SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 61

INTERAKTIVNI GRAFIČKI PREGLEDNIK 3D OBJEKATA TEMELJEN NA TEHNOLOGIJAMA WEB-A

Dario Filipović

Zagreb, lipanj 2010.

Sadržaj

Uvod....1

1 Korištene tehnologije....2

1.1 Xith3D....2

1.1.1 Karakteristike i svojstva....3

1.1.2 Graf scene....4

1.1.3 Xith3D i Java3D....6

1.2 Web usluge....7

1.2.1 SOAP (Simple Object Access Protocol)....8

1.2.2 WSDL (Web Services Description Language)....13

1.2.3 Axis2....18

2 Interaktivni grafički preglednik 3D objekata temeljen na tehnologijama weba....20

2.1 Idejno rješenje....21

2.2 Preglednik 3D modela....22

2.2.1 Obrasci upotrebe preglednika 3D modela....22

2.2.2 Opis rješenja preglednika 3D modela proizvoda....25

2.3 Komunikacija i poslužitelj modela....29

2.3.1 Uputstvo za korištenje preglednika proizvoda....34

2.3.2 Izrada modela proizvoda za preglednik....36

2.3.3 Demonstracijski primjer....38

3 Zaključak....39

Sažetak....40

Abstract....41

Literatura....42

Uvod

Komunikacijske tehnologije su danas jedan od najvažnijih aspekata uspješnog razvoja društva. Godine konstantnog razvoja komunikacijskih tehnologija rezultirale su stvaranjem fleksibilne platforme pogodne za razvoj široke palete primjenjivih aplikacija.

Na webu je moguće je pronaći iznimno velik broj takvih aplikacija koje korisnicima pomažu pri rješavanju raznoraznih konkretnih problema. Jedna grana tih aplikacija bile bi aplikacije za trodimenzionalnu vizualizaciju. Po mišljenju autora ta je grana neopravdano zakinuta i samo je pitanje vremena kada će trodimenzionalna vizualizacija na webu u potpunosti zaživjeti. U prilog tome ide i činjenica da gotovo svako stolno računalo sadrži grafičku karticu koja je sposobna prikazivati kompleksne trodimenzionalne objekte. Dakle prosječan korisnik Interneta bio bi u stanju koristiti uslugu za trodimenzionalnu vizualizaciju.

Interaktivni grafički preglednik 3D objekata je primjer jedne takve aplikacije. On bi omogućio korisniku razgledanje 3D objekta uz određenu razinu interakcije korisnika i 3D objekta. Konkretniji primjer bio bi preglednik 3D modela proizvoda gdje bi se proizvođačima ponudila demonstracija njihovih proizvoda, a korisnicima uvid u karakteristike i detaljni izgled proizvoda uz mogućnost uvida u ograničen skup svojstava proizvoda vezanih uz izgled.

U ovom će se radu razmatrati upravo mogućnost interaktivnog pregleda 3D objekata temeljenog na tehnologijama web-a i to na konkretnom primjeru preglednika proizvoda. Iako već postoje neke standardne tehnologije za trodimenzionalni prikaz objekata kao što je VRML one su prilično generičke i nisu baš pogodne za ovakvu primjenu. Namjera rada je pokazati da je moguće povezati komunikacijske tehnologije i grafičke sustave računala u relativno jednostavan sustav koji omogućuje pregled 3D objekata.

1 Korištene tehnologije

Prije početka izrade programskog proizvoda bilo je potrebno odlučiti se koje će se tehnologije koristiti. Prvenstveno je postojala želja da to bude programsko rješenje koje je što manje ovisno o operacijskim sustavima. Programski jezik *Java* je odlična platforma na kojoj se takvo rješenje može ostvariti. Osim toga Javu je moguće integrirati u web preglednike putem *Java appleta*.

Nadalje, trebalo se odlučiti na koji će se način obavljati iscrtavanje trodimenzionalnih objekata (trodimenzionalnih modela). U tom području postoji nekoliko programskih potpora koje uglavnom pružaju iste mogućnosti, a razlikuju se uglavnom po robusnosti i podršci zajednice. Najveća je dilema postojala između programskih potpora *Xith3D* i *Java3D*.

Posljednji glavni element programskog proizvoda je komunikacija putem mreže. Programiranje klijenata i poslužitelja na temelju *socketa* nije dobro rješenje. Danas se veći naglasak stavlja na interakciju između sustava i jednostavnost korištenja izvora na Internetu. Tako je logičan izbor neka od tehnologija na aplikacijskoj razini koja programerima i korisnicima sakriva nebitne detalje TCP/IP komunikacije.
Web usluge temeljene na SOAP-u se koriste već duže vremena i nakon analize specifičnosti SOAP-a odlučeno je da će se komunikacija odvijati prema SOAP specifikaciji.

1.1 Xith3D

Xith3D je programska potpora koja olakšava izradu i korištenje grafa scene. Karakterizira ju moderan dizajn, velik broj mogućnosti i vrlo dobre performanse pri iscrtavanju. Cilj *Xith3D* projekta je razviti skup alata za stvaranje interaktivnih 3D okruženja korištenjem standardne sklopovske opreme kao što je oprema stolnih računala. Pod time se podrazumijeva izrada igara, vizualizacija podataka, stvaranje prototipova 3D okruženja i slično. Programska potpora *Xith3D* je napisana u programskom jeziku *Java* i kao osnovni sustav za iscrtavanje geometrije koristi sučelje *OpenGL*. Izravna posljedica tih karakteristika je naslijeđena prenosivost i izvršavanje na svim platformama koje imaju

podršku za *Javu* i *OpenGL*. *Xith3D* je tako uspješno testiran na *Linux*, *Windows*, *MacOSX* i mnogim drugim *Unix*-baziranim operacijskim sustavima. [1]

1.1.1 Karakteristike i svojstva

Xith3D je vrlo bogata biblioteka koja upravljanje trodimenzionalnim objektima dovodi na visoku razinu pritom sakrivajući kompleksne mehanizme iscrtavanja i upravljanja ulaznim jedinicama. Krajnje sučelje je vrlo jednostavno i fleksibilno, te omogućuje stvaranja kompleksnih sustava na vrlo jednostavan način.

OpenGL se iz *Xith3D* okruženja koristi pomoću biblioteke niske razine JOGL (*Java OpenGL*) koja je implementirana kao omotač *OpenGL* biblioteke. Velika većina funkcija *OpenGL*-a raspoloživa u toj biblioteci u programskom jeziku *C* moguće je koristiti i u *Javi* kroz biblioteku *JOGL*. Kako je JOGL samo omotač *OpenGL*-a, funkcije *OpenGL*-a se u biblioteci JOGL koriste slično kao i u programskom jeziku *C*, dakle pozivom izloženih funkcija niske razine. JOGL tako nema nikakvu objektno orijentiranu arhitekturu. Iz *OpenGL*-a je naslijeđena mogućnost iscrtavanja geometrije u nekom od slijedećih oblika: polje trokuta, polje trake trokuta, indeksirano polje trokuta, indeksirano polje trake trokuta, polje ventilatora trokuta, polja vrhova, liste prikazivanja i spremnika vrhova. Najkorisnija je mogućnost korištenja grafa scene kao osnovne strukture podataka za oblikovanje scene unutar *Xith3D* okruženja.

Xith3D može kroz JOGL koristiti AWT, *Swing*, SWT i *QT-jambi* biblioteke za upravljanje prozorima. Tako je osim osnovne mogućnosti izravnog stvaranja prozora s Xith3D kontekstom moguće kontekst integrirati u neku složeniju aplikaciju koja koristi prozore.

Ostala važnija svojstva[1]:

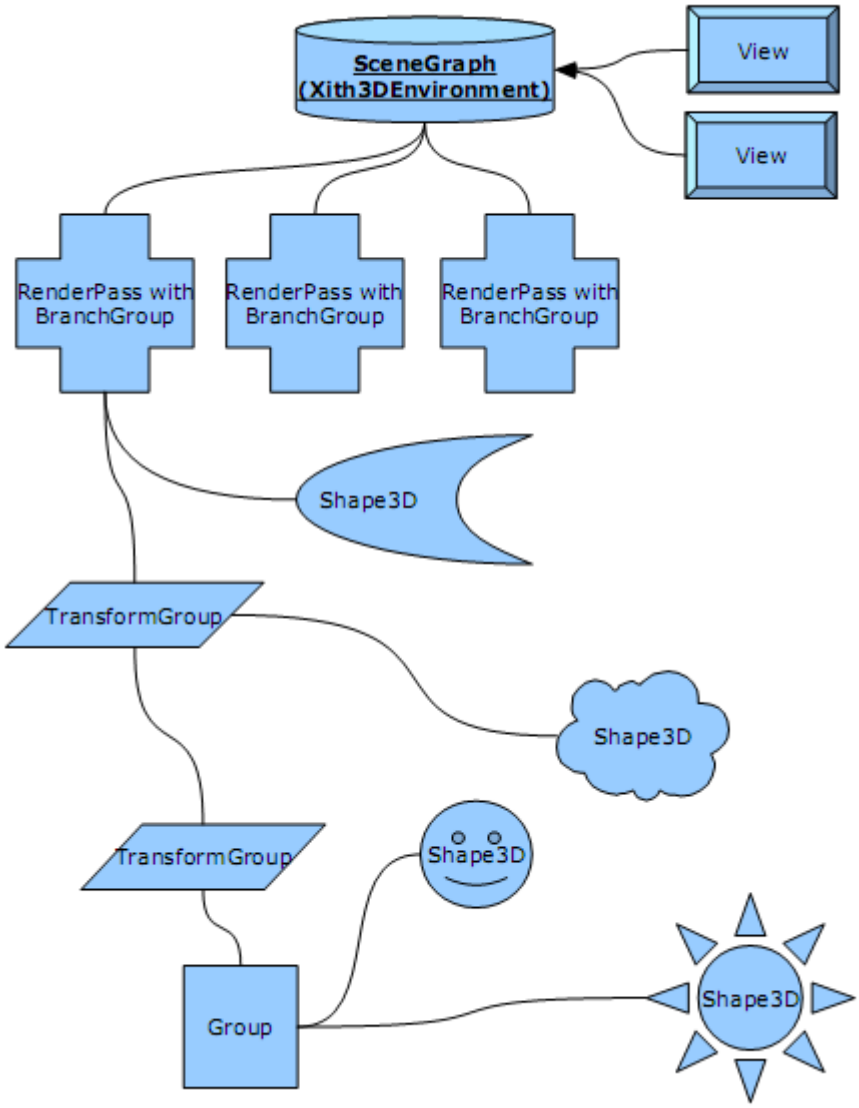
- učitavanje modela i scena - *Xith3D* sadrži klase za učitavanje modela i scena. Trenutno su podržani formati: ASE, AC3D, OBJ, 3DS, MD2, MD3, MD5, Cal3D, COLLADA 1.4 i *Quake3/Half-Life* BSP.
- učitavanje i manipulacija teksturama - različite metode generiranja koordinata tekstura, velik broj podržanih slikovnih formata; PNG, JPG, GIF, BMP, PCX, TGA, DDS, SGI
- čestični sustavi, volumeni sjena, mapiranje sjena, interaktivni odabir elemenata grafa scene
- zvuk - ambijentalni i prostorni zvukovi zasnovani na OpenAL i JavaSound bibliotekama
- prikazivanje terena - učitavanje visinskih mapa i prikazivanje terena uz sjenčanje
- ugrađeno grafičko sučelje - specijalizirano za izradu sučelja u igricama, izvorno je napisano kao dio programske potpore *Xith3D* i izvršava se u *Xith3D* kontekstu
- upravljanje projekcijskim volumenom

1.1.2 Graf scene

Graf scene je osnovni način iscrtavanja u *Xith3D* i najvažniji koncept oko kojeg je izgrađena biblioteka. Stvarno iscrtavanje se izvršava kroz *OpenGL*, a osnovna poveznica između *OpenGL*-a i grafa scene su transformacijske matrice. One su dio *OpenGL*-a i služe za definiranje transformacija primitivnih geometrijskih oblika kao što su, najčešće, trokuti. Transformacije mogu biti translacija, rotacija ili skaliranje, a moguće ih je i akumulirati. Graf scene u *Xith3D* ima oblik podatkovne strukture strogog stabla, što nije nužno slučaj kod drugih izvedbi grafa scene. Stablo se sastoji od čvorova. Čvorovi mogu biti tipa apstrakne grupe čvorova (`TransformNode`) ili listova (`Node`). Grupa čvorova može sadržavati više drugih čvorova bilo kojeg tipa. Svaka grupa čvorova sadrži transformacijske matrice za svaki čvor dijete. Prolaskom kroz stablo od korijena do čvora te se transformacijske matrice akumuliraju pri iscrtavanju. Akumuliranje transformacijskih

matrica ima za posljedicu da transformacije na čvoru koji je bliži korijenu grafa scene utječu na transformaciju čvorova koji su dalje u odnosu na korijen grafa scene uz uvjet da je bliži čvor predak daljnjem čvoru. Listovi grafa scene su najčešće objekti konkretnog tipa *Shape3D* koji sadrži geometriju.

