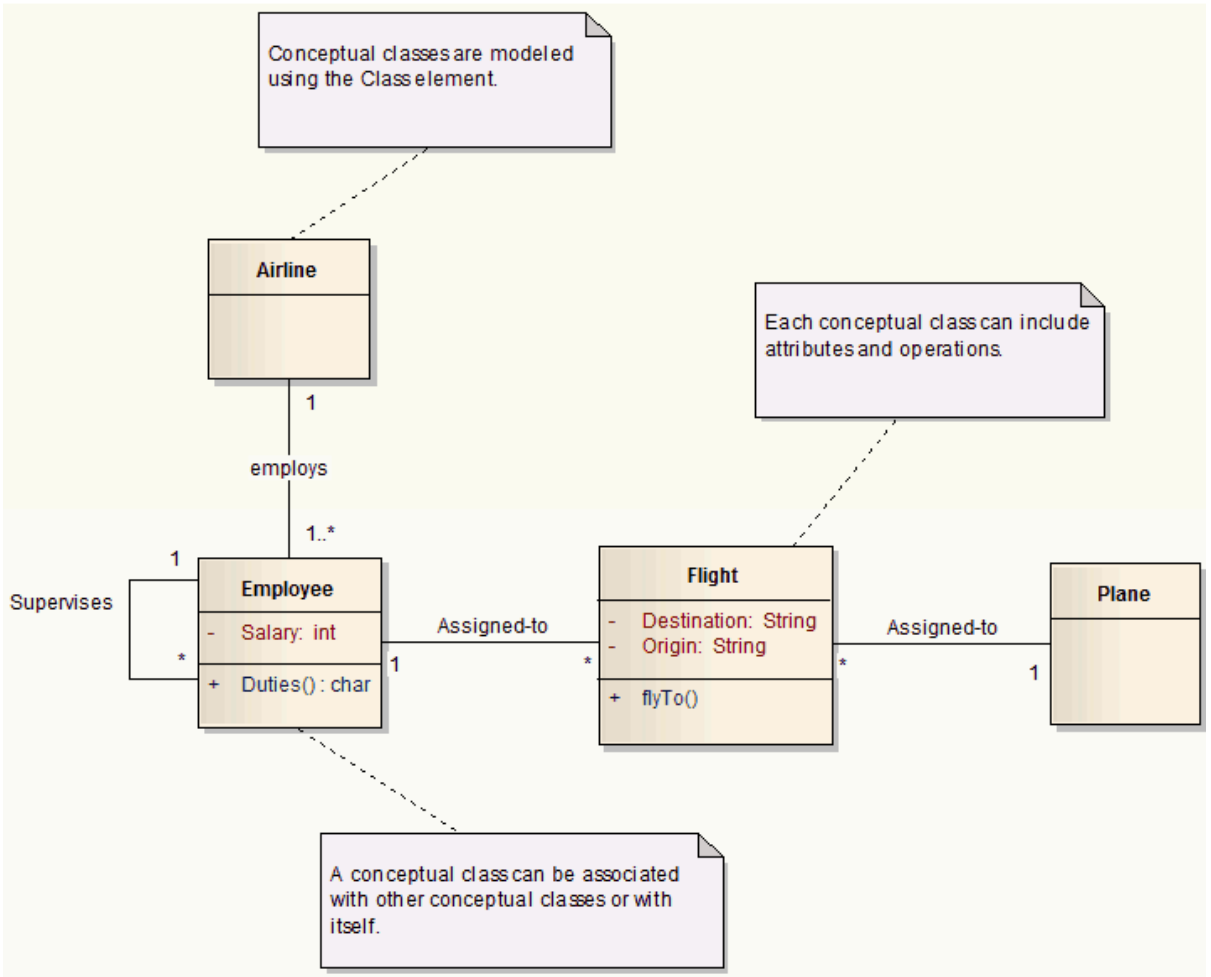
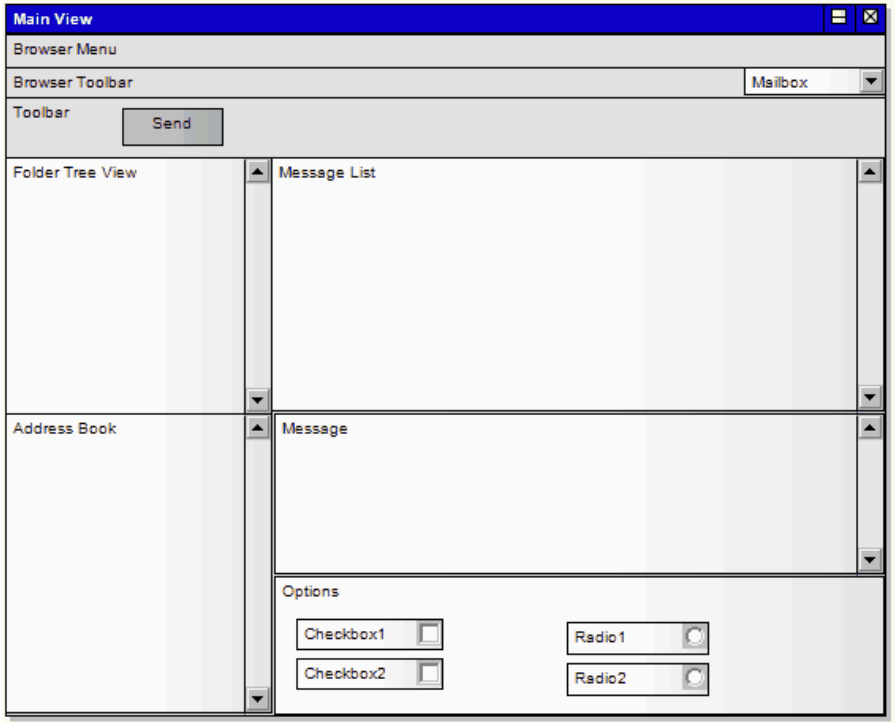
Doménový model (Domain model)

- Doménový model je koncepčný model, ktorý definuje fyzické a abstraktné objekty ktoré sa v projekte /doméne vyskytujú
- Dokumentuje vzťahy medzi zodpovednosťami koncepčných tried
  - to sú iné triedy ako tie z OOP
  - reprezentujú koncepciu skupiny vecí
- Definuje pojmy, ktoré sa v projekte vyskytujú
- Doménový model ukazuje:
  - fyzické a organizačné jednotky
  - vzťah medzi týmito jednotkami
  - multiplicitu týchto vzťahov
- Modeluje sa pomocou
  - diagramu tried
  - E-R diagramom
- Varianta je **Biznis doménový model**
  - Poskytuje **biznis slovník** – pojmy a fakty na základe ktorých môžu byť definované biznis pravidlá



