MASARYKOVA UNIVERZITA

PV213 Enterprise Information Systems in Practice

08 – Integration of EIS with other systems

MASARYKOVA UNIVERZITA

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Reasons to integrate systems

- Increasing productivity of users (decreasing costs)
- Avoiding duplicate and mismatched information
- Avoiding mistakes made by users
- Incremental grow of the infrastructure
  - You can buy systems steps by step as you need them
  - Systems can be provided by different vendors
- Avoiding vendor lock-in
- Reducing complexity of the one big system
  - Smaller specialized sub-systems are easier to manage

Possible approaches for integration

- Without any additional integration system (star integration)
  - Simplest solution
  - Each system can be directly connected to other system
- Mediation (enterprise service bus)
  - Integration system acts as a broker between systems
- Federation
  - Integration system acts as a façade across systems

Mediation and federation allows you to better control flow of control or data between systems

- You can define rules which system can call other system (rights)
- You have an better centralized control what systems can do

File based integration

- The oldest and simplest but still used integration method
- Used mainly for transferring data from one system to another (but files can hold also information about triggering “events”)
- How it works
  - One system generates file (or set of files) at given time
  - Another system (or more systems) afterwards reads file (or set of files)
- All systems must understand the file format
- Compatibility of file formats (versioning) must be solved
- Files cannot be generated too often (performance reasons)
- Synchronization is often done once per day during the night
  - In global world is hard to specify “night”

File based integration – CSV format

- Simple, human readable
- No special parser needed
- Can contain “header line” in the beginning of file
- Ideal for tabular data
- Handling structured data is more complicate
- Separators used as value must be handled specially
- Versioning possible with some limits
  - E.g. new data are added to the end of line
  - Two different systems then can (in theory) read old and new format
- Hard to transfer sensitive data which should not be seen by third parties

File based integration – XML format

- Still (quite) simple, still (mostly) human readable
- XML parser needed, can be problems with performance of big files
- Ideal for structured data
- Versioning for new data solved automatically by definition of XML
  - New data added as additional elements or attributes
- Correctness of the XML can be checked against XML schema
- Files are bigger because they contain metadata
- Again hard to transfer sensitive data which should not be seen by third parties (but encryption of some attributes or elements possible)
- For checking authenticity of the information there exists standard for signing XML files

File based integration – Proprietary format

- Can be textual or binary
- Harder to read for humans (especially binary format)
- You need special parsers
- In binary format versioning is more complicate
  - Usually you need some version tag in the beginning
- Used in cases when you need to integrate with some old exotic system
- Don’t use it unless you have a good reason to do it (e.g. performance)

Database based integration

- Similar to file based integration but exchange of data is done via database
- How it works
  - One system is primary and uses database for storing its data
  - Other systems read data from this database
- Better handling of rights than for file based integration
  - Databases supports different users with different rights to tables (read, read and write)
- Consistency of data solved by database per definition
- Other systems must use special libraries for accessing given database type
- Other systems are directly dependant on given database type
  - Migration to another database vendor can be problematic
- Special table in the database can simulate queue

Platform specific integrations

- Integration of systems on the same platform
- Proprietary formats
  - Very hard then to integrate with other platforms
- Often problems with passing calls through firewalls
- CORBA (Common Object Request Broker Architecture)
  - In theory platform independent
  - Different versions, incompatibilities
  - Supported only on some platforms
  - Often used only for integration on the same platform
- Java
  - Remote Method Invocation (RMI) – based on CORBA
- .NET
  - .NET Remoting

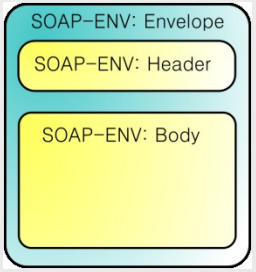
Web based integrations

- Integrations using standard web technologies
  - HTTP(s) protocol
- “Online” integration
- Synchronous or asynchronous
- Firewall friendly
- Supported on wide range of platforms
- Lot of tools
  - Faster development
  - Relatively easy to find possible problems
- Some web protocols supports also advanced features (security, transactions, etc.)