Korijen aplikacije bilo kojeg *Xith3D* okruženja je instanca klase `SceneGraph`. Stvoreni korijen aplikacije je potrebno vezati na instancu klase `Renderer` koja obavlja iscrtavanje. Početni korijenski čvor je tipa `BranchGroup` koja je vezana uz jednu ili više instanci klase `RendererPass`. Klasa tipa `View` pritom definira projekcijski volumen. Primjer jednog grafa scene je na slici 1.



Slika 1: Primjer grafa scene u Xith3D okruženju[3]

Graf scene u *Xith3D* je strogo stablo što može biti problem u slučaju da se neki geometrijski čvor (najčešće tipa *Shape3D*) želi dodijeliti više grupi čvorova. U tom slučaju je moguće koristiti metodu za stvaranje dijeljene kopije geometrije `sharedCopy()`.
Prednost organiziranja scene u stablastu strukturu je i mogućnost optimizacije grafa scene prije same operacije iscrtavanja. Jednostavan primjer takve optimizacije koja se u grafovima scene gotovo uvijek koristi je izrezivanje po projekcijskom volumenu. Ono se zasniva na rekurzivnom prolasku po grafu scene od korijena i ispitivanja presijecanja obujmica 3D objekata s projekcijskim volumenom. Odbacivanjem čvorova koji za krajnje generirani okvir (engl. *frame*) nisu bitni smanjujemo opterećenje grafičkog procesora.

1.1.3 *Xith3D* i *Java3D*

Java3D je programska potpora čija je osnovna namjena ponuditi programerima mogućnost razvijanja 3D interaktivnih aplikacija. Zasnovana je na grafu scene u obliku stroge stablaste strukture kao i *Xith3D*.

Početni razvoj ove biblioteke je bio pod nadzorom *Intela*, *Silicon Graphicsa*, *Applea* i *Suna*, no ona kasnije postaje dostupna javnosti za daljnji razvoj. Od 2008. nema poboljšanja što se tiče funkcionalnosti kako se radi na integraciji s *JavaFX*.

Premda *Java3D* postoji duže od *Xith3D*, po funkcionalnosti i karakteristikama su vrlo bliski. *Xith3D* je razvijan na temelju arhitekture *Java3D* biblioteke, pa je moguće uz manje izmjene izvornog koda razvoj u *Xith3D* prebaciti u *Java3D*. Međutim, *Xith3D* je više orijentiran na brže iscrtavanje grafa scene i općenito ima nešto više korisnih komponenata od kojih treba izdvojiti integrirano grafičko sučelje koje je upotrijebljeno u ovom radu. S duge strane *Java3D* ima vrlo dobru zajednicu koja nudi širok spektar izvora informacija o ovoj programskoj potpori. Unatoč tome za projekt je odabrana programska potpora *Xith3D*

Xith3D se pokazao kao vrlo dobar i stabilan sustav, no glavna zamjerka je nešto lošija dokumentacija koja bi bolje objasnila apstraktne koncepte koji se koriste u arhitekturi i odnose pojedinih komponenata. Nadoknada lošijoj opisnoj dokumentaciji je forum Xith3D zajednice. Primijećeni su problemi pri radu i nedostatna fleksibilnost u

razmještanju grafičkih elemenata integriranog grafičkog sučelja, kao i problemi u integraciji s grafičkom sučeljem domaćina putem biblioteke AWT. Također sustav za odabir elemenata u virtualnom prostoru ponekad ne vraća dobar rezultat.

1.2 Web usluge

Pod web uslugama se najčešće podrazumijeva mrežno programsko sučelje koje se dohvaća putem *Hypertext Transfer Protokola* (HTTP). Izvršavanje se obavlja na udaljenom računalu na kojem se nalazi web usluga. Web usluge se dijele na dvije kategorije:

- Velike web usluge (engl. *Big Web Services*) – temelji se na web API-ju koji koristi XML poruke temeljene na SOAP (*Simple Object Access Protocol*) standardu. Poruke, komunikacijske protokole, tipove podataka i popis operacija web usluge se u pravilu definiraju u jeziku WSDL.
- RESTful web usluge – zasnovane na prijenosu reprezentacija resursa. Resurs može biti bilo koji koherentni i smisleni koncept koji je moguće adresirati. Prijenos reprezentacija resursa se uobičajeno vrši prijenosom dokumenta koji opisuje trenutno ili buduće stanje promatranog resursa. Danas je ovo sve češće korišteni tip web usluga zbog dosta prednosti u velikom broju slučajeva uporabe. Glavni razlog ove pojave leži u tome što RESTful usluge koriste i u potpunosti iskorištavaju HTTP protokol. Moguće je koristiti operacije i mogućnosti koje nudi HTTP i nadograditi ga vlastitim funkcionalnostima. To je bitna razlika u odnosu na Velike web usluge. [5]

Postoje tri uobičajena stila korištenja web usluge:

- Pozivanje udaljene metode (RPC) – pristupa se raspodijeljenoj funkciji ili metodi na temelju operacije definirane WSDL-om. Jedna od velikih mana ovog stila korištenja web usluga leži u vezanju usluge uz programski jezik prilikom implementacije. Karakterističan je za prve web usluge.

- Arhitektura orijentirana na uslugu (SOA) – komunicira se porukama umjesto pozivanja udaljenih operacija. Time se odvaja implementacija od komunikacije.
- Prijenos reprezentacija resursa (REST) – koristi HTTP za pristup sadržajima, ograničavajući na skup standardnih operacija koje koristi HTTP (npr. GET, POST, PUT). Interakcija se vrši sa sadržajem koji je adresabilan putem URI-ja i koji ima promjenjivo stanje i to bez pozivanje udaljenih metoda ili komunikacije porukama.

Prema W3C web usluga je programski sustav dizajniran tako da podržava interoperabilnost između računala preko mreže. Sučelje takvog sustava definirano je računalno obradivim formatom (WSDL). Drugi sustavi pristupaju web usluzi na temelju definiranog sučelja SOAP porukama najčešće koristeći protokol HTTP uz XML serijalizaciju u konjunkciji s drugim standardima povezanih s Web-om.[6]

1.2.1 SOAP (*Simple Object Access Protocol*)

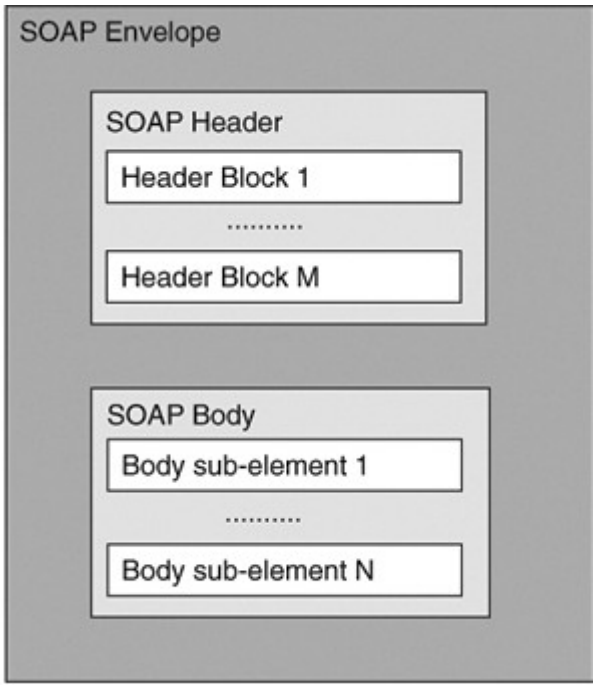
SOAP je specifikacija protokola koja pruža definiciju informacija zasnovanih na XML-u koje se mogu koristiti za prijenos strukturiranih informacija između čvorova decentraliziranog, raspodijeljenog okruženja.

SOAP nudi četiri osnovne komponente sustava[7]:

- standardiziranu strukturu poruke zasnovane na XML Infosetu
- procesni model koji opisuje kako usluga treba obraditi poruku
- mehanizme za povezivanje SOAP poruka s raznim mrežnim protokolima
- način za dodavanje dodatnih enkodiranih informacija u SOAP poruke

SOAP poruka je osnovna jedinica komunikacije između SOAP čvorova. Sastoji se od SOAP ovojnice koja sadrži nula ili više SOAP zaglavlja. Zaglavlja su namijenjena bilo kojem SOAP primatelju koji se možda nalazi na putu SOAP poruke.

SOAP ovojnica isto tako sadrži tijelo SOAP poruke koje je ili korisni sadržaj poruke (engl. payload) ili poslovna informacija. Tijelo SOAP poruke može sadržavati npr. zahtjev za uslugom i ulazne parametre koje usluga treba obraditi.[7] Prilikom obrade SOAP poruke može se dogoditi iznimka. Iznimke su riješene slanjem povratne SOAP poruke koja sadrži SOAP iznimku.