UI Diagram

- Prispôsobené (customized) UML diagramy, ktoré vizuálne reprezentujú používateľské rozhranie pomocou riadicich prvkov, nadpisov, ...
- Niekedy sa robí „drotený model“ (wireframe)
- V návrhu je potrebné okrem popisu prvkov obrazoviek navrhnúť aj prechody medzi obrazovkami



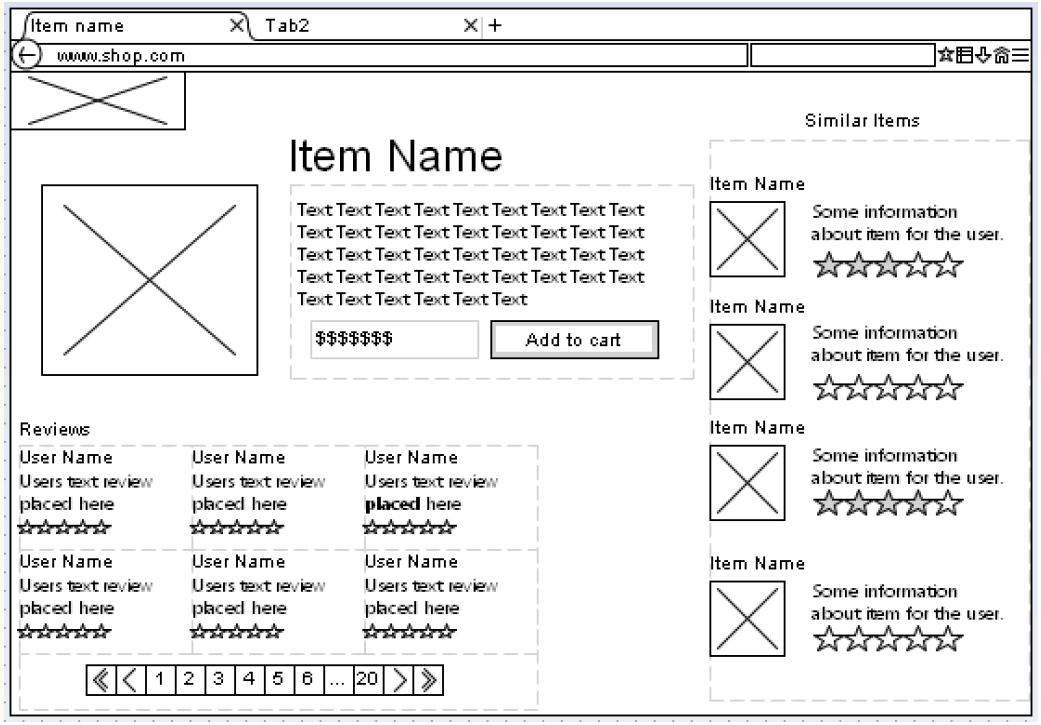
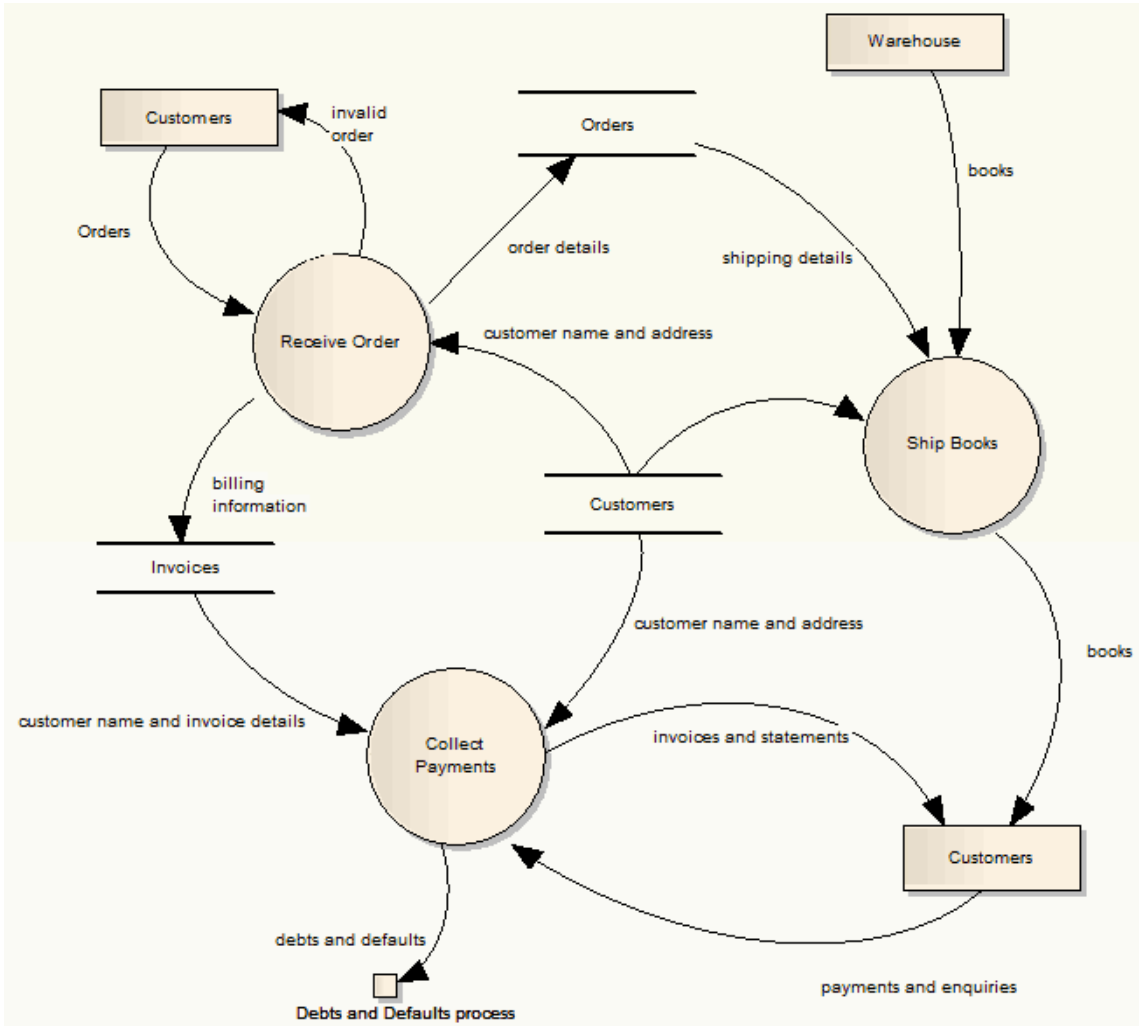
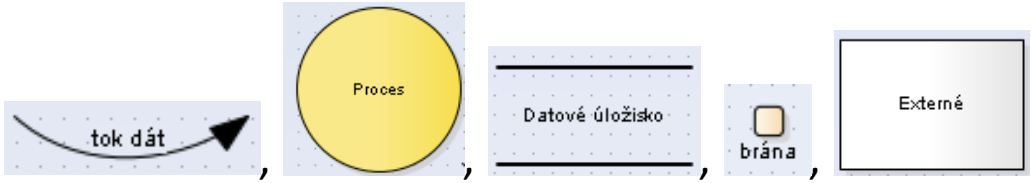


