Univerza
v Ljubljani
Fakulteta
za gradbeništvo
in geodezijo

Jamova 2
1000 Ljubljana, Slovenija
telefon (01) 47 68 500
faks (01) 42 50 681
fgg@fgg.uni-lj.si

Visokošolski program Geodezija,
Smer za prostorsko informatiko

Kandidat:
Aleš Anželak

Interaktivna spletna karta kanalizacije

Diplomska naloga št.: 287

Mentor:
viš. pred. mag. Samo Drobne

Ljubljana, 18. 12. 2008

STRAN ZA POPRAVKE, ERRATA

Stran z napako	Vrstica z napako	Namesto	Naj bo

IZJAVA O AVTORSTVU

Podpisani **ALEŠ ANŽELAK** izjavljam, da sem avtor diplomske naloge z naslovom: »**INTERAKTIVNA SPLETNA KARTA KABELSKE KANALIZACIJE**«. Izjavljam, da prenašam vse materialne avtorske pravice v zvezi z diplomsko nalogo na UL, Fakulteto za gradbeništvo in geodezijo.

Ljubljana,_____

IZJAVE O PREGLEDU NALOGE

Nalogo so si ogledali učitelji prostorske smeri:

BIBLIOGRAFSKO-DOKUMENTACIJSKA STRAN IN IZVLEČEK

UDK: 004.42/.43:004.783.5:528.9:625.78(043.2)
Avtor: Aleš Anželak
Mentor: viš. pred. mag. Samo Drobne
Naslov: Interaktivna spletna karta kabelske kanalizacije
Obseg in oprema: 67 str., 2 pregl., 29 sl..
Ključne besede: interaktivna spletna karta, XML, SVG, XSLT, XPATH, XQUERY

Izvleček

Diplomsko delo obravnava izdelavo interaktivne spletne karte na primeru XML dokumenta kabelske kanalizacije z uporabo transformacijskega jezika XSLT. S pomočjo XQUERY peskovnika je prikazana možnost poizvedovanja, izvoza in spreminjanja atributov v XML dokumentu kabelske kanalizacije.

BIBLIOGRAPHIC-DOCUMENTALISTIC INFORMATION

UDC: 004.42/.43:004.783.5:528.9:625.78(043.2)
Author: Aleš Anželak
Supervisor: Sen. Lect. Samo Drobne, M.Sc
Title: interactive web map, XML SVG, XSLT, XPATH, XQUERY
Notes: 68 p., 2 tab., 29 pic.
Key words: interactive web map, XML, SVG, XSLT, XPATH, XQUERY

Abstract:

The present work focuses on the making of interactive web map using XSLT language from XML document which contains data of canalization network. With use of XQuery Sandbox is presented ability to query, export and change the data inside the XML document.

Anželak, A. 2008. Interaktivna spletna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

ZAHVALA

Zahvaljujem se mentorju viš. pred. mag. Samotu Drobnetu za pomoč in nasvete pri izdelavi diplomske naloge.

Posebna zahvala velja staršem, ki so mi omogočili študij, ter vsem profesorjem in delavcem Fakultete za gradbeništvo in geodezijo, Univerze v Ljubljani, za posredovano potrebno znanje.

KAZALO VSEBINE

Anželak, A. 2008. Interaktivna spletna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

KAZALO PREGLEDNIC

 Anželak, A. 2008. Interaktivna spletna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

KAZALO SLIK

XVIII Anželak, A. 2008. Interaktivna spletna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

OKRAJŠAVE IN ZNAKI

ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
CDATA	Character data
CSS	Cascading Style Sheets
DHTML	Dynamic HTML
DOF	Digitalni orto-foto 1:5000
DOM	Document Object Model
DTD	Document Type Definition
ECMA	European Computer Manufacturers Association
EHIŠ	Evidenca hišnih številk
FLWR	FOR, LET, WHERE, RETURN
GIS	Geografski informacijski sistem
GML*	Generalized Markup Language
GML*	Geography Markup Language
HTML	Hyper Text Markup Language
IBM	International Business Machines Corporation
ITD	Izdelava Tehnične Dokumentacije
OQL	Object Query Language
PDF	Portable Document Format
RTF	Rich Text Format
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SMIL	Sychronized Multimedia Integration Language
SQL	Structured Query Language
SVG	Scalable Vector Graphics
SVGZ	Compressed Scalable Vector Graphics File
URI	Universal Resourse Identifier
UTF-16	16-bit Unicode Transformation Format
UTF-8	8-bit UCS/Unicode Transformation Format

W3C	World Wide Web Consortium
WML	Wireless Markup Language
Xlink	XML Linking Language
XML	eXstensible Markup Language
XML-QL	A Query Language for XML
Xpath	XML (eXstensible Markup Language) Path Language
Xpointer	XML Pointer Language
XQL	XML Query Language
Xquery	XML (eXstensible Markup Language) Query Language
XqueryX	XML Syntax for XQuery
XSD	XML Schema Datatypes
XSL	eXstensible Stylesheet Language
XSL-FO	XSL (eXstensible Stylesheet Language) Formatting Objects
XSLT	eXstensible Stylesheet Language Transformations
YATL	Yet Another Transformation Language

* Okrajšava GML se uporablja v obeh primerih.

1 UVOD

Svetovni splet je najnovejši medij, ki omogoča prikaz in širjenje prostorskih podatkov. Toda šele v devetdesetih smo bili priča hitremu razvoju interneta. V razmeroma kratkem času je postal nepogrešljiv del našega življenja – še več: brez internetne povezave si danes težko predstavljamo moderne komunikacije ter prenos podatkov na daljavo.

Standardizirani podatki so lahko temelj reševanja določenega sklopa problemov. Pogosto so najbolj stabilen del naloge. Standard XML (eXstensible Markup Language) je sprožil reševanje mnogih nalog, ki so bile pretežke ali predrage za obravnavo z računalniki.