Slika 2: Struktura SOAP poruke

Kod komunikacije pozivanjem udaljenih metoda putem SOAP-a (SOAP RPC) možemo na primjeru pozivanja metode proučiti sadržaje SOAP poruke zahtjeva i odgovora

Pretpostavimo da se poziva udaljena metoda:

```
int doubleAnInteger ( int numberToDouble );
```

Poruke generirane na izvoru i odredištu na temelju XML Infoseta bile bi slijedeće[8]:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
   <SOAP-ENV:Envelope
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
          <SOAP-ENV:Body>
                <ns1:doubleAnInteger
                 xmlns:ns1="urn:MySoapServices">
                       <param1 xsi:type="xsd:int">123</param1>
                </ns1:doubleAnInteger>
          </SOAP-ENV:Body>
   </SOAP-ENV:Envelope>
```

Ispis 1: Primjer SOAP zahtjeva

```
<?xml version="1.0" encoding="UTF-8" ?>
  <SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/1999/XMLSchema">
        <SOAP-ENV:Body>
               <ns1:doubleAnIntegerResponse
                xmlns:ns1="urn:MySoapServices"
                SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
                      <return xsi:type="xsd:int">246</return>
               </ns1:doubleAnIntegerResponse>
        </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Ispis 2: Primjer SOAP odgovora

Analizirati ćemo poruku zahtjeva. Od važnijih elemenata u ovim porukama potrebno je istaknuti element `<SOAP-ENV:Envelope ... >` čiji atribut `xmlns:SOAP-ENV` specificira da je SOAP poruka enkodirana na način definiran XMLShemom na adresi

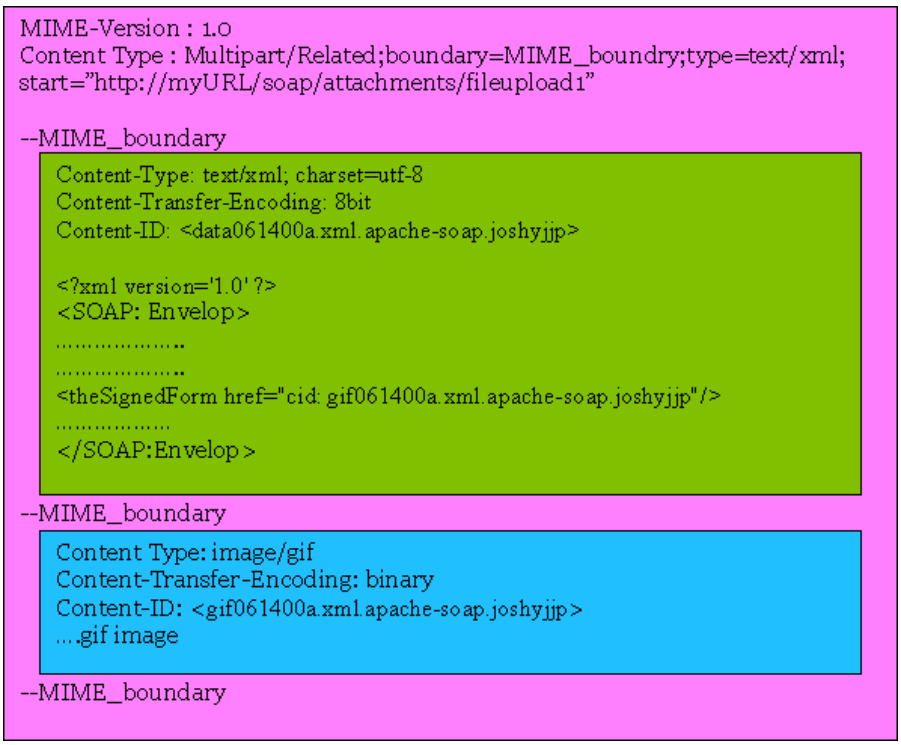
http://schemas.xmlsoap.org/soap/encoding/. Ako nije navedena shema enkodiranja ona se uzima automatski prema SOAP specifikaciji. U primjeru ovojnice se navode još neki standardni prostori imena. Element `<SOAP-ENV:Body>` označava početak tijela poruke, a njegov prvi element-dijete `<ns1:doubleAnInteger ... >` poziv udaljene metode. `ns1` je sufiks za prostor imena usluge koji je u ovom primjeru `urn:MySoapServices`. Slijedi element `<param1 xsi:type="xsd:int">` u kojem se nalazi argument pozvane metode, kao i tip podataka koji se prenosi prema standardnoj xsi *XMLSchemi*.

Poruka s odgovorom je gotovo identična poruci zahtjeva. Bitna razlika je to da sadrži element `<return>` koji u ovom primjeru koristi jednostavan tip podataka koji se vraćaju i vrijednost povratne varijable. Argumenti i povratne vrijednosti mogu biti i složeniji, a definiraju se *XML Infosetom* pri definiranju web usluge jezikom WDSL.

Osim tijela SOAP poruka može sadržavati i priložene binarne datoteke. Binarne datoteke se ne mogu izravno prenositi u XML porukama kao tekst iz razloga što su samo znakovi koje koristi XML dopuštene u porukama. Jedno je rješenje pretvorba niza znakova binarne datoteke u Base64 tip podatka što povećava veličinu poruke u prosjeku 37% i enkodiranje i dekodiranje znakova dugo traje.

To je rješeno dodatnim SOAP specifikacijama: *SOAP Message Transmission Optimization Mechanism* (MTOM) i *XML-binary Optimized Packaging* (XOP). MTOM pruža način identifikacije informacijskih elemenata SOAP poruke koji se mogu optimizirati. XOP specificira način za serijalizaciju tih elemenata i pakiranje u poruku koja se šalje.

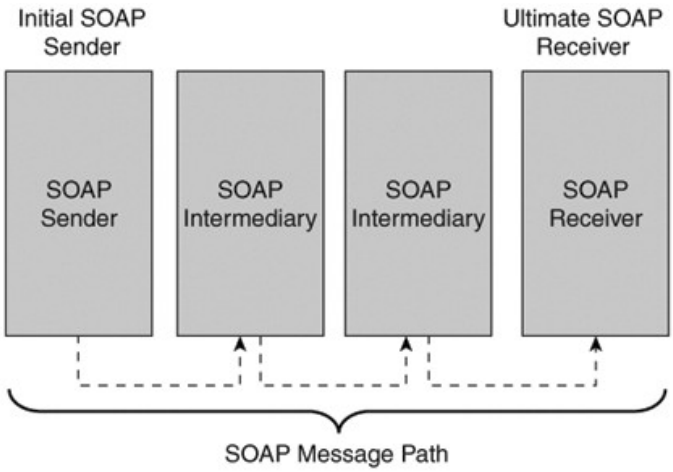
SOAP poruka tada izgleda nešto drukčije, prikazano na slici 3. U primjeru na slici enkodirana je slika, referenca na nju se nalazi u ovojnici SOAP poruke, a sam enkodirani sadržaj kao dodatni dio poruke raspoznatljiv po `Content-ID` elementu zaglavlja MIME dijela poruke.



Slika 3: Struktura SOAP poruke s priloženim binarnim podacima[8]

SOAP čvor je implementacija procesnih pravila opisanih u SOAP specifikaciji. SOAP čvor može slati, primati, obrađivati ili prosljeđivati SOAP poruke. Osim izvršavanja procesnih pravila, čvor može pristupiti i uslugama nižih mrežnih protokola. Pristup je moguć putem SOAP poveznice koja definira pravila za prijenos SOAP poruke povrh nižih mrežnih protokola. Najčešći protokoli su: *HyperText Transport Protocol* (HTTP), *Simple Mail Transfer Prototol* (SMTP) i *Transmission Control Protocol* (TCP).

Svaka SOAP poruka može putovati kroz više SOAP čvorova. Izvor poruke je SOAP pošiljatelj. SOAP poruka potom putuje kroz niz SOAP posrednika koji je prosljeđuju sve do SOAP primatelja. Pri tom procesu postoji mogućnost da se poruka izgubi putem zbog ispada posrednika.



Slika 4: Primjer prolaska SOAP poruke od pošiljatelja do primatelja[7]

1.2.2 WSDL (*Web Services Description Language*)

WSDL je standard za definiranje sučelja web usluga. Izveden je iz XML-a i sadrži ključne informacije o mogućnostima web usluge. WSDL datoteka se sastoji od dva ključna dijela;

- apstraktni dio – opisuje operacijsko ponašanje web usluge, odnosno sadrži definicije poruka koje se izmjenjuju i definiciju udaljenog API sučelja (ulazne, izlazne i poruke iznimke)
- konkretni dio – opisuje lokaciju i način pristupa usluzi – transportni protokol i krajnju točku

WSDL kao takav definira samo komunikacijski dio web usluge bez uplitanja u implementaciju klijentske ili poslužiteljske strane. S druge strane jedan WSDL dokument sadrži dovoljno informacija za stvaranje kostura poslužitelja(*skeleton*) i klijenta(*stub*) iz perspektive komunikacije. Definiranjem pravila komunikacije i API sučelja poslužitelja definiramo i okvir same web usluge.

WSDL ima mogućnost proširenja jezične sintakse, tako da nije ograničen samo na definiranje ključnih informacija za web uslugu. U WSDL-u se mogu nalaziti i definicije složenih tipova podataka koje se koriste u SOAP porukama. Struktura složenih tipova podataka može biti opisana pomoću više opisnih jezika kao što su: *Document Type Definitions* (DTDs), *W3C XML Schema*, i *RelaxNG*. Prema W3C, za definiranje tipova podataka preporuča je XML Schema sustav.

Slika 5: Struktura WSDL 1.1 dokumenta[7]

Cjelokupna struktura WSDL1.1 dokumenta može se vidjeti na slici 5.

```
<types>
  <xsd:schema targetNamespace="http://foo.com"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <element name="ServicePacValidationData">
    <complexType>
     <sequence>
       <element name="Header"
type="ServicePacData:Header"/>
       <element maxOccurs="20" minOccurs="1"
           name="LineItemSegment"
           type="ServicePacData:LineItemSegment"/>
     </sequence>
    </complexType>
   </element>
  <xsd:schema>
</types>
```

Ispis 3: Opis složenog tipa podataka XML Schemom

Korijenski element datoteke je element pod nazivom definicije (engl. *definitions*). Njegovi atributi su prostori imena koji se koriste u dokumentu. Prvi dio poruke definira tipove podataka koji se koriste (engl. *types*). Ispis 3 prikazuje primjer jednog tipa podataka opisanog *XML Schema* dokumentom.

Slijedeći element WSDL dokumenta sadrži popis vrsta poruka koje se prenose (element `message`). Poruke se sastoje od dijelova (element `part`) koji se prenose u poruci. Dio poruke može biti npr. XML dokument, slika ili *Java* objekt. Dijelovi poruke moraju biti razumljivi za parsiranje što se ostvaruje korištenjem definiranih kompleksnih tipova podataka u elementu `<types>`, ako je to potrebno.

```
<message name="ValidationMessage">
   <part name="count" type="xsd:int"/>
   <part name="items" type="tns:ItemType"/>
</message>
```

Ispis 4: Opis naziva poruke i dijelova poruke u WSDL-u

Element tip porta (engl. port type) je imenovani skup koji sadrži popis apstraktnih operacija i apstraktnih poruka koje se koriste. Operacije su opisane imenom, porukama, te smjerom putovanja poruke.

```
<portType name="p1">
   <operation name="op1">
      <input message="x:m1"/>
   </operation>
   <operation name="op2">
      <input message="x:m1"/>
      <output message="y:m2"/>
   </operation>
</portType>
```

Ispis 5: Definicija porta web usluge

SOAP poveznica (engl. SOAP binding) ima namjenu definicije formata SOAP poruka, odnosno opisuje na koji način se stvara ovojnica sadržaja SOAP poruke (`message` elementa SOAP poruke). SOAP poveznica može umetnuti sadržaj SOAP poruke kao dokument – izravno kao dijete `<Body>` elementa SOAP poruke ili kao RPC, gdje je enkodiranje u SOAP poruku nešto drukčije. Osim načina enkodiranja sadržaja, poveznica određuje i transportni protokol koji se koristi pri prijenosu SOAP poruke.

Poveznice se stvaraju na temelju definiranih tipova porta.

```
<binding name="ServicePacValidationBinding"
          type="tns:ServicePacValidationPortType">
   <soap:binding style="document"
           transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="validateServicePac">
      <input>
         <soap:body use="literal"/>
      </input>
      <output>
         <soap:body use="literal"/>
      </output>
   </operation>
</binding>
```

Ispis 6: SOAP poveznica

Posljednji element WSDL dokumenta je definicija usluge (engl. *service*). Usluga je potpuno određena skupom portova koji se sastoje od poveznica i adresa porta usluge.

```
<service name="ServicePacValidationService">
   <port binding="tns:ServicePacValidationBinding"
          name="ServicePacValidationPort">
      <soap:address
           location="http://foo.com/services/SPVPort"/>
   </port>
</service>
```

Ispis 7: Definicija usluge

1.2.3 Axis2

Axis2 je sustav za izvršavanje web usluga uz pomoć protokola SOAP i jezika WSDL. Temeljen je na sustavu *Apache Axis*, te ga je moguće koristiti iz programskih jezika *Java* i *C*. *Axis2* podržava SOAP 1.1 i SOAP 1.2 standarde, no sadržava i integraciju podrške za REST web usluge.