Diagram toku dát (data flow diagram)



- **Proces** = proces, alebo aktivita, v ktorej sú data používané alebo generované
- **Externé**= externý zdroj dát
- **Dátové úložisko** – interné fyzické úložisko údajov
- **Brána** – reprezentuje koncový bod vstupných/výstupných správ pri komunikácii s inými procesmi



Databázy a bezpečnosť - SLQ injection

Majme dva príklady na tvorbu statementu (príklad zo súboru MyDrbWrapper.java):

1	myStatement.executeUpdate("INSERT INTO MYLOG (TSTAMP, MSGTXT) VALUES ('"+myTs+"','called: myQuery')");
2	String mySqlString = "INSERT INTO MYLOG (TSTAMP, MSGTXT) VALUES (?, ?)" ; PreparedStatement ps = mConnection.prepareStatement(mySqlString); ps.setObject(1, myTs); ...

Z uvedených - druhý spôsob využitia je bezpečenjší – menej náchylný n a „SQL injection“ (bezpečnostná chyba založená na možnosti manipulovať s dátami v databáze bez nutnosti vlastníctva legitímnych prístupových údajov)

Príklad - vyrobme funkciu, kontrolujúcu meno+heslo:

```
public void checkLogin(String meno, String heslo) throws SQLException {
    Statement myStatement = mConnection.createStatement();
    ResultSet myResultSet = myStatement.executeQuery("SELECT * FROM uzivatel WHERE meno = '" +meno+"' AND heslo='"+heslo+"' ");
    printResultSet (myResultSet);
 }
```

A volajme ju:

```
    System.out.print("Zadaj meno: ");
    String meno = scanner.nextLine();
    System.out.print("Zadaj heslo: ");
    String heslo = scanner.nextLine();
    System.out.print("pracujem ...\n");
    myWrapper.checkLogin(meno, heslo);
```

V databáze máma tabuľku naplnenú nasledovne:

#	MENO	HESLO
1	jano	jeden
2	fero	dva
3	duro	tri

Normálne použitie:	Útok typu 1 (komentár)	Útok typu 2 (aj podmienka OR)
Zadaj meno: jano Zadaj heslo: jeden pracujem ... MENO: jano, HESLO: jeden	Zadaj meno: jano ' -- Zadaj heslo: netusim ake pracujem ... MENO: jano, HESLO: jeden	Zadaj meno: asdfadsf Zadaj heslo: 'or 1=1 --' pracujem ... MENO: jano, HESLO: jeden MENO: fero, HESLO: dva MENO: duro, HESLO: tri

Útok pomocou Union	Iné útoky
Zadaj meno: jano ' UNION SELECT ZIP, NAME FROM CUSTOMER -- Zadaj heslo: pracujem ... 1: 10095, 2: Old Media Productions 1: 10096, 2: Yankee Computer Repair Ltd 1: 12347, 2: Small Bill Company ...	Postupne sa dá prepracovať k údajom z komplet celej databázy ... Viac informácií napr. tu: https://resources.infosecinstitute.com/dumping-a-database-using-sql-injection/#gref

Ako sa pro SQL injection brániť?

➔ **Používať PreparedStatement, ktorý „escapuje“ znaky a kontroluje / validuje syntax.**

```
public void checkLoginSafe(String meno, String heslo) throws SQLException {
    Statement myStatement = mConnection.createStatement();
    String mySqlString = "SELECT * FROM uzivatel WHERE meno = ? AND heslo=? ";
    PreparedStatement ps = mConnection.prepareStatement( mySqlString );
    ps.setString( 1, meno);
    ps.setString( 2, heslo ) ;
    ResultSet myResultSet =ps.executeQuery();
    printResultSet (myResultSet);
}
```

Java – logovanie

- Je veľa knižníc na logovanie, log4j, ...
- My použijeme java.util.logging.Logger
- Použijeme napr. singleton (jedináčik) vzor:
  - private static final Logger LOGGER = Logger.getLogger("mojLogger");
  - ak to takto definujeme v každej triede, zdieľame ten istý logger.
    - Ak by sme chceli, môžeme v každej triede použiť vlastný Logger, stačí dať iné meno ...
- Pre daný logger stanovíme „handlery“
  - napr. file handler:
    - Handler fileHandler = new FileHandler("./DerbyDemo.log");
    - LOGGER.addHandler(fileHandler);
  - ...
- Potom nastavíme úroveň logovania:
  - LOGGER.setLevel(Level.ALL);
- Následne môžeme logovať, napr. pomocou:
  - LOGGER.log(Level.FINE, "demo logovaci text");
  - Logger sám dopĺňa do logu okrem správy aktuálny čas, triedu, metódu, ...
- Aké môžeme použiť úrovne logovania?
  - SEVERE (najvyššia), WARNING, INFO, CONFIG, FINE, FINER, FINEST (najnižšia)
- Správa sa loguje sa ak je úroveň správy >= úroveň logovania
- Príklad: DerbyDemoLogging (projekt)
- Ďalšie príklady:
  - https://examples.javacodegeeks.com/core-java/util/logging/java-util-logging-example/

Opakovanie z prvej prednášky - Typické fázy procesy vývoja pri použití vodopádovej metodológii

- Zber a analýza požiadaviek
- Návrh riešenia
- Vývoj/implementácia
- Testovanie
- Integrácia
- Nasadenie
- Údržba

Teraz lepšie popíšeme tieto fázy a dokumenty, ktoré pri nich vznikajú

Fáza 1: Zber a analýza požiadaviek (vytvorenie požiadaviek na riešenie)

Cieľ: vytvorenie požiadaviek na riešenie, neuvažuje sa nad konkrétnym spôsobom realizácie požiadaviek.