Web based integrations – Web services

- Term **web service** is usually associated with protocol SOAP (Single Object Access Protocol) but you can consider it as more general – we use now former definition
- Web services is a try to standardize machine-to-machine communication via network
- Based on web technologies
- Uses SOAP for message exchange and WSDL (Web Service Definition Language) as metadata for describing message
- Supported on all major platforms
- Sometimes criticized for too complexity

Web based integrations - Single Object Access Protocol (SOAP)



- Uses XML for encoding the message
- Consists of free parts
  - **Envelope** specifies what is in message
  - Optional **header** contains application specific information (authentication, etc.)
  - Mandatory **body** contains message data itself (method call)
- As a transport protocol uses mainly HTTP but other protocols can be used as well (SMTP)
- Extensible
- Supports advanced features (signing of messages, transactions, etc.)

Web based integrations - SOAP example

SOAP via HTTP (getting last trade price)

- HTTP header (POST)
- HTTP body is SOAP message

```
POST /InStock HTTP/1.1
Host: www.example.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetLastTradePrice xmlns:m="http://www.example.com/stock">
      <m:TradePriceRequest>GOOG</m:TradePriceRequest>
    </m:GetLastTradePrice>
  </soap:Body>
</soap:Envelope>
```

Web based integrations – WSDL, UDDI

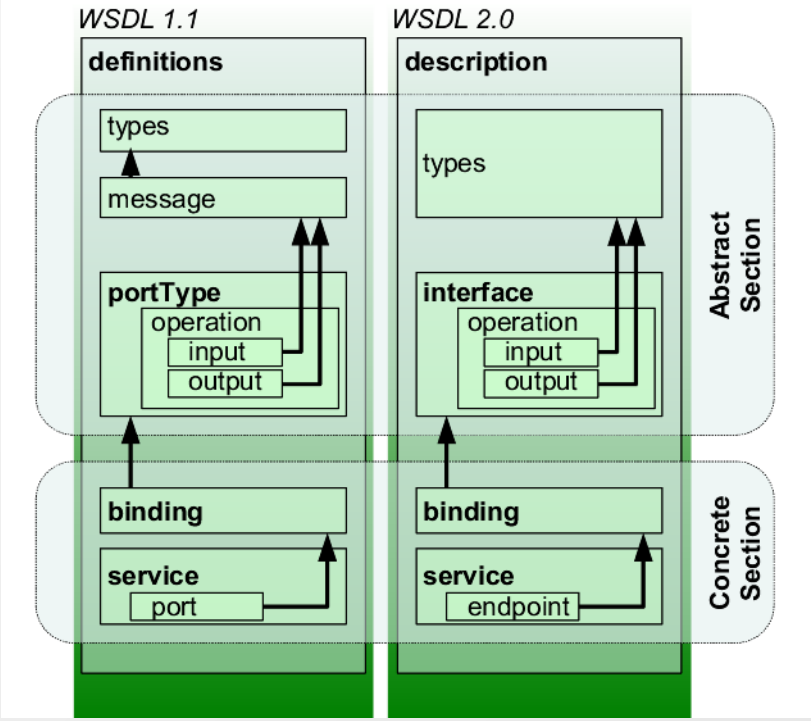
- WSDL (Web Service Definition Language) is a metadata for the service – it describes service itself in machine readable XML format
- Two approaches how to deal with WSDL
  - You can generate WSDL from the code which does your service
  - You start first with WSDL and then you implement the srvice
- From WSDL you can automatically generate stubs for calling web service (stub behaves like any other object or procedure in your favorite language)
- UDDI (Universal Description Discovery and Integration) was a try to do a central registry to allow searching for web services
- Basic idea was that there can be “market for web services” which can be searched and used automatically by machines
- In reality UDDI is not used
  - Some people argues that business is done between humans and not between machines