Axis2 je evoluirao iz *Apache Axis* sustava i kao takav donosi poboljšanja u pogledu efikasnosti, modularnosti i većoj orijentiranosti prema XML-u.

Glavna svojstva *Axis2* sustava su[11]:

- brzina – korištenjem vlastitog objektnog modela i parsiranju temeljenom na *StAX* API-ju poboljšane su performanse od prethodnika.
- stabilnost - Sučelje *Axis2* je unatoč mnogim funkcionalnim promjenama i redizajnu arhitekture relativno malo izmijenjeno u odnosu na *Axis*
- AXIOM – je *Axis2* laki objektni model kojim se daje dodatna fleksibilnost prilikom oblikovanja i obrađivanja poruka
- brzo postavljanje – web usluge se mogu pokretati i zaustavljati za vrijeme rada *Axis2* sustava (poslužitelja usluga)
- asinkrone web usluge – web usluge su u *Axis2* sustavu asinkrone, a omogućeno je i asinkrono pozivanje web usluga korištenjem neblokirajućih klijenata

- fleksibilnost i komponentno orijentiran razvoj - arhitektura sustava omogućava potpunu slobodu umetanja raznih proširenja kao što su npr. prilagođeno obrađivanje zaglavlja ili upravljanje sustavom. Implementacije vlastitih proširenja su ponovno iskoristive za prilagođavanje obrađivanja korisničkih aplikacija
- podrška za WSDL – podržani su WSDL 1.1 i WSDL 2.0, iz kojih je moguće generirati *stubove* i *skeletone* web usluge, te pružiti računalu razumljiv opis web usluge

Axis2 je vrlo moćan sustav, a u pogledu razvoja programskog rješenja pogodan je zbog dobre podrške za web usluge zasnovane na SOAP-u. Još jedno pogodno svojstvo je njegova mogućnost rada kao samostalan sustav pokretanjem izvršne datoteke ili rada kao

dio drugog web poslužitelja koji ima podršku za J2EE spremnik za Java servlete.
Pri radu kao samostalan sustav, Axis2 se ponaša kao vrlo jednostavan web poslužitelj koji nudi informacije o web uslugama koje pruža. U spremničkom obliku rada, Axis2 mora biti zapakiran u .jar datoteku koja je u skladu s definiranim Java Servlet API protokolu. Potom je potrebno stvorenu arhivu postaviti u spremnik nekog web poslužitelja kao što je Apache Tomcat. Pokretanjem web poslužitelja pokreće se i Axis2 servlet.

Informacije koje su vidljive na web stranicama Axis2 sustava su imena usluga i operacija koje pojedina usluga nudi u tekstualnom obliku. U spremničkom obliku rada može se dobiti i više informacija o usluzi – opis usluge, krajnja točka komunikacije i sl. Za svaku uslugu je moguće dohvatiti i WSDL dokument koji ju opisuje. Kako je usluga u potpunosti opisana WSDL dokumentom, dohvaćanjem WSDL dokumenta imamo sve potrebne informacije za korištenje usluge.

2 Interaktivni grafički preglednik 3D objekata temeljen na tehnologijama weba

Rast broja web aplikacija i sve veća potreba za interaktivnom grafičkom reprezentacijom raznih apstraktnih i konkretnih elemenata stvarnog svijeta iziskuje potrebu povezivanja i spajanja te dvije ideje. Proces spajanja se odvija već duži niz godina. Web stranice su u početku bile uglavnom tekstualne i sadržavale su kao elemente statične slike, praktički bez ili s vrlo malo interaktivnih sadržaja.

Ubrzo se, pojavom Weba2.0, pojavila potreba za dijeljenjem informacija, mogućnosti međusobne suradnje raznih ljudskih i ne-ljudskih sustava, te dizajn koji sakriva kompleksnosti sustava od korisnika i korisniku nudi jednostavno i moćno interaktivno sučelje. Zajednica je brzo prihvatila ovakav pristup širenju Interneta. Pojavom veza znatno većeg kapaciteta vrlo brzo Internet počinje vrvjeti audio i video sadržajima, animacijama i interaktivnim aplikacijama.

Interaktivnost sadržaja daje korisnicima osjećaj sudjelovanja u nečemu, dok je vizualno privlačno sučelje maskira sustav. Uz pravilno korištenje elemenata vizualizacije može se na neki način proširiti kapacitet komunikacijskog kanala između računala i čovjeka. Korištenjem dobre tehnike vizualizacije skupa podataka korisnik može u znatno kraćem vremenu primijetiti njemu relevantnu informaciju u odnosu na tekstualni element. U današnje vrijeme to je iznimno važno zbog gotovo konstantnog ubrzavanja frekvencije pojavljivanja raznoraznih događaja oko nas, kao i gomilanje ogromnih količina nerelevantnih ili neprikladno prikazanih informacija koje zagušuju čovjekovu percepciju.

Rješenje problema zagušenja ljudske percepcije je sustavno filtriranje informacija uz kvalitetnu vizualizaciju i interaktivno sudjelovanje korisnika usluge. Sustavno filtriranje informacija je vrlo kompleksno. Ono podrazumijeva korisničko definiranje njemu relevantnih informacija, te uspoređivanje te definicije sa skupom poznatih podataka/informacija. U ovom radu će se ovaj problem sagledati iz perspektive vizualizacije i interaktivnosti.

2.1 Idejno rješenje

Osim sustavskih programa i ozbiljnijih računalnih igara gotovo sve nove aplikacije se razvijaju kao web aplikacije, tzv. Rich Internet Applications (RIA) koje često u pozadini koriste raspodijeljenost mrežnih sustava za obavljanje neke operacije. Time se mogu postići bolje performanse ili pak podržati veći broj korisnika. Postoji mnogo prednosti korištenja raspodijeljene mrežne arhitekture.

Za izradu preglednika 3D objekata prednost korištenja raspodijeljene arhitekture je u mogućem raspoređivanju opterećenja u smislu generiranog prometa na više poslužitelja. Prijenos 3D modela, te njima pripadajućih tekstura može stvoriti velik mrežni promet. Pri tome je svaki poslužitelj (čvor) u raspodijeljenoj mreži zadužen za svoj skup 3D objekata. Dobar primjer primjene takvog sustava, koji je i osnovna motivacija za izradu projekta, je pregledavanje 3D modela proizvoda. Osnovna problematika je kako strukturirati trodimenzionalni model i prikazati ga korisniku. Kod preglednika proizvoda osnovna struktura grafa scene je nadopunjena meta podacima o samom proizvodu. Glavno je pitanje kako virtualno oblikovati neki proizvod. Očigledno rješenje je modeliranje proizvoda i njegovo prikazivanje u npr. web pregledniku ili nekom obliku klijentske aplikacije.

Modeliranje 3D modela iziskuje puno vremena i stručnosti, time i novaca, i to je jedan od važnijih razloga zašto ne predstavljati proizvode kao 3D modele. Međutim i taj je problem moguće ublažiti. Ako ponovno pogledamo neki proizvod, primjerice mobilni telefon. On se sastoji od LCD displeja, baterije, kućišta, raznih konektora... Postoji dobra vjerojatnost da je neka druga kompanija razvila barem neke vidljive komponente tog mobitela. Čak i ako to nije slučaj, često se iste kompanije komponente koriste u više modela nekog proizvoda, odnosno standardizirane su unutar kompanije.

Ponuđeno rješenje bi bilo da svaka kompanija koja bi bila u raspodijeljenoj mreži nudila vlastite proizvode koji bi mogli poslužiti kao komponente nekih drugih proizvoda. Ponovno iskorištavanje komponenata bi smanjilo potrebna ulaganja u modeliranje, pa možda čak i svela modeliranje proizvoda na pozicioniranje već postojećih komponenata proizvoda u virtualnom prostoru.

Klijentska aplikacija bi povezivanjem na poslužitelj nekog proizvođača mogla vidjeti koje proizvode proizvođač nudi, odnosno mogla bi preuzeti njihove opise i modele. Korisnik bi tada mogao interaktivno obavljati određene unaprijed definirane operacije nad 3D modelom, odnosno operacije nad proizvodom ili njegovim komponentama. Svaki proizvod bi imao vlastiti opis i skup osnovnih informacija o njemu. Komponente proizvoda bi također imale skup informacija koje ih definiraju. Kako je komponenta istovremeno i proizvod, ovisno o perspektivi promatranja, tip skupa informacija koje opisuju proizvod opisivale bi i svaku komponentu.

2.2 Preglednik 3D modela

2.2.1 Obrasci upotrebe preglednika 3D modela

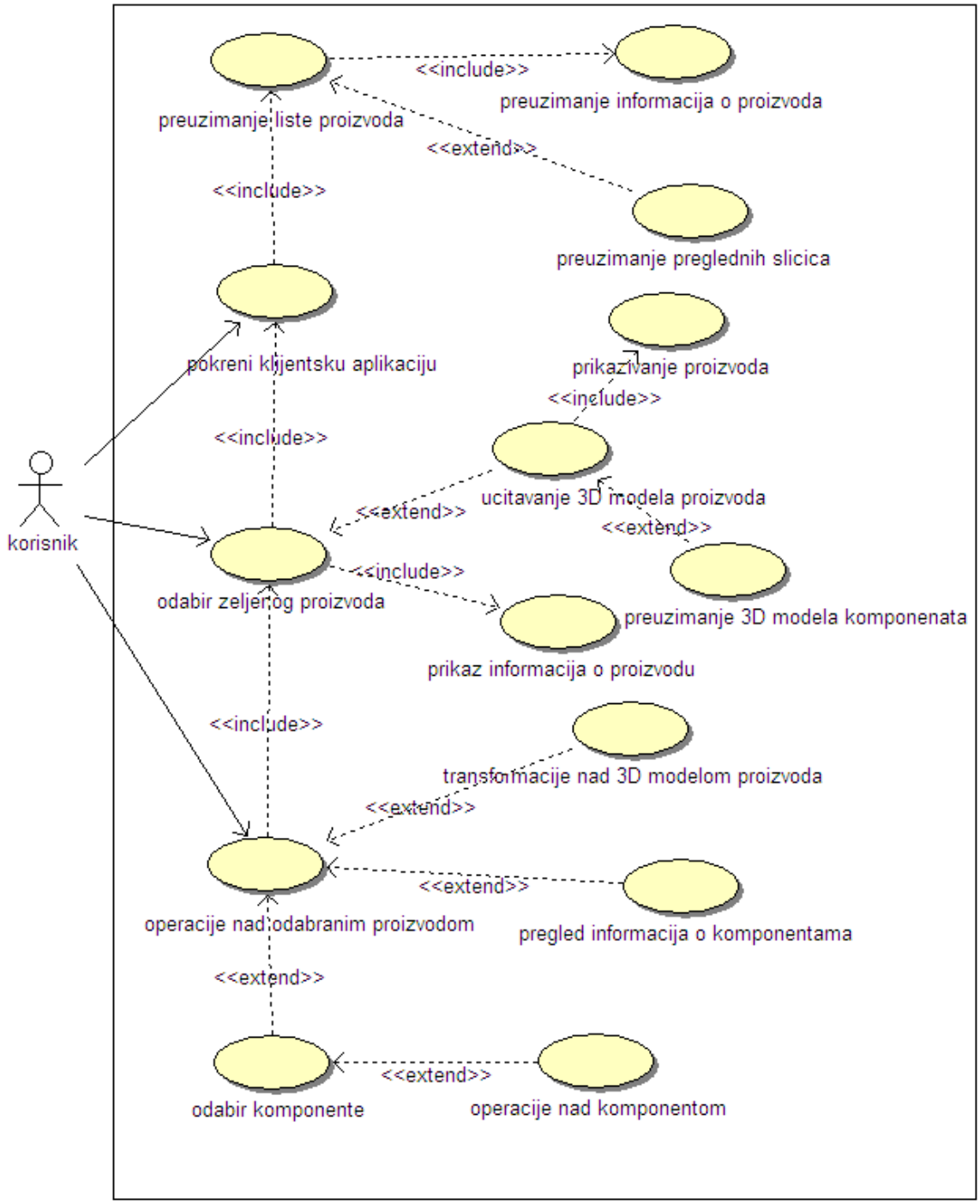
Za početak ćemo definirati kakvo će sučelje klijentska aplikacija imati prema krajnjem korisniku sustava na nešto detaljnijoj razini. Korisničke interakcije se mogu svesti na tri mogućnosti (dijagram 1). To je pokretanje klijentske aplikacije, odabir željenog proizvoda i obavljanje operacija nad odabranim proizvodom. Te su tri moguće interakcije međusobno zavisne.
Detaljniji pogled na moguće interakcije:

- pokretanje klijentske aplikacije – preduvjet za korištenje aplikacije, važan korak iz perspektive razumijevanja toka unutarnjih promjena stanja sustava. Prilikom pokretanja preglednika potrebno je dohvatiti informacije o proizvodima na temelju unaprijed poznate krajnje točke SOAP web usluge. Neki proizvodi mogu imati sličicu za brzi pregled proizvoda. Ako ju imaju, i ona se preuzima.
- odabir željenog proizvoda - prilikom odabira proizvoda prikazuju se informacije o proizvodu i sličica za brzi pregled proizvoda. Sličica za brzi pregled i općenite informacije o proizvodu trebali bi biti dovoljni korisniku da na temelju njih odluči želi li još detaljniji pregled proizvoda u obliku 3D modela. Ako želi, potrebno je

provjeriti jesu li sve komponente raspoložive lokalno. Ako jesu one se učitavaju iz lokalnog repozitorija. U slučaju da komponente nisu raspoložive lokalno kontaktira se krajnja točka web usluge i počinje proces dobavljanja potrebnih komponenata. Učitani model se prikazuje korisniku nakon uspješnog učitavanja

- operacije nad odabranim proizvodom – sve su operacije opcionalne, a preduvjet je da je proizvod odabran, odnosno da je trodimenzionalni model učitan u preglednik. Prva moguća operacija nad proizvodom je transformiranje 3D modela. Moguće su transformacije rotiranja oko globalnih osi i transformiranje pozicije centra rotacije modela proizvoda.

  Prelaskom tipke miša preko komponenata proizvoda prikazuje se tekstualni skup informacija o komponenti. Odabirom komponente korisnik može odabrati i neku od implementiranih operacija nad tom komponentom (npr. sakrivanje modela komponente ili neka druga vizualna operacija nad odabranim čvorom grafa scene)



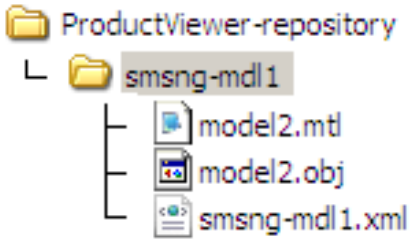
Dijagram 1: Obrasci upotrebe klijentske aplikacije

2.2.2 Opis rješenja preglednika 3D modela proizvoda

Osnovna klasa preglednika 3D modela proizvoda zove se `ProductViewerClient`. Instanciranjem te klase kreira se *Xith3D* okruženje. *Xith3D* okruženje interno stvara prozor unutar operacijskog sustava i kreira se površina *OpenGL* konteksta unutar kojeg se odvija isctravanje scene. Osnovni parametri aplikacije nalaze se unutar klase `GlobalSettings` kao globalno dostupne statične varijable. Nakon pridruživanja korijenskog čvora tipa `BranchGroup` *Xith3D* okruženju stvara se objekt tipa `InputEventsHandler` koji upravlja korisničkim unosima s tipkovnice. Kako je klijentska aplikacija namijenjena krajnjem korisniku sve se izvršava na temelju korisničkog unosa, pa `InputEventsHandler` ujedno ima i centralnu ulogu u pregledniku. Uloga klase `InputEventsHandler` je višestruka; osluškivanje korisničkih unosa s tipkovnice i miša, obavljanje općenitih operacija na temelju pritisnute tipke na tipkovnici, upravljanje grafičkim sučeljem i odabirom elemenata u virtualnom prostoru na temelju podataka o trenutnom stanju miša.
Naposljetku se korijenskom čvoru *Xith3D* okruženja dodaje i grafičko sučelje tipa `HUD`. Sučelje se sastoji od više grafičkih elemenata koji omogućavaju interaktivno upravljanje objektima u sceni. Grafičko sučelje je detaljnije opisano u poglavlju .

Repozitorij sadržaja

Prilikom instanciranja objekta tipa `ProductViewerClient` stvara se i lokalni repozitorij. Klasa zadužena za upravljanje lokalnim repozitorijem se naziva se `ClientModelsRepository`. Zadaća `ClientModelsRepository`-a je stvoriti u datotečnom sustavu direktorij koji će sadržavati 3D modele proizvoda, kao i datoteke s informacijama o proizvodima i sličice za brzi pregled ako postoje.
Na slici 6 se može vidjeti struktura jednog takvog zapisa u repozitoriju. Zapis sadrži XML datoteku s opisom proizvoda, te OBJ datotekom koja je trodimenzionalni model proizvoda ili jedne komponente i odgovarajuće MTL datoteke koja opisuje materijale 3D modela.



Slika 6: Primjer proizvora u repozitoriju

Kod upravljanja proizvodima u repozitoriju pojedini proizvodi se razlikuju na temelju ključa, po uzoru na baze podataka. Kao ključ, odnosno identifikator nekog proizvoda (ili komponente) je odabrano njegovo ime modela proizvoda – u primjeru: smsng-mdl1. Za svaki unos u repozitorij stvara se direktorij s imenom modela proizvoda koji mora odgovarati imenu XML datoteke koja opisuje unos. Repozitoriji na klijentskoj i poslužiteljskoj strani imaju slične karakteristike, no različite namijene. Poslužiteljski repozitorij služi samo za čitanje u svrhu isporuke datoteka, dok je klijentski repozitorij namijenjen za lokalnu pohranu i kasnije čitanje dohvaćenih datoteka s poslužitelja. Osim toga, klijentski repozitorij ne može samostalno raditi pošto ne zna koje komponente u lokalnom repozitoriju jesu proizvodi. Ta se informacija traži od poslužiteljskog repozitorija. Klijentski i poslužiteljski repozitoriji su usko vezani komunikacijom putem mreže.

Ako su sve potrebne datoteke u lokalnom klijentskom repozitoriju, tek je tada moguće iscrtati scenu. U slučaju da je potrebno, klijentski repozitorij koristi instancu klase `PVSoapClient` da bi komuniciranjem s poslužiteljima dohvatio datoteke koji nedostaju. Izmjena datoteka se vrši na razini proizvoda, odnosno jedna transakcija prenosi sve datoteke za jedan proizvod.

Klasa `Model je` reprezentacija proizvoda unutar preglednika. Repozitorij se ponaša kao tvornica objekata tipa `Model`. Time je onemogućeno učitavanje proizvoda izvan repozitorija i eliminirane su potencijalne pogreške koje bi mogle nastati takvim učitavanjem.

Opis modela pomoću XML datoteka

Svi su proizvodi/komponente opisani pomoću XML datoteka(ispis 8). XML datoteka se preslikava u klasu `ComponentHeader` koju koriste klase `Model` i `Component` i predstavlja svojevrsno zaglavlje za proizvode i modele.

Korijenski element je pod nazivom `model`, a sastoji se od obaveznih atributa; `name` – ime proizvoda, `ID` – niz znakova na temelju kojih se modeli (proizvodi) razlikuju, `sourceID` – izvor modela, u primjeru preglednika proizvoda ekvivalent proizvođaču.

Prvi element-dijete `description` sadrži tekstualni element s opisom proizvoda. Slijedi element `components` koji sadrži listu komponenata od kojih se sastoji proizvod.

Svaka komponenta obavezno mora imati atribute `compositeComponent`, `ID` ili `model3d`, te `sourceID`. Element `compositeComponent` definira je li trenutni model sastavljen od više komponenti koje se mogu odabirati unutar preglednika. Time se ograničava dubina do koje se mogu odabirati čvorovi u grafu scene.

Ako postoji atribut `ID`, tada se pri stvaranju grafa scene učitava komponenta s tim identifikatorom iz repozitorija. Komponente koje imaju atribut `model3d` su zapravo listovi grafa scene i oni fizički pokazuju na datoteku koja sadrži trodimenzionalni model neke komponente. Atribut `sourceID` unutar elementa `component` koriste poslužitelji usluge da bi mogli uz pomoć tablice s krajnjim točkama komunikacije kontaktirati zadužene poslužitelje.

Elementi `translation`, `rotation` i `scale` sadrže tekstualne zapise s transformacijama čvora grafa scene. Opcionalno, komponenta može imati i elemente `opearations` i `effects`. Elementi tipa `operations` su lista operacija koje je moguće izvršiti nad tim čvorom (komponentom) u grafu scene. U primjeru je navedena samo jedna operacija za prvu komponentu – `visibility`. Element `effects` je jednostavan element koji se sastoji od dva atributa – `vs` (program za sjenčanje vrhova, engl. *vertex shader*) i `fs` (program za sjenčanje fragmenata, engl. *fragment shader*). Svaki od ta dva atributa je neobavezan i pokazuje na odgovarajuću datoteku s programom za sjenčanje vrhova ili fragmenata. Programi za sjenčanje daju osobi koja modelira veliku slobodu u grafičkoj reprezentaciji nekog modela, odnosno omogućava postizanje