Aktivity

- Štúdia realizovateľnosti (vrátane časti Cost Business Analysis – CBA). Odpovedá na otázky:
  - Ako prispeje SW k obchodným cieľom organizácie ?
  - Je možné integrovať SQ k existujúcim riešeniam ?
  - Je potrebný nový HW, poprípade nový SW ?
  - Dá sa realizovať s danými zdrojmi a v danom čase?
- Zber a analýza požiadaviek
- Definícia a špecifikácia požiadaviek

Účastníci:

- Používateľ (zákazník) – pozor, často v podstate zákazník nevie čo chce a vyjde to najavo až počas procesu tvorby požiadaviek a systematického popisu potreby
- Analytik (môže byť od zákazníka, alebo zavolaný na vykonania analýzy, … atď)

Metódy procesu

- Rozhovor so zákazníkom, prípadne dotazníky
- Monitorovanie pracovných postupov zákazníka, Priama účasť na prácach zákazníka
- Analýza existujúceho softvéru, analýza a štúdium dokumentov a štandardov
- Výsledky procesu je treba aj priebežne písomne zaznamenávať

Fáza 1: Zber a analýza požiadaviek (vytvorenie požiadaviek na riešenie) 2

Typy požiadaviek:

1. **Funkčné** - Určujú chovanie, funkcie softvéru – čo to má robiť (čo má byť výsledkom) a nie ako to má dosiahnuť.
2. **Nefunkčné** - Špecifikujú vlastnosti a obmedzujúce podmienky daného softvéru
   a. Požiadavky na prevádzku systému (počet používateľov, ...)
   b. Požiadavky na softvér (efektívnosť, bezpečnosť, HW prostredie, OS, iný spolupracujúci SW)
   c. Externé požiadavky (legislatívne, etické požiadavky)

Čo stanovujú požiadavky?

Neformálna rovina	**Formálna rovina IEEE 830-1998**
• Na čo bude SW slúžiť • Aké informácie sa budú uchovávať • Kto ho bude používať • Aké funkcie, operácie bude zabezpečovať • Aké sú vstupy a výstupy • Aké sú interakcie s inými systémami.	"Špecifikácia požiadaviek" • Prirodzený jazyk • Formuláre • Prípady použitia • Pseudokódy, scenáre, ...

Vysledok tvorby požiadaviek - forma

- Spravidla Špecifikácia požiadaviek je v dokumente „**výzva na podávanie projektov**“ (RFP request for proposals)
  - Pozor, nepomýliť si s RFI – request for information . toto je nezáväzné vyžiadanie stanoviska /názoru od perspektívnych uchádzačov v priebehu fázy analýzy a tvorby požiadaviek
  - Spôsob výberu riešiteľov
    - Priame oslovenie
    - Výberové konanie
      - Formálne
      - V zmysle zákona (verejné obstarávanie/elektronické aukcie)
- Iné formy, napr. na základe zmluvy, obsahujú „popis požadovaného technického riešenia“ ...

Následné kroky

Zadávateľ	Realizátor
1) Vytvorenie ponuky ktoré obsahuje návrh riešenia v požadovanom rozsahu, navrhuje spôsob plnenia požiadaviek.	
	2) Vyhodnotenie ponúk a vybratie výhercu
3) Spísanie zmluvy	
4) Poskytnutie súčinnosti	5) Podrobný návrh riešenia

...

Forma požiadaviek a reakcií na ne

- Zvyčajne má každá požiadavka identifikátor a textový popis a požiadavky sú v tabuľkách
- Zvyčajne je v ponuke napísané ako presne bola požiadavka pochopená a akým spôsobom sa splní
- Už teraz je dôležité myslieť na interpretáciu, aby neboli nezhody pri akceptácii riešenia (rôzny výklad, nekompletná realizácia atď ...)
- Male projekty majú niekoľko desiatok požiadaviek, veľké (a premyslené pojekty) aj tisic a viac ...

Študenti

- Ako ste dostali sformulované požiadavky na riešenie vášho zadania?
- Boli ste spokojní s detailom zadania?
- Bolo príliš obmedzujúce, či naopak príliš voľné?
- Spĺňalo byt toto zadanie aj miniatúrne aj gigantické riešenie?
- T.j. bolo zadanie dobré a dostatočné?
    - Nedostatočne premyslené zadanie môže byť pre zadávateľa OBROVSKÝ PROBLÉM! (čo vlastne dostane, bude s tým spokojny?). Často sa to podceňuje!

Fáza 2: Návrh riešenia

Architektonický návrh

- Celková koncepcia architektúry SW
- Definícia funkcionality softvéru
- Plán nasadenia SW

Podrobný návrh

- Návrh komponentov systému
- Špecifikácia ošetrenia chybových stavov

Fáza 3: Implementácia

- Kódovanie komponentov
- Dokumentácie komponentov
- Testovanie komponentov

Fáza 4: Integrácia a testovanie

- Integrovanie komponentov do modulov
- Testovanie modulov a celého systému

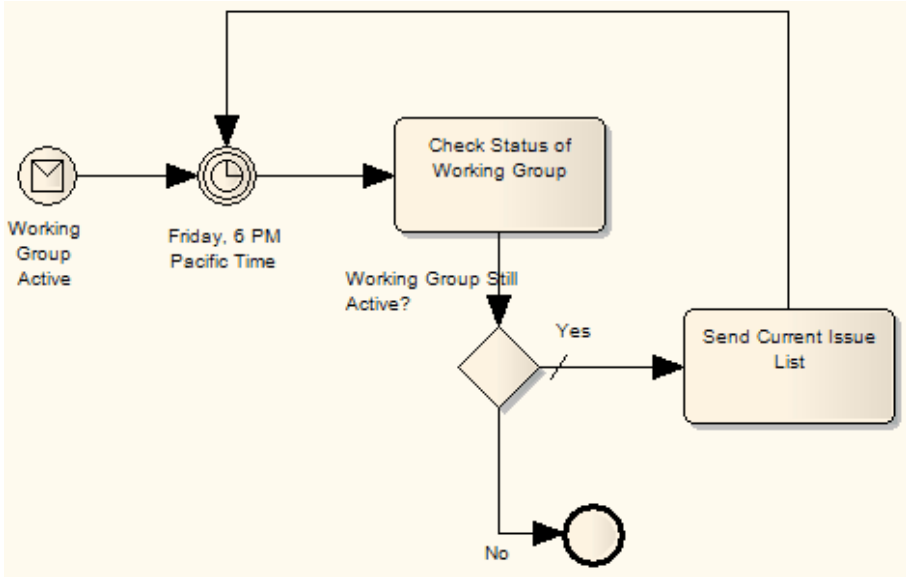
Fáza 5: Akceptačné testovanie a inštalácia

- Testovanie používateľom
- Školenie používateľom
- Nasadenie SW
- Dodanie dokumentácie
  - veľmi variabilné, rozsah záleží od rozsahu projektu a dohody
  - spravidla minimálne
    - užívateľská príručka (čo systém robí z pohľadu užívateľa - funkcionalita a obrazovky)
    - popis technického riešenia (architektúra systému)
    - administračná príručka (nasadenie a administrácia/prevádzkovanie)

Biznis procesy – BMPN & BPEL

- BPMN=Business process Modeling notation
- BPEL= Biznis process execution language, je to skratka názvu Web Service BPEL (WS-BPEL)
  - jazyk na orchestráciu, ktorý je serializovaný v XML
  - používa WSDL (Web Service Descritpion Language)