Pred standardom XML je vsak program po svoje shranjeval podatke. Če so bili ti podatki namenjeni ljudem, težav praviloma ni bilo, ker so človeški možgani izredno prilagodljivi. Kadar pa je treba rezultate enega programa prebrati kot podatke v drugem programu drugega razvijalca, lahko pride do problemov.

Standard XML se zato največ uporablja za standarden prenos podatkov med računalniškimi programi. Standardni zapis podatkov in standardna orodja za obdelavo podatkov lahko te postopke zelo olajšata.

Ena izmed težav, ki jih uspešno rešuje standard XML, je tudi podvajanje zapisa podatkov. Pogosto je potrebno iste podatke predstaviti na različne načine. Zato jih v računalniku shranjujemo v več različicah. Standard XML omogoča zapis podatkov v nevtralni obliki brez predstavitvenih elementov, ki jo lahko XML procesorji na standarden način predelajo v druge oblike.

Pred začetkom rabe XML standarda so vektorske tehnologije prikazovale prednosti v upodabljanju vektorskih kart na strani uporabnika v primerjavi z razširjeno kartografsko spletno infrastrukturo, ki je generirala karto na strežniku v obliki slikovne datoteke na zahtevo uporabnika.

SVG (Scalable Vector Graphics) je standardni zapis za opisovanje vektorske in delno tudi rastrske grafike, ki temelji na jeziku XML. Vključuje zapis treh različnih tipov objektov (vektorske grafične objekte, sestavljene iz linij ali krivulj, rastrske slike in besedila) in omogoča interaktivnost, skriptiranje, animacijo in posebne učinke. Kmalu po nastanku je SVG postal zanimiv tudi za posredovanje prostorskih podatkov, saj je imel precej prednosti pred večino obstoječih načinov za posredovanje prostorskih podatkov na internetu.

Kabelska kanalizacija je povezan sistem cevi, ki jih zaključujejo kabelski jaški. Trasa kabelske kanalizacije predstavlja potek cevi med trasnimi objekti. Sestavljena je iz trasnih odsekov, ki so na obeh straneh zaključeni s trasnim objektom.

V diplomski nalogi je opisan standard XML in nekatere z njim povezane tehnologije. Na primeru XML dokumenta kabelske kanalizacije, ki predstavlja del izvršilne tehnične dokumentacije kabelske kanalizacije, je z uporabo stilnega jezika XSLT (eXstensible Stylesheet Language Transformations) podana možnost upodobitve prostorskih podatkov zapisanih v XML standardu s SVG zapisom.

Interaktivno spletno karto poganja spletni strežnik eXist-db, ki je v celoti zgrajen na XML tehnologiji. S pomočjo XQuery (XML Query Language) peskovnika je prikazana možnost poizvedovanja, izvoza in spreminjanja atributov v XML datoteki. Spletna karta ni javno dostopna in je zgolj rezultat praktičnega dela v okviru te diplomske naloge.

2 STANDARD XML

XML (eXstensible Markup Language) je svetovni podatkovni standard in spada med najuspešnejše standarde na področju računalništva in informatike. Je temelj učinkovitega razvoja novih programov in uspešne obdelave podatkov. Predstavlja pomemben mejnik pri reševanju nalog z računalniki. Določa pravila za zapis hierarhično urejenih označenih podatkov v obliki besedila. XML je samo standardni označevalni jezik za zapis podatkov.

Standard XML se je uveljavil praktično na vseh področjih uporabe računalnikov. Omogoča zanesljiv zapis kakršnihkoli elektronskih dokumentov. Je odlično orodje za izdelavo podatkovnih slovarjev. Z njegovo pomočjo lahko hitro in zanesljivo definiramo svoj podatkovni jezik. Standard XML uvrščamo zaradi te lastnosti tudi med meta jezike. Pomembna so široko dostopna splošna programska orodja za procesiranje XML dokumentov. Mnogi XML jeziki so bili razviti na že uveljavljenih področjih (Kovačič, 2005)

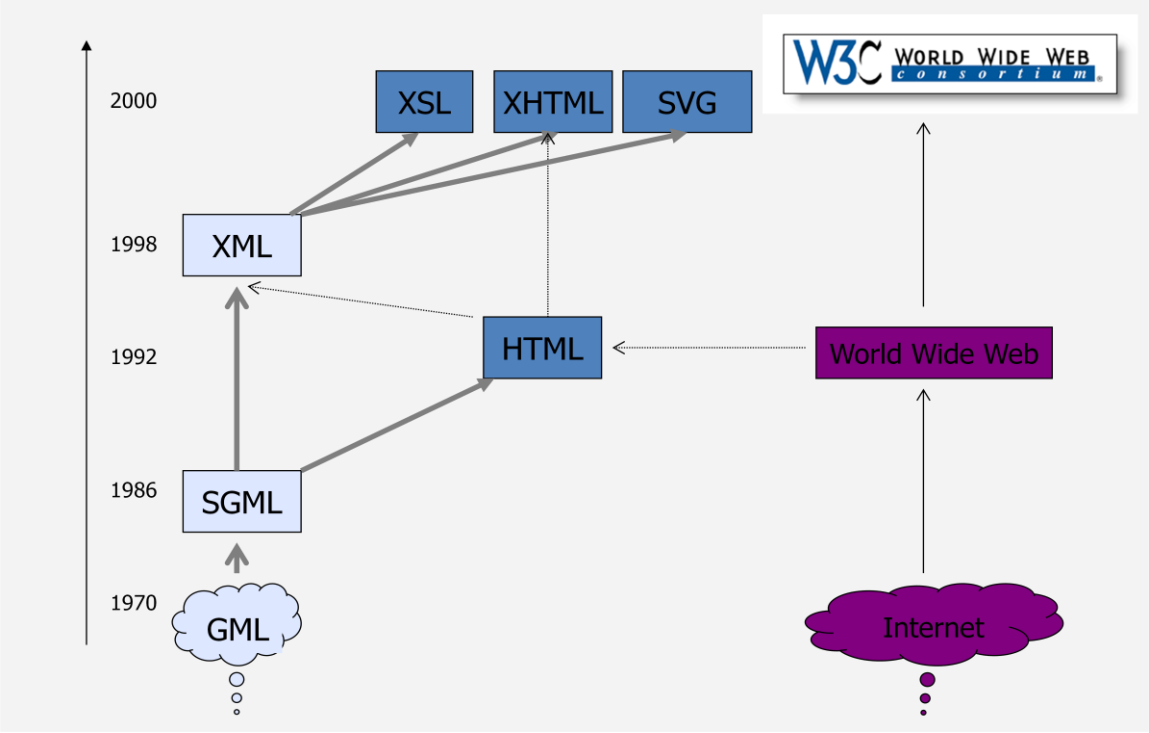
XML se je kot temeljni podatkovni standard uveljavil pozno zaradi premalo zmogljivih računalnikov, ki pred letom 2000 niso bili sposobni dovolj učinkovito procesirati besedil. Računalniki še danes bistveno hitreje procesirajo števila zapisana v binarni obliki. Večina programov starejše generacije podatke trajno zapisuje v binarni obliki. Binarnih podatkov pa se ne da učinkovito standardizirati. XML je zelo primeren izmenjevalni format, uporablja se za prenos podatkov med programi na istem računalniku ali med računalniki povezanimi v omrežje.