specijalnih efekata nad pojedinim dijelom ukupnog modela oblikovanog pomoću XML datoteka. Programi dodijeljeni nekom čvoru propagiraju sve do listova koji sadržavaju geometriju i izvršavaju se nad geometrijom.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <model xmlns="hr.fer.df41962" sourceID="sourceID"
       ID="cell-m1" name="Cellphone">
    <description />
    <components>
      <component compositeComponent="false" sourceID="sourceID" ID="topp-m1">
        <translation>0.0 0.65 0.0</translation>
        <rotation>-0.0 -0.0 -0.0</rotation>
        <scale>1.0 1.0 1.0</scale>
        <operations name="visibility">
                    <arguments />
        </operations>
        <effects fs="shader.fs" vs="shader.vs" />
      </component>
    <component compositeComponent="false" sourceID="sourceID" model3d="Back.obj">
      <translation>0.0 -0.4 0.0</translation>
      <rotation>-0.0 -0.0 -0.0</rotation>
      <scale>1.0 1.0 1.0</scale>
    </component>
    </components>
</model>
```

Ispis 8: Primjer proizvoda opisanog pomoću XML-a

Proizvodi (klasa `Model`) i komponente (klasa `Component`) su izvedeni iz *Xith3D* klase `TransformationGroup`. Ta klasa reprezentira čvorove grafa scene koje je moguće transformirati. Uz naslijeđene podatke o samoj geometriji i transformacijama, klasa je proširena i korisnim informacijama vezanim uz primjenu pregledavanja proizvoda.

Trodimenzionalni model strukturiran pomoću XML datoteka je sama jezgra ovog preglednika 3D modela proizvoda. Svaki čvor se može reprezentirati pomoću takve datoteke i potom preslikati u graf scene. Osim toga, relativno je jednostavno dodati proširenja u pogledu elemenata XML-a, odnosno zaglavlja, koji bi omogućili drukčiju namjenu preglednika.

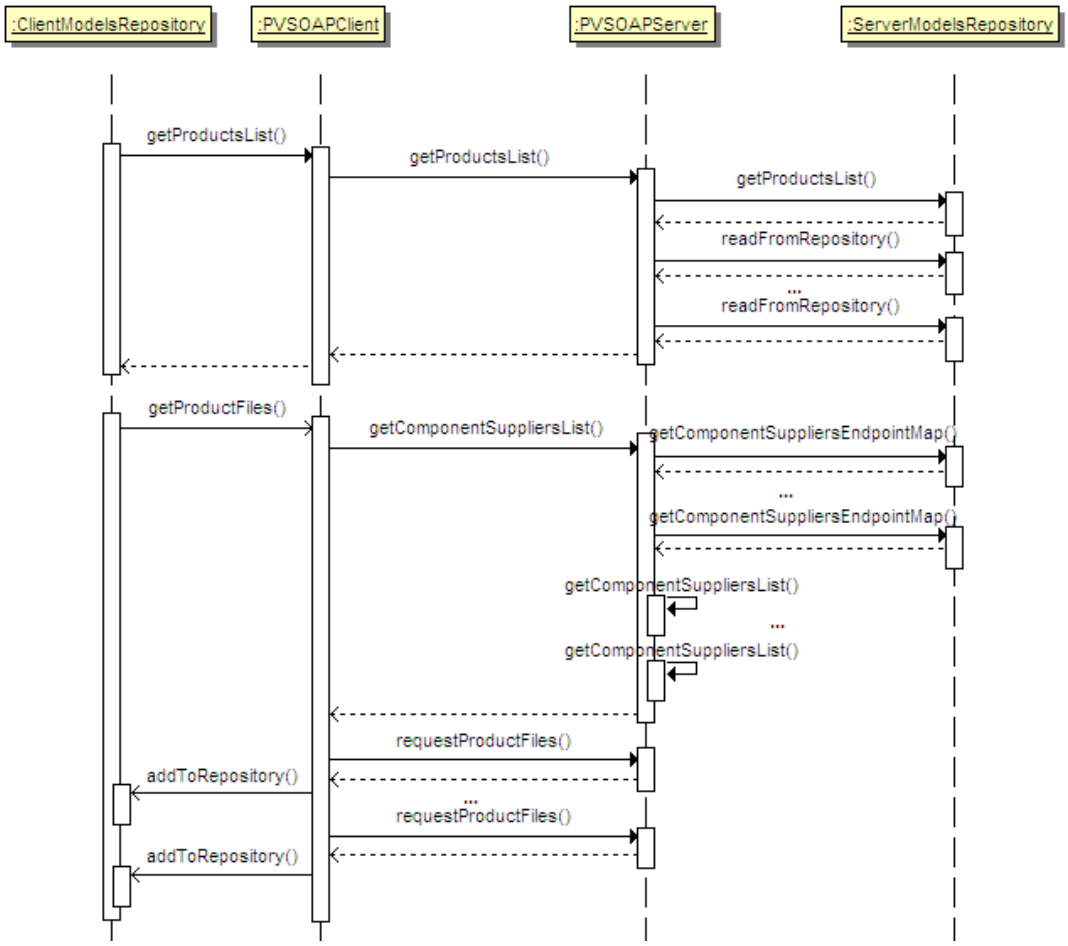
Operacije nad komponentama

Operacije nad komponentama su moguće ako je unutar XML datoteke proizvoda za određenu komponentu definirana operacija. Ime operacije se mora podudarati s imenom neke od implementiranih i registriranih operacija u pregledniku proizvoda. Registriranje se obavlja nad objektom `OperationManager` pozivom metode `registerOperation()`. Za primjer operacija `visibility` je operacija koja mijenja stanje vidljivosti čvora u grafu scene i implementirana je pod imenom `VisibilityOperation`.

Svaka implementirana operacija mora biti izvedena iz sučelja `Operation`. Sučelje ima samo dvije metode; `initOperation()` koja vraća objekt tipa `Panel` i `getType` čija je povratna vrijednost niz znakova koji predstavlja ime operacije. `Panel` je klasa Xith3D-a koja predstavlja površinu za iscrtavanje grafičkog sučelja. Prilikom aktiviranja operacije pokreće se metoda `initOperation()` koja mora vratiti površinu s grafičkim elementima kao što su gumbi, klizne trake i sl. U isto vrijeme se grafički elementi vežu uz trenutno aktivan čvor grafa scene i ovisno o operaciji oni omogućuju korisniku aplikacije interaktivno upravljanje odabranim čvorom. Operacija navedena u XML datoteci može preuzeti i argumente iz elementa `operations` u obliku teksta, ako je to potrebno za tu operaciju.

2.3 Komunikacija i poslužitelj modela

Interakcija preglednika i poslužitelja je prilično jednostavna. Osnovni protokol po kojem se odvija komunikacija se može vidjeti na dijagramu toka (dijagram 2). Između klasa `PVSOAPClient` i `PVSOAPServer` i pri refleksivnim pozivima na klasi `PVSOAPServer` se komunikacija odvija prema SOAP specifikaciji putem mreže.



Dijagram 2: Tok komunikacije između klijenta i poslužitelja

SOAP *stub* `PVSOAPClient` pruža dvije moguće metode; `getProductsList()` i `getProductFiles()`.

`GetProductsList()` je jednostavna metoda koja samo prosljeđuje upit SOAP *skeletonu* `PVSOAPServer`. *Skeleton* od poslužiteljskog repozitorija dohvaća popis proizvoda i kreira poruku s listom proizvoda. Slijedeći je korak da se za svaki od proizvoda XML datoteke i sličice za brzi pregled dodaju kao MIME privitak SOAP poruci. SOAP poruka s privicima se vraća *stubu*.

Metoda `getProductFiles()` je nešto kompleksnija. Poziva se kada je potrebno učitati trodimenzionalni model, a potrebne datoteke nisu raspoložive lokalno kod klijenta. Prvi je korak prikupljanje svih potrebnih točaka za komunikaciju odgovarajućih identifikatora komponenti koje je potrebno preuzeti na temelju identifikatora proizvoda

koji se prenosi kao argument. SOAP *skeleton* za svaku komponentu koju on posjeduje upisuje svoju adresu, a za sve koje nema kontaktira udaljeni *skeleton* koji je zadužen za tu komponentu. Zaduženosti se znaju prema imenu proizvođača (identifikator *sourceID*) i mapi koja proizvođača preslikava u krajnju točku SOAP komunikacije, a koja postoji na poslužitelju kao datoteka *sources.xml.* Ova se komunikacija može odvijati rekurzivno sve dok se ne dobiju sve krajnje točke i identifikatori potrebnih komponenata. Moguće je pojavljivanje višestrukih unosa u popis, što se eliminira korištenjem mape. Pretpostavlja se da dva proizvođača nemaju proizvode s istim identifikatorom, u suprotnom je ovakva eliminacija višestrukih unosa neprikladna. Moguća su preklapanja modela u repozitoriju.

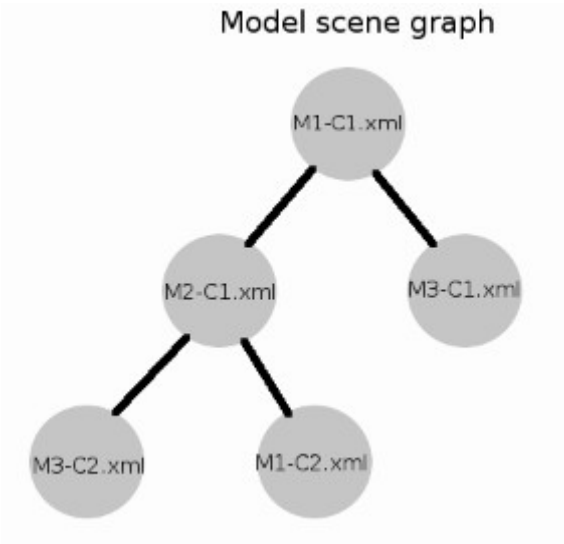
Nakon što *stub* ima popis svih komponenata i popis krajnjih točaka od kojih mora tražiti datoteke poziva se udaljena metoda `requestProductFiles()` s identifikatorom proizvoda kao argumentom. Potrebne se datoteke stavljaju kao privici u SOAP poruku i prenose; po jedna poruka po jednom zahtjevu. Po primitku svakog odgovora kreira se novi unos u klijentskom repozitoriju.

Primjer dohvaćanja potrebnih komponenti proizvoda

Pretpostavimo da imamo proizvod koji se sastoji od komponenata koje su opisane XML datotekama čija se imena nalaze unutar kružnica na slici 7. Identifikatori modela proizvoda odgovaraju imenima XML datoteka i to tako da je svaka komponenta jedinstvena u sustavu i vezana uz proizvođača. Struktura grafa scene koji je opisan XML datotekama je također vidljiva na slici.

Klijentu je poznata samo krajnja točka komunikacije proizvođača proizvoda koji se zahtijeva i identifikator modela proizvoda koji želi preuzeti. To znači da sveukupna struktura grafa scene klijentskoj aplikaciji nije poznata.

Struktura grafa scene je nebitna za preglednik, važno je samo da sve komponente koje su potrebne za stvaranje konačnog grafa scene postoje u lokalnom klijentskom repozitoriju. Da bi se one preuzele treba znati identifikatore potrebnih komponenata i krajnje točke komunikacije.



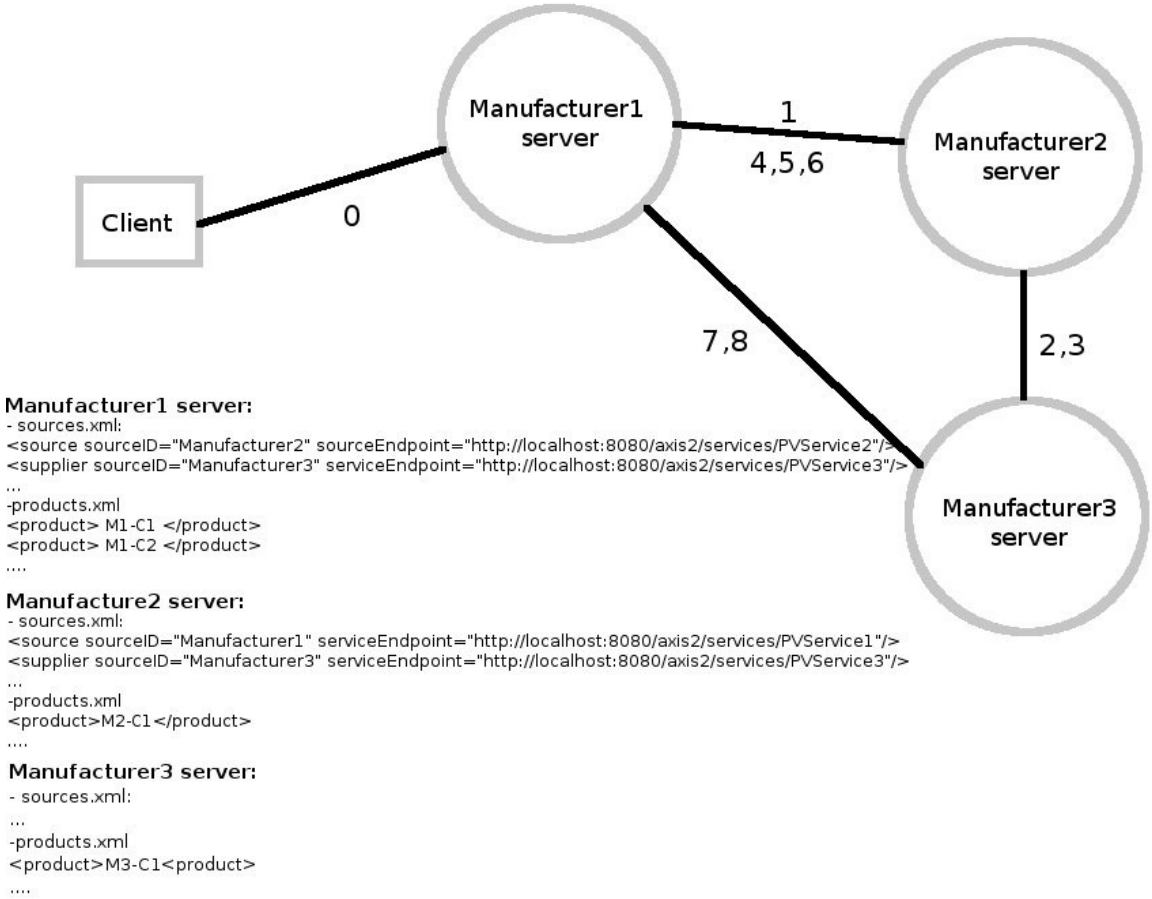
Slika 7: Struktura grafa scene modela proizvoda

Kreiranje tog popisa je prikazan na slici 8. Svaki poslužitelj ima popis poznatih proizvođača i popis proizvoda koje nudi. Za primjer bitni unosi u te datoteke su prikazani na slici 8.