BPMN príklad



BPEL príklad

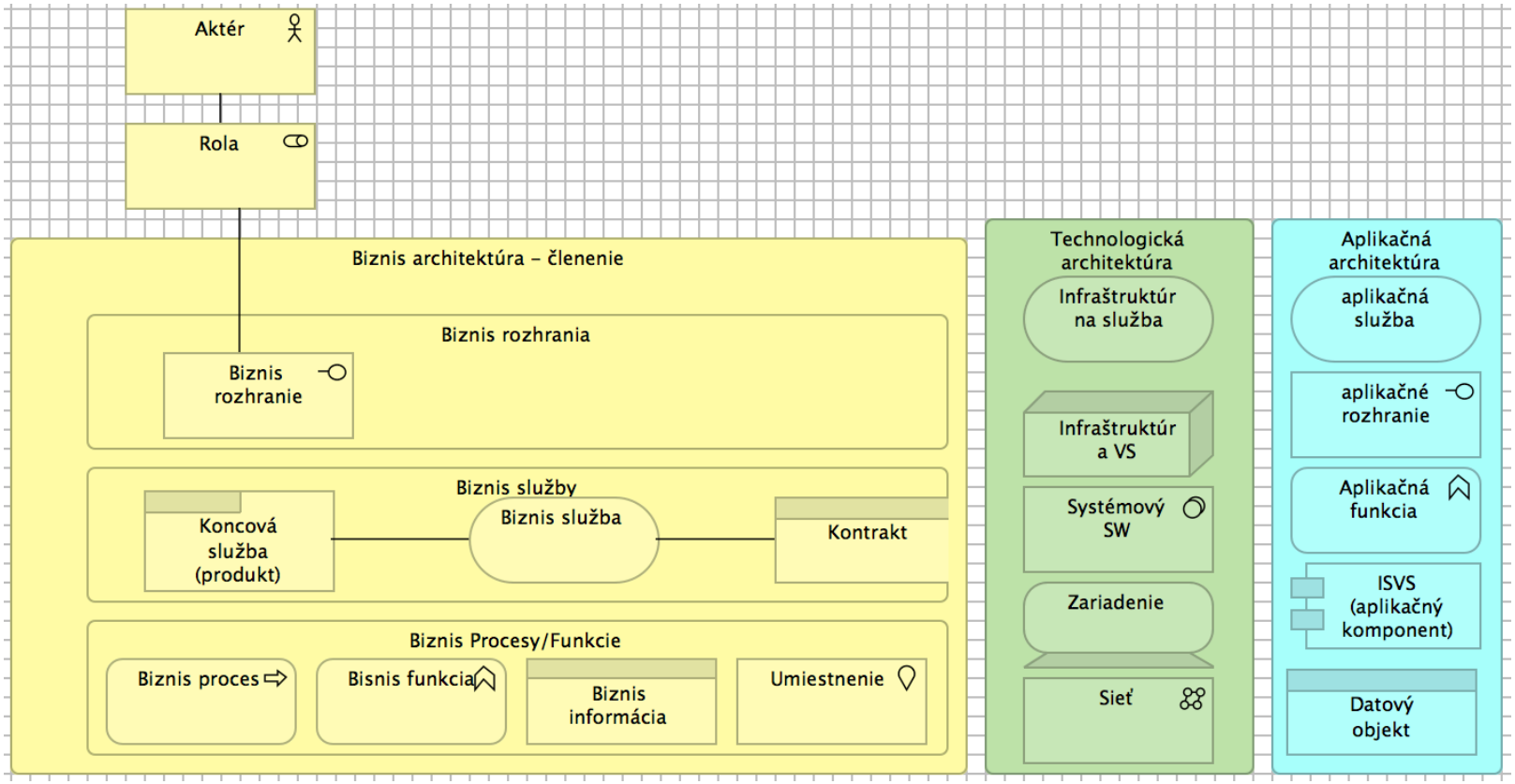
```
<?xml version="1.0"?>
<bpel:process name="SampleBPELProcess"
              abstractProcess="no"
              queryLanguage="XPath 1.0"
              suppressJoinFailure="no"
              enableInstanceCompensation="no"
              targetNamespace="www.samplebpelprocess.com"
              xmlns:bpel="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
              xmlns:sb="www.samplebpelprocess.com"
              xmlns:bpw="www.samplebpelprocess.com/wsdl">

    <bpel:partnerLinks>
        <bpel:partnerLink name="MsgDisplayer" partnerLinkType="sb:MsgDisplayer_LT" myRole="MsgDisplayer"/
    </bpel:partnerLinks>

    <bpel:variables>
        <bpel:variable name="request" messageType="sb:request"/>
        <bpel:variable name="response" messageType="sb:response"/>
    </bpel:variables>

    <bpel:sequence>
        <bpel:receive name="StartEvent1" createInstance="yes" variable="request"
                      partnerLink="MsgDisplayer" portType="sb:DisplayMessagePT" operation="DisplayMessage
        <bpel:sequence>
            <bpel:assign>
                <bpel:copy>
                    <bpel:from>Hello World</bpel:from>
                    <bpel:to variable="response" part="message"/>
                </bpel:copy>
            </bpel:assign>
            <bpel:empty name="Activity1"/>
        </bpel:sequence>
        <bpel:reply name="EndEvent1" variable="response" partnerLink="MsgDisplayer" portType="sb:DisplayM
                    operation="DisplayMessageOpr"/>
    </bpel:sequence>
</bpel:process>
```

Biznis architektúra - Archimate diagramy

- Cieľom Archimate štandardu je poskytnúť grafický jazyk na reprezentáciu enterprise architektúry v čase vrátane plánovania zmien, migrácií a motivácií a dôvodov k nim. Archimate štandard je tesne zviazaný s TOGAF štandardom, napr. neposkytuje vlastné definície pojmov, ale používa TOGAF definície.
- Snahou TOGAFu (The Opeg Group Architecture Framework) je na základe komunitných štandardov popísať metódy a nástroje používané pri tvorbe architektúry.
- Základný referenčný rámec Verejnej správy (VS) SR vychádza z TOGAF a ArchiMate architektonických rámcov. Prispôsobený metamodel je kompatibilnou podmnožinou ArchiMate meta modelu a bude sa postupne, s rastúcou architektonickou zrelosťou VS, rozvíjať do úplného ArchiMate meta modelu.
- Jednotlivé vrstvy a aspekty s príslušnými konceptami sú definované nasledovne:
  - motivácia,
  - biznis architektúra,
  - architektúra informačných systémov (aplikácie a dáta),
  - technologická architektúra,
  - bezpečnostná architektúra
  - implementácia a migrácia.