XML je samo standardni označevalni jezik za zapis podatkov. XML (Kovačič, 2005):

- ni programski jezik,
- ni podatkovna baza in
- ni omrežni transportni protokol.

2.1 Ozadje in razvoj standarda XML

Korenine XML standarda segajo štiri desetletja v preteklost, v dobo osrednjih računalnikov, ko je IBM (International Business Machines Corporation) razvijal način za strukturiranje dokumentov. Cilj je bil pospešiti izmenjavo in obdelavo podatkov. Rezultat teh naporov je bil GML (Generalized Markup Language). Medtem, ko je bil GML zgolj v interni uporabi IBM-a, drugje ni bil sprejet. Druge organizacije so sicer razvile podobne jezike za strukturiranje dokumentov, vendar so bili le-ti nezdružljivi med sabo.



Slika 1 Razvoj XML standarda

2.1.1 SGML

SGML (Standard Generalized Markup Language) predstavlja prvi uspeh pri razvoju standardiziranih označevalnih jezikov. Kot standard je bil sprejet že leta 1986 in ima bogato zgodovino. Je semantični in strukturni označevalni jezik za besedilne dokumente. XML je bil

prvotno zamišljen kot okrnjena, poenostavljena različica standarda SGML in mnogo rešitev iz SGML je bilo prenesenih v XML. SGML je izjemno zmogljiv in zato tudi kompleksen. Uspešen je bil le pri obsežnih in zapletenih dokumentih. Široko se ni nikoli uveljavil. Jezik HTML (Hyper Text Markup Language) spada med SGML aplikacije.

2.1.2 HTML

HTML je označevalni jezik, s katerim opisujemo oblikovne lastnosti in strukturo spletnih strani. Zgleduje se po splošnem in zelo kompleksnem jeziku SGML. S pomočjo jezika HTML lahko ustvarimo hiperbesedilo. To je besedilo z izpostavljenimi besedami, ki se navezujejo na novo vsebino poljubnega formata, npr. na drugo hiperbesedilo, sliko, zvok, video sekvenco.

Jezik HTML ne le, da omogoča medsebojno povezavo dokumentov, ampak tudi predpisuje oznake za način prikaza dokumentov. S HTML standardom je bil rojen svetovni splet, sestavljen iz mreže med seboj povezanih HTML dokumentov. Prvi uporabniki spleta so bili znanstveniki, ki jih je bolj zanimala vsebina kot pa predstavitev. Zato je imel HTML na začetku skromne zmožnosti oblikovanja.

Kasneje je bil HTML razširjen s standardom CSS (Cascading Style Sheets), ki omogoča ločevanje vsebine od oblike, vendar popolna ločitev ni možna. Prednosti ločitve so lažje vzdrževanje in spreminjanje, bolj konsistentno oblikovanje in enostavnejše prikazovanje na različnih platformah in medijih.

Navkljub ogromnemu uspehu, pa ima HTML nekatere pomembne omejitve. Na začetku je več kot zadoščal, vendar pa so s širitvijo spleta oblikovalci spletnih strani naleteli na omejitve. Naloge, za katere HTML ni bil nikoli namenjen; na primer animacije, dostop do zbirk podatkov in interaktivnost, so programerje pripeljale do roba zmožnosti HTML. S pomočjo nestandardnih dodatkov HTML in pomožnih tehnologij so ustvarili spletne strani, ki jih lahko vidimo danes.

Sčasoma je postalo jasno, da so meje zmogljivosti HTML presežene. Ena izmed bistvenih omejitev je določen nabor oznak in nezmožnost ustvarjanja novih, da bi se zadostilo novim potrebam. Z drugimi besedami, HTML je nerazširljiv. Druga omejitev je prepletenost oznak za strukturo z oznakami za prikaz. Zato sta struktura in prikaz neločljivo povezana.

Iskanje rešitve je prevzela skupina znotraj W3C. Začel se je razvoj standarda, ki bo enostaven kot HTML in bo obenem temeljil na jeziku SGML. Vodila razvijalcev novega standarda za predstavitev podatkov so bila ustvariti standardni jezik za označevanje dokumentov, ki bo podpiral širok spekter aplikacij in bo primeren zlasti za široko uporabo na internetu. Ker je nastajanje standardov javen in odprt proces, je le-ta neizogibno počasen. Za objavo XML priporočil je W3C (World Wide Web Consortium) od začetka razvoja do objave potreboval dve leti, preden jih je kot verzijo 1.0 izdal leta 1998.