Web based integrations - WSDL example for SOAP 1.1 over HTTP

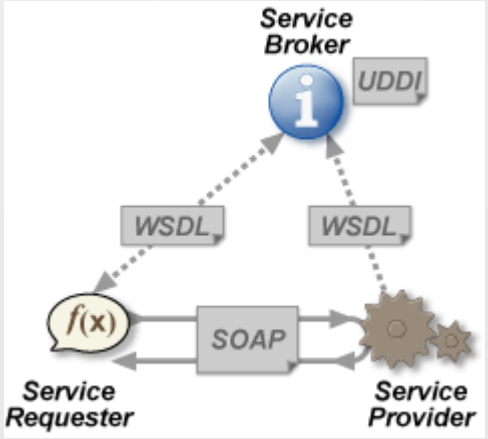
```
<?xml version="1.0"?>
<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
       <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="TradePriceRequest">
              <complexType>
                  <all>
                      <element name="tickerSymbol" type="string"/>
                  </all>
              </complexType>
           </element>
           <element name="TradePrice">
              <complexType>
                  <all>
                      <element name="price" type="float"/>
                  </all>
              </complexType>
           </element>
       </schema>
    </types>
    <message name="GetLastTradePriceInput">
        <part name="body" element="xsd1:TradePriceRequest"/>
    </message>
    <message name="GetLastTradePriceOutput">
        <part name="body" element="xsd1:TradePrice"/>
    </message>
    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
    </portType>
    <binding name="StockQuoteSoapBinding"
      type="tns:StockQuotePortType">
        <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetLastTradePrice">
           <soap:operation
      soapAction="http://example.com/GetLastTradePrice"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
           </output>
        </operation>
    </binding>
    <service name="StockQuoteService">
        <documentation>My first service</documentation>
        <port name="StockQuotePort"
      binding="tns:StockQuoteBinding">
           <soap:address
      location="http://example.com/stockquote"/>
        </port>
    </service>
</definitions>
```

Web based integrations - WSDL, UDDI II

WSDL 1.1 and 2.0 differences



Integration of UDDI



Web based integrations - Representational State Transfer (REST)

- Called also RESTful web services
- Alternate approach to SOAP based web services
- In comparison with SOAP it simplifies the whole mechanism
- Uses standard HTTP methods for CRUD (Create/Read/Update/Delete) operations
  - POST for creating new resource
  - GET for getting existing resource or listing resources in collection
  - PUT for updating resource
  - DELETE for deleting of resource
  - HEAD for special operations (e.g. getting metadata about resource)
- Uses URL for location of resource
- Doesn't use any metadata like WSDL for SOAP web services
- Doesn't use any registry like UDDI for SOAP web services

Web based integrations - REST example

- Getting latest trade price for GOOG
  - GET http://example.com/LastTradePrice/GOOG
- Updating last trade price for GOOG
  - PUT http://example.com/LastTradePrice/GOOG
  - Value is in the body of the HTTP request
- Getting latest trade prices for all companies
  - GET http://example.com/LastTradePrice
- Getting latest trade prices for all companies starting with G
  - GET http://example.com/LastTradePrice?name=G*

Note: Of course security has to be taken into considerations for this example (can be solved by standard HTTP authentication mechanisms).