Archimate – príklad

Príklady, napr v AV_EAVSSR_2014-2020_Architektonicke_ramce_VS.pdf [www.informatizacia.sk]. Množina prvkov je takáto:



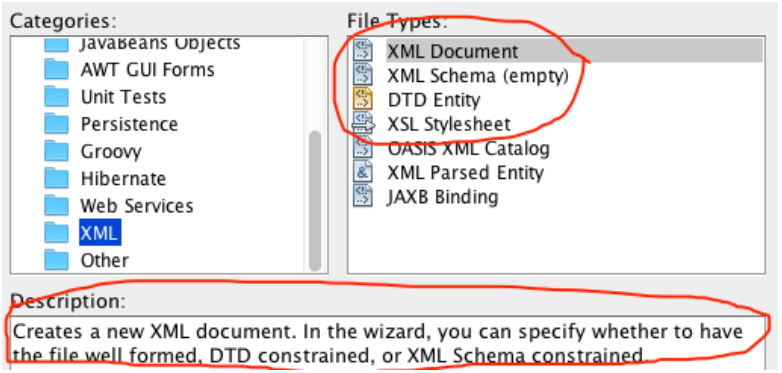
Objekty v menu SW produktu “Archi”

Tvorba SW a CASE nástroje

- CASE= computer aided software engineering
- Pojem vznikol cca v r. 1970
- Pokrýva celý vývojový cyklus SW, vrátane
  - nástroje na návrh
  - generovanie kódu
  - automatické generovanie dokumentácie
  - prototypovanie
  - repozitáre
  - ďalšie pomocné nástroje na zvýšenie produktivity
- na tomto predmete je váš CASE nástroj Netbeans ...

XML

- eXtensible markup language (značkovací jazyk)
- nemá preddefinované značky (nagy, názvy elementov)
- umožnuje opisovať (značkovať) ľubovoľné údaje
- pomocou XSL (eXtensible stylesheet language) transformácie (XSLT) vieme transformovat XML
  - na iné XML
  - na iný formát, napr. do HTML
- XML dokumenty spĺňajú XML špecifikáciu a majú správnu štruktúru, sa nazývajú **well formed (správne štrukturované**
- XML dokumenty, ktoré naviac spĺňajú požiadavky schém definované v jazykoch DTD, alebo XSD sa nazývajú **validné**
  - DTD (document type definition) –schéma , resp. gramatika, ktorá definuje štruktúru dokumentu a platné elementy a atribúty
  - XSD (Xml Schema Definition) – samotná schéma je napísana v jazyku XML
    - Príklad napr. na http://www.w3.org/TR/xmlschema-0/
- XML je textový formát, ako kódovať binárne údaje
  - Pomocou base64 (RFC 4648)
    - `<stuff dt:dt="binary.base64">84592gv8Z53815Zb82bA68g</stuff>`
  - XOP (XML-binary Optimized Packaging) – mechanizmus, ktorý umnožňuje obchádzať base64 enkódovanie binárnych informácií, pomocou multipart MIME ...
    - http://www.w3.org/TR/xop10/
- Netbeans - ako riadny CASE nástroj, obsahuje xml editor →