Konzorcij svetovnega spleta, običajno poznan kot W3C, je javna neprofitna organizacija za standardizacijo, zato tudi plačevanje licenčnih pravic za uporabo standardov ni potrebno. Standardi, ki jih razvije so splošno dostopni. Aktivnosti organizacije so namenjene usmerjanju razvoja svetovnega spleta z razvojem splošno uporabnih in nadgradljivih podatkov.

2.2 Cilji pri razvoju XML standarda

Cilji, ki so jih upoštevali pri razvoju XML jezika so (W3C, 2008):

- XML naj bo enostaven za uporabo preko interneta;
- XML naj podpira širok nabor programskih aplikacij;
- XML naj bo skladen s SGML;
- pisanje programov, ki procesirajo XML dokumente, naj bo enostavno;
- število pomožnih dodatkov v XML naj bo čim manjše, idealno bi bilo brez njih;
- XML dokumenti naj bi bili pregledni in čitljivi;
- načrt XML naj se da hitro popraviti;

- načrt XML naj bo formalen in jedrnat;
- XML dokumenti naj bodo preprosti za izdelavo;
- natančnost izboljšave XML je minimalnega pomena.

2.3 Prednosti XML standarda

Specifikacija jezika XML določa njegovo namembnost in iz nje so razvidne prednosti in pomen predstavitve podatkov v jeziku XML (Sturm, 2000):

- Razširljivost; Možnost definiranja novih slovarjev glede na potrebe konkretne aplikacije, panoge ali načina komuniciranja.
- Izmenjava podatkov in komunikacija; Izmenjava podatkov je možna med heterogenimi sistemi.
- Neodvisnost podatkov; Dokument XML je neobčutljiv za nadgradnjo programske in strojne opreme. Predstavite v XML-u je neodvisna od programske in strojne opreme.
- Večja uporabnost podatkov; Podatke v XML-u lahko uporabi katerakoli programska rešitev, ki podpira jezik XML.
- Internacionalizacija; XML je zasnovan na 16-bitni kodi Unicode sistema, HTML in SGML pa temeljita na ASCII (American Standard Code for Information Interchange) standardu. Zatorej lahko z XML-om uporabljamo tudi tuje slovnice, ne pa samo angleško, ki je osnova pri ASCII standardu.
- Podpora razvoju; Obstaja veliko dostopnega gradiva za izobraževanje uporabnikov, predvsem na svetovnem spletu, zato imajo razvijalci sistema zelo dobro podporo za črpanje znanja.
- Ločitev podatkov od njihove predstavitve; Iste podatke je možno predstaviti na več načinov. Način predstavitve podatkov lahko spreminjamo neodvisno od sistema, ki ga uporabljamo.

- Enostavna uporaba; Zaradi enostavne sintakse je jezik XML primeren z jezikom HTML in iz dneva v dan ga uporablja več ljudi. Posledično z večjim številom uporabnikov se na trgu veča tudi izbira programov, ki podpirajo XML.

2.2 Spletna kartografija z uporabo XML standarda

Spletne karte postajajo interaktivni prikaz tematskih prostorskih podatkov iz porazdeljenih informacijskih sistemov namenjene določeni skupini uporabnikov in rabi. Ta revolucionarna sprememba v internetni kartografiji se je pojavila zaradi napredka v porazdeljenem računalništvu in predstavitvi XML jezika in povezanih tehnologij.

XML tehnologija je močno vplivala na množico pristopov in orodij, uporabljenih na svetovnem spletu. XML standard se uporablja na vseh področjih interneta, od enostavnih spletnih strani do kompleksnih spletnih aplikacij. Jedrnata in kratka specifikacija XML jezika na nivoju priporočila iz leta 1998 je vsebovala samo 26 strani, kar je bila bistvena razlika v primerjavi z njegovimi predhodniki. Navsezadnje je bil namen XML jezika povzeti nabor novih pristopov v računalništvu vključno z novimi vzorci za izmenjavo, predstavitev in obdelavo podatkov preko interneta. Glavni namen XML tehnologije je vzpostaviti spletne podatke in storitve kot »vklopi in poženi« komponente, ki se lahko kličejo in kombinirajo na zahtevo (Pavlicko, 2002).

Medtem ko se moderna spletna kartografija razvija od prikazovanja statičnih slikovnih kart do prilagodljivih, dinamičnih in interaktivnih kart, ki vključujejo tudi multimedijske vsebine iz različnih porazdeljenih virov, je bilo nujno omogočiti medopravilnost med kartografskimi podatki in storitvami. Potreba po odprtih in od platforme neodvisnih standardih za potrebe spletne kartografije je pripeljala kartografe in ponudnike spletnih kartografskih orodij do uporabe XML tehnologij. XML standard je vplival na spletno kartografijo v naslednjih pomembnih smereh (Zaslavsky, 2003):

- prikaz digitalnih prostorskih metapodatkov podpira izdelavo različno oblikovanih predstavitev podatkov iz istega meta podatkovnega jedra, in omogoča avtomatizirano transformacijo podatkov za integrirano spletno kartiranje;
- določa univerzalen standard za opisovanje prostorskih pojavov, in je neodvisen od ponudnika podatkov in platforme;
- jeziki za prikaz 2D in 3D grafičnih vsebin, ki temeljijo na XML tehnologiji skrbijo za upodabljanje vsebin v grafično nezmogljivih brskalnikih;
- kartografske storitve, ki temeljijo na XML tehnologiji omogočajo enostavno vključitev v ostale standardne spletne aplikacije;

Ob jeziku HTML, ki je bil osredotočen na obravnavo besedilnih dokumentov, so kartografi iskali nove alternativne pristope, ki bi podpirali objavo kart na internetu. XML tehnologije omogočajo medopravilnost med raznovrstno programsko opremo in podatki, in pomagajo združevati podatkovne sloje iz bistveno različnih domen v spletne kartografske produkte. V spletni kartografiji je to omogočeno s standardnim zapisom prostorskih pojavov, možnostjo ponovnega preoblikovanja in transformacije prostorskih vsebin in vektorsko upodobitvijo, ki temelji na XML standardu.