Primitkom zahtjeva za listom parova identifikator komponente - krajnja točka komunikacije(0) poslužitelj 1 iz repozitorija deserijalizira XML datoteku s proizvodom za koji je potrebno vratiti listu(M1-C1). U listu se dodaju parovi identifikator komponente - krajnja točka komunikacije za poslužitelje 2 i 3 na temelju podataka o komponentama iz učitane M1-C1.xml datoteke. Prva komponenta proizvoda M1-C1 sadrži identifikator komponente M2-C1 čiji je proizvođač manufacturer2. Iz datoteke *sources.xml* čita se krajnja točka komunikacije za poslužitelj 2. Poslužitelj 1 kontaktira poslužitelj 2 sa zahtjevom za listom parova identifikator komponente - krajnja točka komunikacije za proizvod M2-C1(1). U povratnu listu poslužitelja 2 se dodaju krajnje točke komunikacije za poslužitelje 1 i 3 na isti način kao i kod poslužitelja 1. Poslužitelj 2 zahtjeva od poslužitelja 3 listu za komponentu M3-C2(2). Kako je to list stabla grafa scene, poslužitelj 3 vraća *null*(3). Poslužitelj 2 ponavlja istu operaciju za M1-C2 na poslužitelju 1(4,5) i dobiva isti odgovor. Po prolasku kroz sve komponente poslužitelj 2 vraća stvorenu listu poslužitelju 1(6). Poslužitelj 1 dodaje nepraznu listu svojoj listi uz uklanjanje višestrukih unosa i traži od poslužitelja 3 listu za M3-C1. Po dobivenoj *null* listi klijentu se vraća lista stvorena na poslužitelju 1.

Klijent sada ima skup parova identifikator komponente – krajnja točka komunikacije, pa se

dohvaćanje komponenata svodi na slijedno kontaktiranje poslužitelja sa zahtjevima za datotekama proizvoda u njihovim repozitorijima.



Slika 8: Primjer dohvaćanja popisa krajnjih točaka komunikacije

Sustav je oblikovan tako da postoji mala međuovisnost između grafa scene i podataka u repozitorijima. Graf scene opisan je u datotekama koje se prenose SOAP porukama, što omogućuje prenamijenu sustava uz male izmjene. XML datoteke se mogu proširiti dodatnim elementima bez potrebe mijenjanja poslužitelja. Kako su poslužitelji jednostavni postoji potreba za većom obradom na klijentskoj strani što rezultira kompleksnijom klijentskom aplikacijom (engl. *fat client*). Deserijaliziranje na klijentskoj strani mora biti u stanju prepoznati sve elemente unutar XML datoteke i poduzeti odgovarajuće akcije na temelju njih.

Problem koji se može pojaviti u ovakvom sustavu su petlje u komunikaciji što je vrlo ozbiljna mana. Problem je tim veći što SOAP usluge same po sebi ne pamte stanja što

dodatno komplicira rješenje problema. Strukture grafova scena nisu poznate poslužiteljima, pa ako je je neki od grafova scene loše oblikovan to se može izravno preslikati u komunikacijsku petlju. Problem se može pojaviti samo prilikom zahtjeva za listom parova identifikator komponente – krajnja točka komunikacije. Vjerojatno najprikladnije rješenje bilo bi prebacivanje dohvaćanje te liste na klijenta koji bi rekurzivno zahtijevao listu od poslužitelja uz praćenje postojanja petlje u grafu scene i prekidanje u tom slučaju.

Serijaliziranje i deserijaliziranje XML datoteka obavlja sustav za povezivanje XML-a JAXB. JAXB je standardni dio programskog jezika Java i omogućuje povezivanje Java objekata s elementima XML datoteka efektivno preslikavajući XML u objekte i obratno. Preslikavanje je opisano *XMLSchema* dokumentom iz koje su generirani odgovarajući objekti.

2.3.1 Uputstvo za korištenje preglednika proizvoda

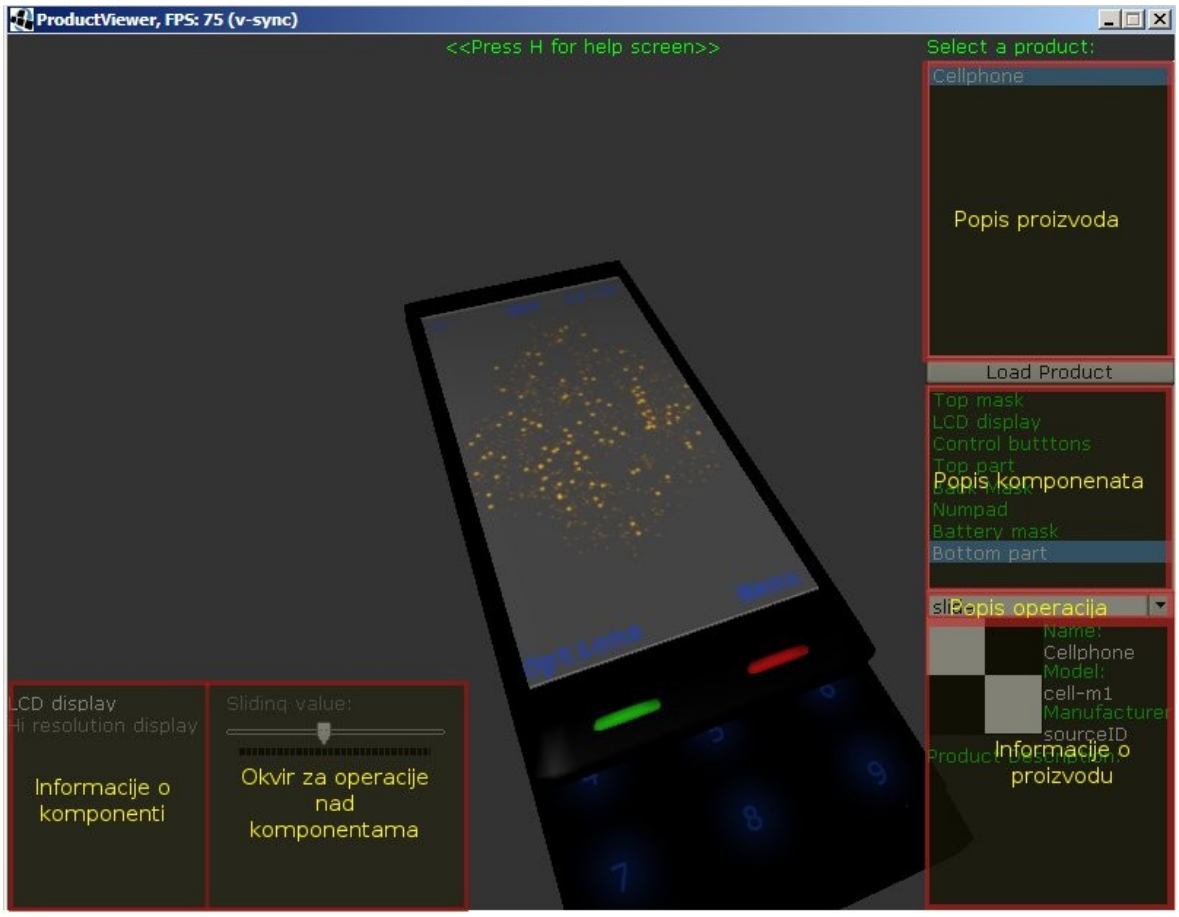
Sučelje se sastoji od izbornika proizvoda koji se popunjava popisom proizvoda preuzetim s poslužitelja čija je krajnja točka komunikacije poznata. Proizvodi koji nisu lokalno dostupni označeni su žutom bojom i njih je potrebno preuzeti s poslužitelja. Klikom na gumb *Load Product* počinje preuzimanje. Zelenom bojom označeni su proizvodi koji se mogu odmah pregledati. Ispod izbornika proizvoda nalazi se izbornik komponenata odabranog proizvoda. Padajući izbornik sadrži popis mogućih operacija za odabranu komponentu. Donji desni kut sadrži osnovne informacije o proizvodu i sličicu za brzi pregled koji se preuzimaju zajedno s popisom proizvoda.

Informacije o odabranoj komponenti mogu se vidjeti u donjem lijevom kutu. Nakon odabira komponente i željene operacije stvara se okvir koji se sastoji od elemenata grafičkog sučelja kojima je moguće obavljati operacije nad odabranom komponentom.

Komponente se mogu odabrati i klikom lijeve tipke miša na njihov trodimenzionalni prikaz u sceni. Prelaskom kursora miša preko komponente privremeno se prikazuju informacije o komponenti.

Držanjem desne tipke miša i povlačenjem kursora po prostoru za iscrtavanje je moguće rotirati objekt, a efekt približavanja ili udaljavanja objekta od korisnika postiže se

kotačićem na mišu. Više informacija o korištenju aplikacije može se dobiti pritiskom na tipku *h* u aplikaciji. Izgled sučelja može se vidjeti na slici 9.



Slika 9: Raspored elemenata grafičkog sučelja

2.3.2 Izrada modela proizvoda za preglednik

Za izradu modela za preglednik je napisan vrlo jednostavan uređivač modela. Pomoću njega je moguće vizualno rasporediti elemente unutar grafa scene unošenjem parametara transformacije u tekstualna polja. Osim pozicioniranja komponenti proizvoda omogućuje i unos imena proizvoda, identifikatora, proizvođača (identifikatora izvora – *sourceID*). Uređivač je vrlo skromnih mogućnosti i napisan je prvenstveno za oblikovanje primjera za demonstraciju. Operacije i efekte (sjenčanje) e također moguće definirati u uređivaču.

Gumb *LoadModel* učitava trodimenzionalni model iz datoteke u graf scene dodajući ga izravno u korijen novog proizvoda. Model mora biti poranjen u odgovarajućoj datoteci koja odgovara identifikatoru proizvoda iz razloga što se model može sastojati od više datoteka (npr. datoteke tekstura i materijala), što uređivač ne može prepoznati. Klikom na model on postaje aktivan. Nakon što je model aktivan odabirom odgovarajuće transformacije u uređivaču lijevim klikom miša na model, upisom x, y i z komponente transformacije i klikom na tipku *Apply* ta transformacija se primjenjuje na model i vizualno prikazuje u sceni. Gumb *LoadComponent* učitava XML datoteku s opisom komponente. Komponenta se dodaje u korijen proizvoda i ponaša se isto kao i model. Transformacije na komponentu se primjenjuju na isti način kao i kod trodimenzionalnog modela. Komponenta može sadržavati jedan ili više listova, pa se transformiranjem komponente transformiraju i svi listovi komponente. *LoadProduct* učitava cijeli proizvod kao novi korijenski čvor.

Podaci o proizvodu se upisuju u tekstualna polja *name*, *model*, *manufacturer* i *description*. Polje *model* je identifikator proizvoda, a *manufacturer* se preslikava u *sourceID* element u XML datoteci.

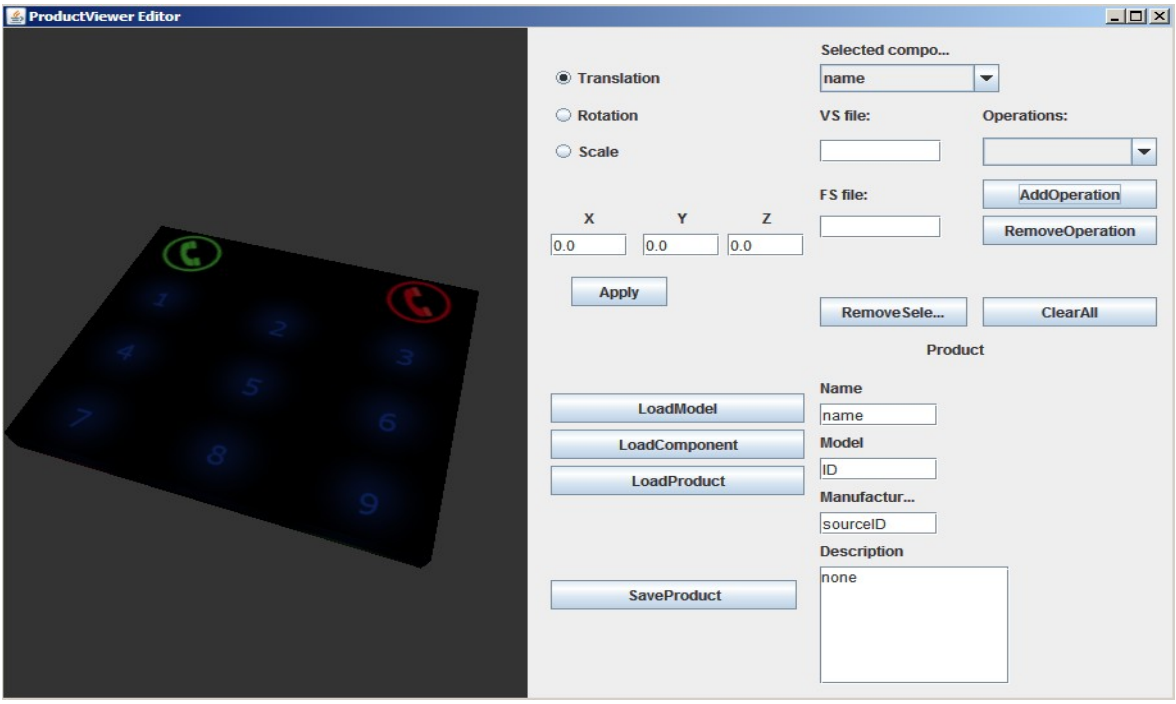
Operacije je moguće definirati odabirom komponente nad kojom se želi definirati operacija klikom na komponentu unutar preglednika ili odabirom iz padajućeg izbornika. Klikom na gumb *AddOperation* se otvara novi prozor u kojem je potrebno napisati ime i argumente operacije. Za svaku je komponentu moguće definirati više operacija. Operacija mora biti implementirana u pregledniku, inače se ignorira. Efekti nad komponentom se dodaju upisom naziva datoteka s efektima u kućice *VS file* i *FS file.* Te se datoteke moraju

nalaziti unutar direktorija proizvoda. Svaka se promjena mora potvridit tipkom *Apply*.

Gumb *RemoveSelected* uklanja odabranu komponentu ili model, a *ClearAll* uklanja sve čvorove osim korjenskog.