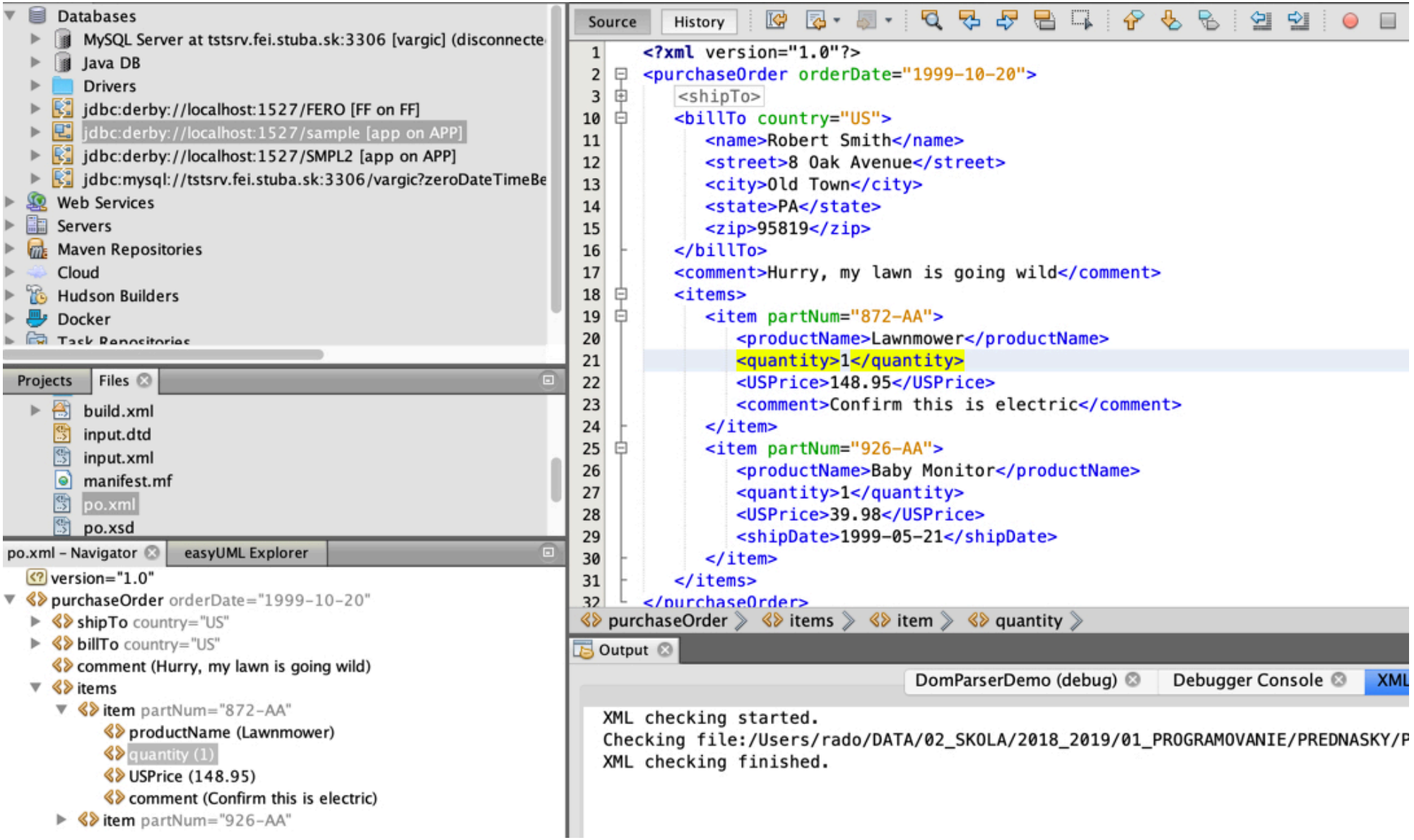


Príklad XML/XSD z
http://www.w3.org/TR/xmlschema-0/

```xml
<?xml version="1.0"?>
<purchaseOrder orderDate="1999-10-20">
   <shipTo country="US">
      <name>Alice Smith</name>
      <street>123 Maple Street</street>
      <city>Mill Valley</city>
      <state>CA</state>
      <zip>90952</zip>
   </shipTo>
   <billTo country="US">
      <name>Robert Smith</name>
      <street>8 Oak Avenue</street>
      <city>Old Town</city>
      <state>PA</state>
      <zip>95819</zip>
   </billTo>
   <comment>Hurry, my lawn is going wild<!/comment>
   <items>
      <item partNum="872-AA">
         <productName>Lawnmower</productName>
         <quantity>1</quantity>
         <USPrice>148.95</USPrice>
         <comment>Confirm this is electric</comment>
      </item>
      <item partNum="926-AA">
         <productName>Baby Monitor</productName>
         <quantity>1</quantity>
         <USPrice>39.98</USPrice>
         <shipDate>1999-05-21</shipDate>
      </item>
   </items>
</purchaseOrder>
```

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:annotation>
    <xsd:documentation xml:lang="en">
     Purchase order schema for Example.com.
     Copyright 2000 Example.com. All rights reserved.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="purchaseOrder" type="PurchaseOrderType"/>

  <xsd:element name="comment" type="xsd:string"/>

  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="shipTo" type="USAddress"/>
      <xsd:element name="billTo" type="USAddress"/>
      <xsd:element ref="comment" minOccurs="0"/>
      <xsd:element name="items"  type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
  </xsd:complexType>

  <xsd:complexType name="USAddress">
    <xsd:sequence>
      <xsd:element name="name"   type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
      <xsd:element name="city"   type="xsd:string"/>
      <xsd:element name="state"  type="xsd:string"/>
      <xsd:element name="zip"    type="xsd:decimal"/>
    </xsd:sequence>
    <xsd:attribute name="country" type="xsd:NMTOKEN"
                   fixed="US"/>
  </xsd:complexType>

  <xsd:complexType name="Items">
    <xsd:sequence>
      <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="productName" type="xsd:string"/>
            <xsd:element name="quantity">
              <xsd:simpleType>
                <xsd:restriction base="xsd:positiveInteger">
                  <xsd:maxExclusive value="100"/>
                </xsd:restriction>
              </xsd:simpleType>
            </xsd:element>
            <xsd:element name="USPrice"  type="xsd:decimal"/>
            <xsd:element ref="comment"   minOccurs="0"/>
            <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
          </xsd:sequence>
          <xsd:attribute name="partNum" type="SKU" use="required"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <!-- Stock Keeping Unit, a code for identifying products -->
  <xsd:simpleType name="SKU">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{3}-[A-Z]{2}"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

Práca s XML v prostredí Netbeans



NETBEANS umožnuje pri XML kontrolovať aj

- či je dobré po syntaktickej stránke (wellformed)
- či je validné po obsahovej stránke na základe schéme (DTD, XSD)

Základná práca s XML v Jave

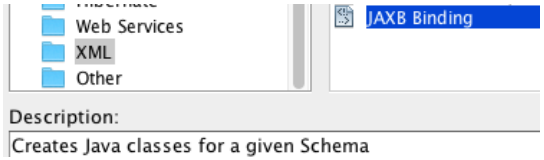
- Príklad: DomParserDemo (projekt)
  - Príklad založený na : https://www.tutorialspoint.com/java_xml/java_dom_parse_document.htm
- Použijeme DOM (Document Object Model) - objektový model dokumentu
  - DOM je objektovo orientovaná reprezentácia dokumentu XML alebo HTML, kde dokument môže byť ďalej spracovávaný a výsledok tohto spracovania prezentovaný.
  - DOM konzorcia W3C je štandardné API rozhranie pre štruktúru dokumentov.
  - Vďaka DOM je možné pristupovať k objektom, mazať, pridávať alebo editovať ich vrátane obsahu, atribútov a vzhľadu.
- Vytvorenie DOM:

```
File inputFile = new File("input.xml");
DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(inputFile);
```

- Zistenie root elementu:

  System.out.println("Root element :" + doc.getDocumentElement().getNodeName());
- Vytvorenie zoznamu elementov, ktoré sú typu „študent“
  - *NodeList nList = doc.getElementsByTagName("student");*
- Pridávanie nových uzlov, atribútov, ...
  - nNode.appendChild(...)
  - eElement.setAttribute(name, value);

WSDL

- WSDL = Web service definition language
- Spravidla opisuje SOAP komunikáciu
  - SOAP = Simple Object Access Protocol, protokol na výmenu správ založený na XML
  - nástupca XML-RPC
- Je zapisaný vo formáte XML
  - Špecifikácia je prístupná na http://www.w3.org/TR/wsdl20/
- Napr:
  - https://www2.uvo.gov.sk/sluzby1
    - Zoznam podnikateľov: http://www2.uvo.gov.sk/uvoservices/RegPod?wsdl
    - Vestník verejného obstarávania: http://www2.uvo.gov.sk:80/uvoservices/Vestnik?wsdl
  - http://nipi.sazp.sk/arcgis/services/env_zataze/environmentalna_zataz/MapServer?wsdl
    - Služby sú na ďalšom slide (vizualizácia po naimportovaní WSDL do EA)
- Netbeans view pracovat s WSDL
  - Pomocou JAXB (Java Architecture for XML Binding API) vie napr. generovat komplet triedy na základe zadania URL s WSDL
    - 