Vektorske tehnologije so že pred začetkom rabe XML standarda nakazovale prednosti v upodabljanju vektorskih kart na strani uporabnika v primerjavi z razširjeno kartografsko spletno infrastrukturo, ki je generirala karto na strežniku v obliki slikovne datoteke na zahtevo uporabnika.

V standardnih spletnih brskalnikih je bila upodobitev vektorske grafike možna z uporabo namenskih Java klientov, ActiveX kontrolnikov ali vtičnikov. Z uporabo vektorskih jezikov za upodabljanje grafike, ki temeljijo na XML standardu, je postalo upodabljanje vektorske grafike na stani uporabnika enostavnejše. Upodabljanje vektorske karte na strani uporabnika poveča interakcijo uporabnika z uporabo zaslonskih namigov in poudarjanja določenih objektov, možnosti dogodkov, ki jih sproži uporabnik z miško ali tipkovnico, dinamični prikaz simbolov, kot tudi hitrejšo posodobitev prikaza. Izdelava uporabniške programske opreme za pregledovanje kart je postala relativno enostavno z uporabo skriptnih jezikov.

Uporabnik potrebuje za prikaz kart v najslabšem primeru pregledovalnik vektorskih podatkov, ki omogoča upodabljanje podatkov iz različnih podatkovnih zbirk ali spletnih storitev.

3 XML DOKUMENT

3.1 Sestavni deli XML dokumenta

XML (eXstensible Markup Language) dokument je besedilo, ki je logično lahko sestavljeno iz dveh delov: prologa in vsebine. Prolog je neobvezen. Vsebino pa določa obvezni korenski element, to je izhodiščni XML element dokumenta.

```
<?xml version="1.0" encoding="windows-1250"?>
<TKOmrezje OE="NG" FL="6001_PSX NOVA GORICA" Kabel-kanal="NG" Vrsta-dok="DOKUMENTACIJA KANALIZACIJE" Firma-izv="GEOM" Izdelal="MALNARIČ" Izmeril="STERLE" Datum-kreiranja="22.02.08" Skrbnik-email="" Komentar="" Zap-stev-sh="" PID="600111743"
        Projekt="ITD-NG (600111743)" Rok-projekta="22.05.2008" Izvorni-TKI-ID="" Pozic-izv-TKI-papir="20|150" Zacet-smer-izr="" Datum-izvoza="22.02.2008" TRSDelNaziv="" Kabel="NG" Del-kabla="0" Kabel-smer="">
</TKOmrezje>
```

Slika 2 Deklaracija in korenski element datoteke A_6001_NG_TKPIS.xml

3.1.1 Prolog XML dokumenta

Prolog ima lahko naslednje sestavne dele (W3C, 2008):

- Na začetku sme biti XML deklaracija, ki določa osnovne podatke o dokumentu.
- Neobvezna informacija v prologu določa tip (strukturo) dokumenta.
- Prolog sme vsebovati tudi procesne inštrukcije.

XML deklaracija je lahko zapisana samo v prologu XML dokumenta. Pred deklaracijo ne sme biti nobenih drugih znakov, tudi presledkov in komentarjev ne. Razčlenjevalniku pove verzijo v dokumentu upoštevanega XML standarda in uporabljeni sistem kodiranja znakov. Če jo izpustimo, veljajo privzete vrednosti.

Različni razčlenjevalniki lahko prepoznajo različne kodirne sisteme. Obvezani so prepoznati UTF-16 (16-bit Unicode Transformation Format) in UTF-8 (8-bit UCS/Unicode Transformation Format).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" [
    <!ATTLIST svg
  xmlns:attrib CDATA #IMPLIED
  xmlns:batik CDATA #IMPLIED
>
<!ATTLIST g
  batik:static CDATA #IMPLIED
>
<!ATTLIST image
  batik:static CDATA #IMPLIED
>
<!ATTLIST path
  batik:static CDATA #IMPLIED
>
]>
<?AdobeSVGViewer save="snapshot"?>
```

Slika 3 Prolog datoteke index.svg

Uporaba atributa encoding in standalone v XML deklaraciji ni obvezna. Opcijski atribut standalone XML procesorju pove, naj uporablja zunanji DTD (Data Type Definition), ki ni vključen v XML dokument.

```
<?xml version
        opt._encoding
        opt._standalone?>
```

Slika 4 Splošen format XML deklaracije

```
<?xml version="1.0" encoding="windows-1250"?>
```

Slika 5 XML deklaracija, ki prepozna šumnike in sičnike

Procesna inštrukcija je v XML dokument vgrajeno navodilo razčlenjevalnikom, kako naj obdelujejo XML dokument. XML procesor ni obvezan upoštevati procesnih inštrukcij.

