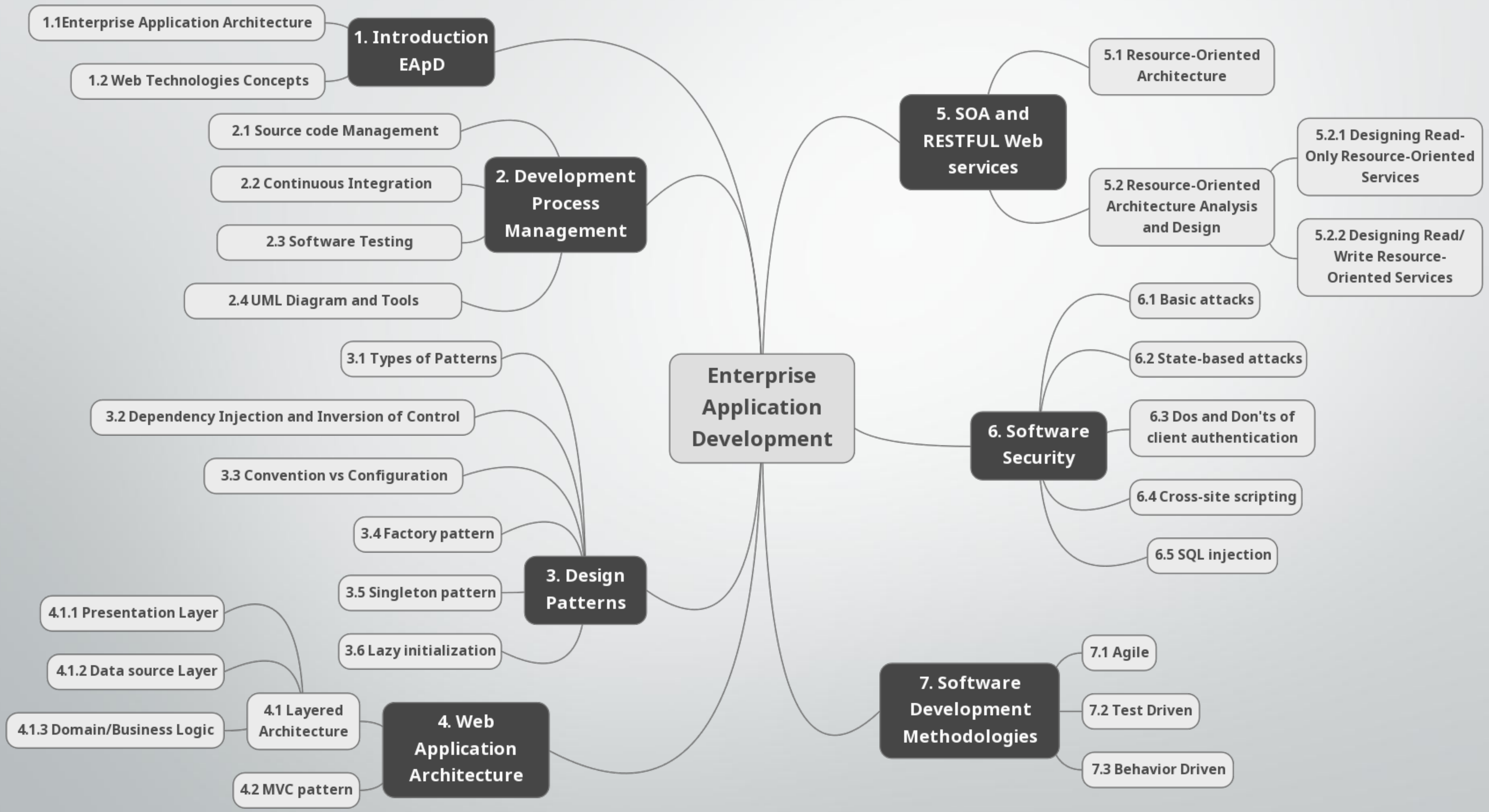
ebXML Specifications


Business Process, Core Components
Registry/ Repository
Process Reengineering
Process Definition
Process Evolution
Partner Discovery
Collaboration Protocol Profile
Electronic Business Collaboration
Process Management
Partner Sign-Up
Collaboration Protocol Agreement
Business Process Management
Process Execution
Electronic Plug-in
Message Service, Business Service Interface
Business Service Interface

ebXML Architecture



Enterprise Application Development


Enterprise Application Development
1. Introduction EApD
1.1Enterprise Application Architecture
1.2 Web Technologies Concepts
2. Development Process Management
2.1 Source code Management
2.2 Continuous Integration
2.3 Software Testing
2.4 UML Diagram and Tools
3. Design Patterns
3.1 Types of Patterns
3.2 Dependency Injection and Inversion of Control
3.3 Convention vs Configuration
3.4 Factory pattern
3.5 Singleton pattern
3.6 Lazy initialization
4. Web Application Architecture
4.1 Layered Architecture
4.1.1 Presentation Layer
4.1.2 Data source Layer
4.1.3 Domain/Business Logic
4.2 MVC pattern
5. SOA and RESTFUL Web services
5.1 Resource-Oriented Architecture
5.2 Resource-Oriented Architecture Analysis and Design
5.2.1 Designing Read-Only Resource-Oriented Services
5.2.2 Designing Read/Write Resource-Oriented Services
6. Software Security
6.1 Basic attacks
6.2 State-based attacks
6.3 Dos and Don'ts of client authentication
6.4 Cross-site scripting
6.5 SQL injection
7. Software Development Methodologies
7.1 Agile
7.2 Test Driven
7.3 Behavior Driven

Resources

- https://www.w3.org/TR/PR-rdf-syntaxhttps://www.w3.org/TR/PR-rdf-syntax
- www.**ebxml**.org/
- http://msdn.microsoft.com/webservices/understanding/webservicebasics/default.aspx
- http://www.w3schools.com/
- http://uddi.microsoft.com/Default.aspx
- http://www.developer.com/services/article.php/2195981
- Many more on the web…