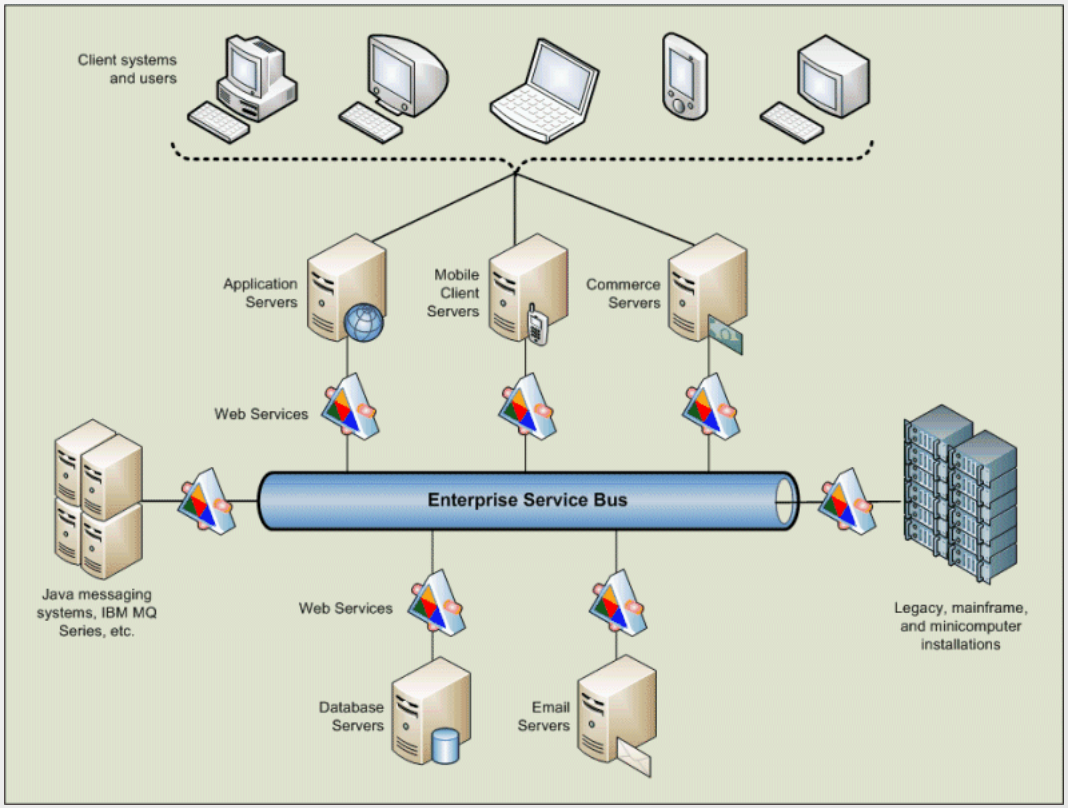
Web based integrations – proprietary HTTP

- Proprietary variant of RESTful web services
- Doesn't strictly follow CRUD (Create/Read/Update/Delete) semantics
- Usually uses only HTTP GET and POST methods
- For variable parameters use usually query parameters instead part of the URL
- All other features are the same as for RESTful web services
- Example
  - GET http://example.com/GetLastTradePrice?name=GOOG
  - POST http://example.com/SetLastTradePrice
    - Name of the stock and value are in the body of the HTTP request

Service Oriented Architecture (SOA)

- Approach how to design systems
  - Services should be high-level and focused on business and not on technology
- Benefits
  - Reusability – e.g. you can use same business logic for different clients (web client, fat client, mobile client)
  - Loose coupling – it is possible to replace services incrementally
  - Composability – you can combine several services and create new service
  - Business value – you can sell service to thirds parties

Enterprise service bus (ESB) I



Enterprise service bus (ESB) II

- Communication is asynchronous and message oriented
- Decouples systems from each other
  - It is then easy to replace one system by another (if interface is the same)
- How it works
  - One system sends message to another system
  - ESB can transform the message to the format receiver understand
  - ESB informs receiver about the message
- ESB supports one to one and one to many communication
- Messages can be easily controlled and monitored
  - Filtering of messages
  - Re-sending of messages in case of temporary unavailable system
- Messages can be orchestrated by ESB
  - Flow of messages through different systems (BPEL - Business Processing Execution Language)

Děkuji za pozornost.

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.

EVROPSKÁ UNIE esf

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost 2007-13

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