```
<?ime-inštrukcije podatki-inštrukcije ?>
```

Slika 6 Splošen format procesne inštrukcije

```
<?AdobeSVGViewer save="snapshot"?>
```

Slika 7 Procesna inštrukcija za SVG pregledovalnik podjetja Adobe

Prolog XML dokumenta lahko vsebuje informacijo o tipu dokumenta. Tip dokumenta so slovnična in sintaktična pravila, ki podrobno določajo, kako mora biti sestavljen veljaven XML dokument. Če dokument ne vsebuje informacije o tipu dokumenta, veljavnost ni določena.

3.1.2 Vsebina XML dokumenta

Vsebina XML dokumenta je drevesasta struktura XML elementov. Na začetku mora biti en sam XML element (korenski element), ki sme vsebovati druge XML elemente.

3.1.2.1 XML elementi

XML element ima praviloma začetno oznako, vsebino in končno oznako. Logično predstavlja XML element enostaven ali sestavljen označen podatek.

Začetna oznaka vsebuje ime v zašiljenih oklepajih. Ime praviloma označuje podatke, ki sledijo začetni oznaki. Presledki v oznaki niso dovoljeni. Male in velike črke v oznaki niso enakovredne. Končna oznaka se od začetne razlikuje le v znaku /, ki mora slediti začetnemu znaku <.

Vsebina XML elementa so podatki zapisani med začetno in končno oznako XML elementa. Za vsebino veljajo naslednja pravila (W3C, 2008):

- vsebina sme biti prazna (nič znakov ali presledki),

- vsebina je lahko besedilo (niz znakov, brez vstavljenih XML elementov),
- vsebina je lahko niz drugih XML elementov,
- lahko pa je mešanica besedila in drugih XML elementov.

XML elemente brez vsebine so prazni XML elementi. Lahko jih zapišemo na dva načina: v običajni obliki z začetno in končno oznako in v okrajšani obliki (s posebnim zapisom začetne oznake, ki se konča z />).

3.1.2.1.1 Atributi v XML elementu

XML element sme imeti poleg vsebine med začetno in končno oznako tudi atribute v začetni oznaki. Atributi vsebujejo informacijo oziroma podatke. Vsak atribut ima dva dela: ime atributa in pripadajočo vrednost. Med njima je znak enačaj (=). Vrednost atributa je dodaten podatek, ki poleg vsebine določa element. Za ime atributa veljajo enaka pravila kot za ime elementa.

Za atribute veljajo naslednja enostavna in intuitivna pravila (W3C, 2008):

- Atributi so vedno del začetne oznake določenega XML elementa.
- Število atributov v začetni oznaki ni posebej omejeno.
- Vrstni red atributov ni pomemben. Razčlenjevalnik ne sme delati razlike glede vrstnega reda atributov.
- Vsak atribut ima eno izmed oblik: ime = "vrednost" ali ime = 'vrednost'. Pred in za enačajem sme biti 0 ali več belih presledkov.
- Atribut mora biti enoličen v začetni oznaki, kjer obstaja. V isti začetni oznaki ne sme biti dveh atributov z istim imenom.
- Prvi atribut mora biti od imena oznake ločen z vsaj enim belim presledkom. Isto pravilo velja za ločilo med atributi istega XML elementa.
- Vrednost atributa je vedno nesestavljena. Ne sme vsebovati nobenih XML elementov.
- Atribut med dvojnimi narekovaji sme vsebovati enojne narekovaje in obratno.
- Vrednost atributa sme vsebovati eno ali več CDATA (Character data) sekcij.

```
<TRObjekt DatumSpr="" Dimenzija="0.000" FL-prip="6001_PSX NOVA GORICA" Natancnost="2" OE-prip="NG" OZNAKA="" OznakaSTR="TS" Poz_x="395302.933" Poz_y="91314.423" Poz_z="94.100" PrevObjectHandle="" Projekt="600111743" RELDist="0" RELSmer="0" Rotation="0" Simb_x="395302.933" Simb_y="91314.423" Simb_z="94.100" Status="1" Status-dok="" TROBJ-TDID="600110454_550F4" TROBJ-TKISID="600110454_550F4" TipSpremembe="0" VirPodloge="2" VisinaObjekta="0.000"></TRObjekt>
```

Slika 8 Okrajšan zapis XML elementa z atributi brez vsebine

3.1.2.1.2 CDATA sekcija

Kadar želimo zapisati vsebino elementov brez uporabe posebnih znakovnih entitet za znaka & in >, lahko uporabimo CDATA (Character Data) sekcijo. To je niz znakov, ki ima točno določen začetek (<![CDATA[) in konec (]]>). Vse kar je vmes, je vsebina CDATA sekcije, ki pa nikjer ne sme vsebovati niza znakov]]>.

```
<xsl:text disable-output-escaping="yes"><![CDATA[<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" [
    <!ATTLIST svg
  xmlns:attrib CDATA #IMPLIED
  xmlns:batik CDATA #IMPLIED
>
<!ATTLIST g
  batik:static CDATA #IMPLIED
>
<!ATTLIST image
  batik:static CDATA #IMPLIED
>
<!ATTLIST path
  batik:static CDATA #IMPLIED
>
]>
]]></xsl:text>
```

Slika 9 CDATA sekcija dokumenta xml2svg.xslt

3.1.2.2 Komentar

Komentar je informacija, ki je razčlenjevalniki ne upoštevajo. Dodatno pojasnjuje določene dele XML dokumentov in je praviloma namenjen razvijalcem. Pravilo za zapis komentarjev je preprosto: komentar se začne z <!-- in konča z -->. V komentarjih so lahko poljubni znaki, tudi & in >, ne sme vsebovati niza znakov-->.

3.2 Uporaba imenskih prostorov

XML dokument lahko vsebuje veliko podatkov. Pogosto ga sestavimo iz drugih XML dokumentov, kar lahko povzroči naslednjo težavo: isto ime XML elementa lahko v različnem delu XML dokumenta predstavlja različne podatke z različno strukturo. Posledica so težave pri procesiranju takega dokumenta.

Problem rešujejo imenski prostori. Ime XML elementa lahko uvrstimo v imenski prostor. Pravimo, da ime kvalificiramo z imenskim prostorom. Preprosto to pomeni, da imenu predpišemo, kateremu imenskemu prostoru pripada. Kvalificirano ime XML elementa je sestavljeno iz dveh delov, ki sta ločena z dvopičjem: iz oznake imenskega prostora in iz imena elementa. Imena XML elementov smejo biti kvalificirana, lahko pa tudi niso.

Imenski prostor v standardu XML ima globalno enolično oznako (ime), ki jo določa URI (Universal Resourse Identifier) referenca. Imenski prostor v začetni oznaki XML elementa določa atribut xmlns: xmlns="URI". Njegov doseg je v tem elementu in v vseh vsebovanih XML elementih, vendar njegova uporaba ni obvezna. Isti imenski prostor lahko uporabimo v različnih XML elementih dokumenta (Kovačič, 2005).

Ker je URI referenca pogosto razmeroma dolg niz znakov, jo lahko preslikamo v kratko predpono dolžine nekaj znakov (1-3). Ime predpone je lahko poljubno XML ime. Zapišemo xmlns:predpona = "URI". Predpono potem uporabimo za kvalifikacijo imen XML elementov.