Klikom na *SaveProduct* se pohranjuju svi podaci o proizvodu i sve transformacije komponenata u odgovarajući direktorij u repozitoriju na temelju identifikatora proizvoda. Svi modeli i odgovarajuće datoteke s teksturama, opisom materijala i slično moraju se dodati u direktorij tog proizvoda u repozitoriju. Podržani modeli su formata OBJ i MD2. Preporuča se koristiti format OBJ. Komponente koje se učitavaju u uređivač moraju biti u repozitoriju uređivača proizvoda.

Sličica za brzi pregled nije obavezna, a može se dodati kopiranjem slike u direktorij proizvoda. Podržan format sličice za brzi pregled je JPEG. Ime sličice za brzi pregled mora biti *preview.jpg*.



Slika 10: Grafičko sučelje uređivača proizvoda

2.3.3 Demonstracijski primjer

U direktoriju s izvršnim datotekama nalaze se poddirektorije *axis2-1.5*, *ProductViewer.jar* i ProductViewer.bat. U primjeru se kao model proizvoda koristi mobitel čije su komponente raspoređene po tri poslužitelja. Prvi poslužitelj je zadužen za maske(gornji dio, donji dio i poklopac baterije), i upravljačke tipke, a kao vidljivi proizvod nudi mobitel. Drugi poslužitelj nudi model baterije, a treći poslužitelj kao vidljive proizvode koje se koriste u mobitelu nudi LCD displej, numeričku tipkovnicu. Njihovi repozitoriji se nalaze u direktoriju *axis2-1.5* kao poddirektoriji redom: *ProductViewer*, *ProductViewer1*, *ProductViewer2*.

Sve tri poslužitelja usluga se podižu na lokalnom računalu pokretanjem Axis2 poslužitelja. Java arhive sa SOAP uslugama su u direktoriju *axis2-1.5\repository\services* pod imenima *PVService.aar, PVService1.aar* i *PVService2.aar.*

Krajnje točke komunikacije su:

- PVService - *http://localhost:8080/axis2/services/PVService*
- PVService1 - *http://localhost:8080/axis2/services/PVService1*
- PVService2 - *http://localhost:8080/axis2/services/PVService2*

Poslužitelji demonstracijskog primjera se pokreću klikom na *start_servers.bat*. Ova pomoćna datoteka pokreće datoteku *axis2-1.5\bin\axis2server.bat* čime se pokreće *Axis2* poslužitelj koji pruža prije spomenute tri usluga.
Klijentska aplikacija se počinje izvršavati pokretanjem datoteke *ProductViewer.bat* ili *ProductViewer.jar* s krajnjom točkom komunikacije kao jedinim argumentom.

3 Zaključak

Predloženo rješenje interaktivnog grafičkog preglednika 3D objekata temeljenog na tehnologijama weba nudi prednosti pri modeliranju 3D modela proizvoda kao što su jeftinije i jednostavnije modeliranje, raspodjela zaduženosti po komponentama i mogućnost proširenja sustava. Proširenje sustava s većim brojem poslužitelja je moguće zbog toga što se mreža poslužitelja ponaša kao nestrukturirani *peer-to-peer* sustav. Svaki poslužitelj pozna samo određen broj njemu bitnih susjednih poslužitelja. Mogućnost proširenja sustava je jako ovisna o dinamičkim svojstvima sustava koja trenutno nisu analizirana. Analiza dinamičkih svojstava sustava bi podrazumijevala neki oblik simulacije ponašanja sustava. Simulacijom bi bilo zanimljivo vidjeti mijenjanje generiranog mrežnog prometa ovisno o broju poslužitelja, utjecaj raspoređivanja modela po poslužiteljima na opterećenje poslužitelja, te istražiti ponašanje poslužitelja pri velikom broju zahtjeva, odnosno sposobnost posluživanja većeg broja klijenata. Za očekivati je da bi se mogao pojaviti problem preopterećenih poslužitelja iz razloga što bi neki modeli bili puno češće traženi od drugih. Na primjeru preglednika proizvoda bi to bila neka standardna komponenta koja se koristi u velikom broju proizvoda.

Slaganje modela komponenata u model proizvoda je uz odgovarajući alat iznimno jednostavno. U tu svrhu bi bilo potrebno doraditi uređivač proizvoda koji je priložen programskom proizvodu i prilagoditi ga za jednostavniju upotrebu.

Važna je činjenica da preglednik od korisnika ne zahtjeva nikakvo posebno znanje. U isto vrijeme korisnik može obavljati kompleksne operacije nad elementima grafa scene koji odgovara 3D objektu.

Sažetak

Ovaj rad pokriva problematiku prikazivanja 3D objekata strukturiranih u graf scene, a čiji se čvorovi nalaze u raspodijeljenoj mreži poslužitelja opisani kao XML datoteke. Pokazano je da je moguće ostvariti odgovarajuće preslikavanje i pri tome dobiti određene prednosti pri oblikovanju 3D objekata. Programski proizvod kao konkretnu problemsku bazu ima trodimenzionalni prikaz modela proizvoda.

Pri oblikovanju programskog proizvoda posebna je pozornost posvećena na proširivost u pogledu dodavanja interaktivnih korisničkih operacija nad komponentama programskog proizvoda. Za iscrtavanje je korištena programska potpora Xith3d, a komunikacijska platforma je bila temeljena na SOAP protokolu. Primijećeni su i neki nedostaci sustavu koji ostaju otvoreni za daljnju analizu i rješavanje.

Ključne riječi: Xith3D, Axis2, SOAP, 3D objekt, 3D model, interaktivno, iscrtavanje, graf scene, OpenGL, Java, poslužitelj, klijent, raspodijeljeno.

Abstract

This paper covers area of rendering 3D objects witch are structured as a distributed scene graph. Nodes of scene graph are stored as XML files in a distributed network of servers. We have showed that it is possible to create adequate mapping of scene graph to distributed scene graph while at the same time achieving certain advantages during a 3D object modeling process. Produced software uses 3D product viewing as a base problem.

While designing software, extendability of interactive user operations on product components was brought to special attention. Xith3D was used as a scene graph renderer and communication was based on SOAP protocol. This paper addressed some of the system drawbacks that yet need to be analyzed and resolved.

Keywords: Xith3D, Axis2, SOAP, 3D object, 3D model, interactive, rendering, scene graph, OpenGL, Java, server, client, distributed.

Literatura

[1]Home of Xith3D Project, http://xith.org/, 16.april 2010.

[2]M. Fröhlich, A. Wenger, Xith3D in a Nutshell 2nd edition, 10. siječnja 2008.

[3]M. Fröhlich, Building up a SceneGraph in Xith3D,

[4]SOAP, 2. lipnja 2010, *SOAP (protocol),* http://en.wikipedia.org/wiki/SOAP_(protocol), 28.svibnja 2010.

[5]Web service, 2. lipnja 2010 , Web services, http://en.wikipedia.org/wiki/Web_service, 28. svibnja 2010

[6]W3C Working Group Note 11 February 2004, Web Services Architecture, 1.4 What is a Web service?, http://www.w3.org/TR/ws-arch/#whatis, 20. aprila 2010.

[7]S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D. F. Ferguson, Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More, Chapter 4. SOAP, Prentice Hall PTR, 22. ožujka 2005

[8]N. Quaine, SOAP Basics, 3. SOAP Messages, http://www.soapuser.com/basics3.html, 23 aprila 2010.

[9]Michael Galpin, Sending Attachments with SOAP, 1. listopada 2007., *Sending Attachments with SOAP*, http://www.theserverside.com/news/1363957/Sending-Attachments-with-SOAP , 26. aprila 2010.

[10]Sample WSDL document, *Using RDF for description and modeling in Web services*, http://www.w3.org/2001/04/wsws-proceedings/uche/wsdl.html, 26. aprila 2010.

[11]Welcome to Apache Axis2/Java, *Apache Axis2/Java - Next Generation Web Services* , http://ws.apache.org/axis2/, 27 aprila 2010.