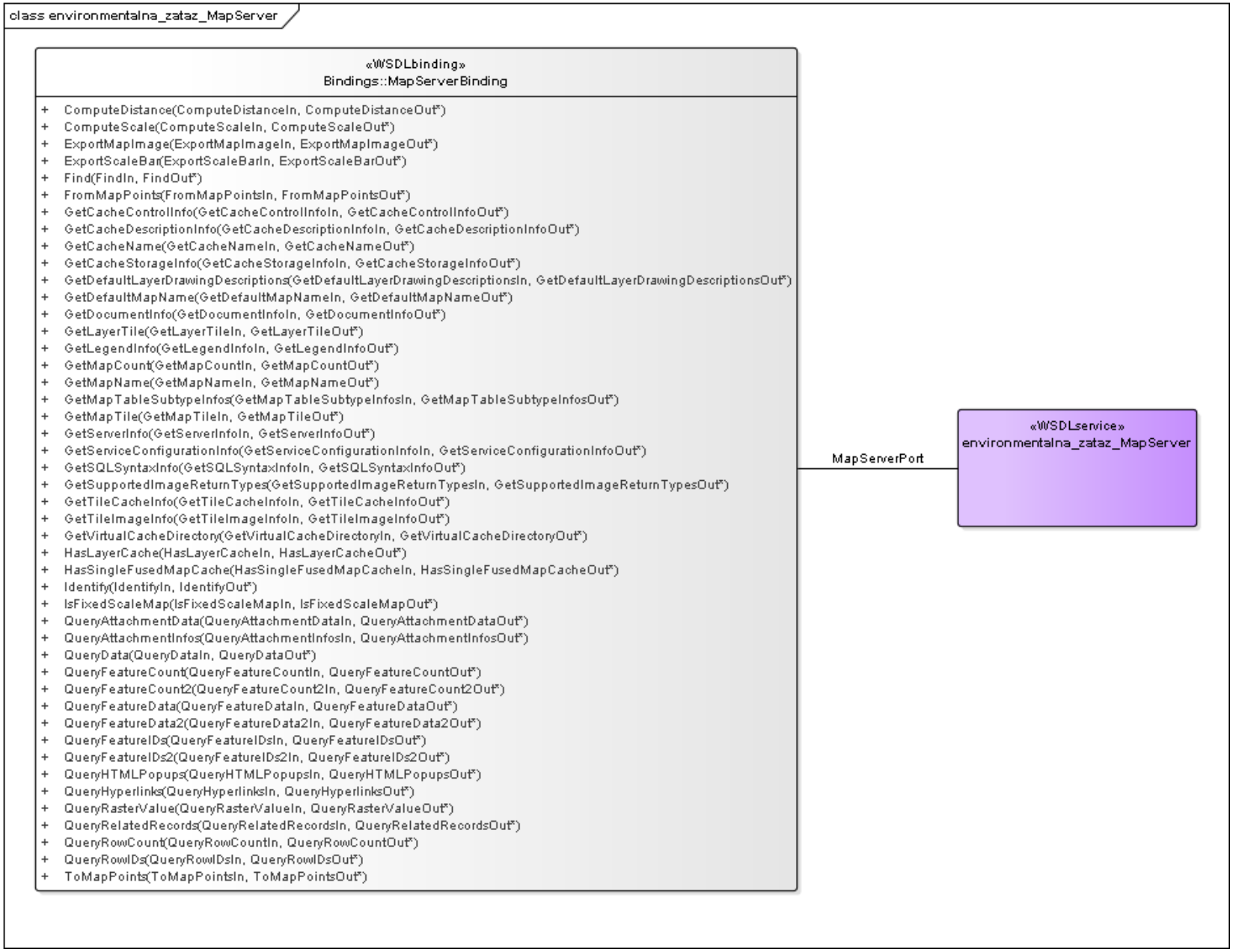
    - viac napr. tu https://netbeans.org/kb/74/websvc/jaxb.html

Príklad WSDL (http://www2.uvo.gov.sk/uvoservices/RegPod?wsdl)

```
<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
 xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
 xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:tns="http://external.services.portal.uvo.gov.sk/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://external.services.portal.uvo.gov.sk/" name="RegPodService">
  <wsp:Policy wsu:Id="RegPodPortBinding_MTOM_Policy">
    <ns1:OptimizedMimeSerialization xmlns:ns1="http://schemas.xmlsoap.org/ws/2004/09/policy/optimizedmimeserialization" wsp:Optional="true"/>
  </wsp:Policy>
  <types>
    <xsd:schema>
      <xsd:import namespace="http://external.services.portal.uvo.gov.sk/" schemaLocation="http://www2.uvo.gov.sk:80/uvoservices/RegPod?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="SearchPod">
    <part name="parameters" element="tns:SearchPodRequest"/>
  </message>
  <message name="SearchPodResponse">
    <part name="parameters" element="tns:SearchPodResponse"/>
  </message>
  <message name="Exception">
    <part name="fault" element="tns:Exception"/>
  </message>
  <message name="GetPod">
    <part name="parameters" element="tns:GetPodRequest"/>
  </message>
  <message name="GetPodResponse">
    <part name="parameters" element="tns:GetPodResponse"/>
  </message>
  <portType name="RegPodWS">
    <operation name="SearchPod">
      <input wsam:Action="SearchPod" message="tns:SearchPod"/>
      <output wsam:Action="http://external.services.portal.uvo.gov.sk/RegPodWS/SearchPodResponse" message="tns:SearchPodResponse"/>
      <fault message="tns:Exception" name="Exception" wsam:Action="http://external.services.portal.uvo.gov.sk/RegPodWS/SearchPod/Fault/Exception"/>
    </operation>
    <operation name="GetPod">
      <input wsam:Action="GetPod" message="tns:GetPod"/>
      <output wsam:Action="http://external.services.portal.uvo.gov.sk/RegPodWS/GetPodResponse" message="tns:GetPodResponse"/>
      <fault message="tns:Exception" name="Exception" wsam:Action="http://external.services.portal.uvo.gov.sk/RegPodWS/GetPod/Fault/Exception"/>
    </operation>
  </portType>
  <binding name="RegPodPortBinding" type="tns:RegPodWS">
    <wsp:PolicyReference URI="#RegPodPortBinding_MTOM_Policy"/>
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="SearchPod">
      <soap12:operation soapAction="SearchPod"/>
      <input>
        <soap12:body use="literal"/>
      </input>
      <output>
        <soap12:body use="literal"/>
      </output>
      <fault name="Exception">
        <soap12:fault name="Exception" use="literal"/>
      </fault>
    </operation>
    <operation name="GetPod">
      <soap12:operation soapAction="GetPod"/>
      <input>
        <soap12:body use="literal"/>
      </input>
      <output>
        <soap12:body use="literal"/>
      </output>
      <fault name="Exception">
        <soap12:fault name="Exception" use="literal"/>
      </fault>
    </operation>
  </binding>
  <service name="RegPodService">
    <port name="RegPodPort" binding="tns:RegPodPortBinding">
      <soap12:address location="http://www2.uvo.gov.sk:80/uvoservices/RegPod"/>
    </port>
  </service>
</definitions>
```

http://nipi.sazp.sk/arcgis/services/env_zataze/environmentalna_zataz/MapServer?wsdl

- Služby - vizualizácia po naimportovaní WSDL do EA



Iné zaujímavé a dôležité témy, ktoré sme neprebrali podrobnejšie

- REST a SOAP verzus REST
  - CRUD operácie
  - idempodencia
- JSON
- Java script
- AJAX a AJAJ
- HTML, CSS
- Servlety
- Enterprise systémy
  - ich vysoká dostupnosť
  - intergrácie
- ERP systémy
- DWH a podrobnejšie
- Bezpečnostné aspekty
- ....
- a veľa ďalšieho ...