```
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"/>
```

Slika 10 Kvalificirano ime elementa s predpono

Privzeti imenski prostor določimo z atributom xmlns brez predpone. Privzeti imenski prostor velja za element, kjer je definiran (če ni kvalificiran) in za vse vsebovane XML elemente, ki niso kvalificirani. V istem XML dokumentu smemo uporabiti več imenskih prostorov. Vsak element sme imeti svoj privzeti imenski prostor.

V spodnjem primeru SVG (Scalable Vector Graphics) dokumenta vsi elementi pripadajo privzetemu imenskemu prostoru, ki ga določa URI: http://www.w3.org/2000/svg.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">
  <defs>
    <symbol id="SimbolObjekt">
      <circle cx="0" cy="0" r="0.2"></circle>
    </symbol>
  </defs>
</svg>
```

Slika 11 SVG dokument s privzetim imenskim prostorom

3.3 Slovnična pravilnost in veljavnost

XML dokumenti morajo biti napisani po določenih slovničnih pravilih. Slovnično pravilnost dokumenta preverja razčlenjevalnik. Če dokument vsebuje slovnične napake, nam razčlenjevalnik javi sporočilo o napakah. Dokumenti, ki ne upoštevajo teh pravil, niso slovnično pravilni in zato tudi niso veljavni.

Pravilno oblikovan XML dokument ustreza vsem slovničnim pravilom standarda XML, ki veljajo obvezno za vse XML jezike. Če XML dokument ni pravilno oblikovan, sploh ni XML dokument in ga nobeden razčlenjevalnik ne sme obdelati.

Veljavni XML dokumenti morajo biti pravilno oblikovani ter hkrati izpolnjevati predpisano sestavo in zgradbo. Upoštevajo slovnico jezika, ki je izrecno predpisana, in vsebujejo samo oznake in atribute, ki so določeni v slovnici jezika. Veljavnost dokumenta izrecno preverja ustrezen poseben razčlenjevalni program.

3.4 XML dokument kabelske kanalizacije

Aplikacijski modul XML izvoz programskega orodja ITD (Izdelava Tehnične Dokumentacije, Telekom Slovenije, d.d.) omogoča izvoz kanalizacije (trasnih objektov in trasnih segmentov) v XML dokument. Modul v ozadju vedno najprej požene aplikacijski modul kontrola simbologije. Če je kontrola simbologije brez napak, modul tvori pravilno oblikovan XML dokument.

Aplikacijski modul kontrola simbologije omogoča kontrolo skladnosti izdelane tehnične dokumentacije s predpisanimi pravili. Rezultat kontrole je poročilo o napakah, ki je osnova za odpravljanje napak, oziroma potrdilo, da je dokumentacija pravilno izdelana.

```
<ITDTest>
  <Info>kontrola trasnih objektov</Info>
  <Info>kontrola trasne topologije</Info>
  <Info>kontrola koordinat</Info>
  <Info>kontrola atributov</Info>   <Info>Datum projekta:22.02.08</Info>
  <Info>Datum kontrole:17.03.2008</Info>
  <Info>OE:NG</Info>
  <Info>FL:6001_PSX NOVA GORICA</Info>
  <Info>Število trasnih segmentov: 71, skupna dolžina: 1913.47 metrov.</Info>
  <Info>Število trasnih objektov: 72</Info>
  <Result>Risba je v skladu s simblogijo ITD(0)</Result>
</ITDTest>
```

Slika 12 Potrdilo o skladnosti XML dokumenta dokumentacije

V tej diplomski nalogi je bil v SVG zapis transformiran XML dokument kabelske kanalizacije A_6001_NG_TKPIS.xml, ki vsebuje prostorske podatke o kabelski kanalizacije na izbranem območju v Novi Gorici.

4 DOLOČANJE BESEDNJAKOV

Razširljivost omogoča svobodo pri oblikovanju standardov za izmenjavo podatkov. Zato so potrebni mehanizmi, ki zagotavljajo ustrezno strukturo datoteke. Nastalo je mnogo XML (eXstensible Markup Language) jezikov, ki so definirani bodisi v obliki DTD (Data Type Definition) bodisi z XML Schema.

Obstajata dve vrsti XML razčlenjevalnikov: eni preverjajo veljavnost dokumenta, drugi pa ne. Preverjanje veljavnosti je lahko zamudno, zato se običajno preverjajo XML dokumenti tujega izvora. Če je XML dokument ustvaril preverjen računalniški program, njegove veljavnost ob vsakokratnem procesiranju dokumenta ni potrebno preverjati.

4.1 Definicija tipa dokumenta

Standard DTD (Data Type Definition) obstaja že precej časa, saj je nastal v času predhodnika XML, jezika SGML (Standard Generalized Markup Language). V resnici je standard DTD odgovoren za nastanek najbolj razširjenega opisnega jezika na svetu, jezika HTML (Hyper Text Markup Language). HTML je namreč dokumentni tip, ki je v osnovi nastal z definicijo sheme DTD nad opisnim jezikom SGML (Standard Generalized Markup Language) (Kovačič, 2005).

Dokument DTD določa pravila, po katerih so strukturirani podatki v XML dokumentu. Določiti je treba, katere elemente vsebuje XML dokument, v kakšnem vrstnem redu si sledijo, katere atribute vsebujejo in kakšna je dovoljena vsebina elementov in atributov. Dokument je veljaven, kadar ustreza pravilom, določenim v opisu tipa dokumenta (Jurič, 2001).

Tip dokumenta je lahko podan na dva načina: dokument bodisi vsebuje sklicevanje na tip dokumenta bodisi je definicija tipa dokumenta vključena v sam dokument. Prvi način je običajnejši, ker je lahko tip dokumenta zapisan v datoteki samo enkrat.

Za deklaracijo tipa elementa velja naslednja sintaksa:

- Prazen element določimo z: <!ELEMENT ime-elementa EMPTY>.
- Element, ki vsebuje poljubno kombinacijo znakov in drugih elementov, predpišemo z: <!ELEMENT ime-elementa ANY>.
- Element, ki vsebuje samo niz znakov brez drugih elementov, predpišemo z:
- <!ELEMENT ime-elementa (#PCDATA)>.
- Element, ki vsebuje kombinacijo niza znakov in drugih elementov, podamo z:
- <!ELEMENT ime-elementa (#PCDATA|ime1|ime2|...)*>.
- Element, ki vsebuje druge elemente v predpisanem vrstnem redu, zahtevamo z: <!ELEMENT ime-elementa (ime1, ime2, ...)>.

Pri določanju sestavnih elementov smemo uporabiti tudi naslednje indikatorje števila pojavov:

- ? opcijski indikator (0 ali 1)
- - določa število pojavov elementa med 0 ali več
- + določa število pojavov elementa 1 ali več.

Za vsak element določimo vse pripadajoče atribute z ATTLIST deklaracijo, ki lahko vsebuje naštete kombinacije deklaracij posameznih atributov.

Določilo #REQUIRED zahteva obvezno prisotnost atributa v elementu. Določilo #IMPLIED prepušča razčlenjevalniku odločitev, kakšno vrednost uporabiti, če atribut v dokumentu manjka. Določilo #FIXED predpisuje vrednost atributa, ki se upošteva v odsotnosti atributa. Če ni niti določila #REQUIRED, niti določila #IMPLIED, potem se v primeru odsotnosti atributa uporabi privzeta vrednost.

Tip vrednosti je lahko bodisi CDATA ali v okroglih oklepajih naštete dovoljene vrednosti ločene z znakom |.

Atribut, ki ima lahko za vrednost poljuben niz znakov določimo z:
ime-atributa CDATA #REQUIRED.

Atribut z vnaprej določenimi naštevnimi vrednostmi:
dan (PON|TOR|SRE|ČET|PET|SOB|NED) #REQUIRED.

Spodnja slika prikazuje primer določitve besednjaka XML dokumentu z uporabo standarda DTD. Pravila, ki določajo veljavnost dokumenta, so zapisana v prologu dokumenta znotraj CDATA (Character Data) sekcije.

```
<?xml version="1.0" encoding="windows-1250"?><!DOCTYPE TKOmrezje [
<!ELEMENT TRObjekt EMPTY>
<!ATTLIST TRObjekt
   Poz_x CDATA #REQUIRED
   Poz_y CDATA #REQUIRED
   Poz_z CDATA #REQUIRED>
<!ELEMENT TRObjekti (TRObjekt)+>
<!ELEMENT SITGEOMETRY (Coor)+>
<!ELEMENT Coor (#PCDATA)>
<!ATTLIST Coor ID CDATA #REQUIRED>
<!ELEMENT SITGEOMETRYZ (Coor)+>
<!ATTLIST SITGEOMETRYZ VertNumber CDATA #REQUIRED>
<!ELEMENT TRSegment (SITGEOMETRY,SITGEOMETRYZ)>
<!ATTLIST TRSegment Dolžina CDATA #REQUIRED>
<!ELEMENT TRSegmenti (TRSegment)>
<!ELEMENT TKOmrezje (TRObjekti,TRSegmenti)>
<!ATTLIST TKOmrezje OE CDATA #REQUIRED>
]>
<TKOmrezje OE="">
  <TRObjekti>
    <TRObjekt Poz_x="394980.368" Poz_y="91399.208" Poz_z="93.150"></TRObjekt>
    <TRObjekt Poz_x="395561.518" Poz_y="91147.116" Poz_z="97.440"></TRObjekt>
  </TRObjekti>
  <TRSegmenti>
    <TRSegment Dolžina="0.49">
      <SITGEOMETRY>
        <Coor ID="0">394980.368</Coor>
        <Coor ID="1">91399.208</Coor>
        <Coor ID="2">394980.496</Coor>
        <Coor ID="3">91399.690</Coor>
        </SITGEOMETRY>
      <SITGEOMETRYZ VertNumber="3">
        <Coor ID="0">93.150</Coor>
        <Coor ID="1">93.150</Coor>
      </SITGEOMETRYZ>
    </TRSegment>
  </TRSegmenti>
</TKOmrezje>
```

Slika 13 Določitev veljavnosti z uporabo DTD

Prednosti uporabe DTD jezika so v večini primerov omejene na preverjanje veljavnosti XML dokumentov. Strukturo dokumenta XML je namreč mogoče z omenjeno shemo pregledati z uporabo večine razčlenjevalnikov.

Slabosti DTD jezika:

- Edini podatkovni tip, ki ga DTD podpira, je besedilo, koncepta števil v DTD ni.
- Določenih omejitev ni mogoče zapisati (dolžina podatka, predpisana oblika podatka).
- Števila ponovitev posameznega XML elementa/atributa ni modoče določiti.
- DTD ne uporablja imenskih prostorov za imena elementov.
- DTD sintaksa ne ustreza standardu XML.

4.2 XML Shema

Standard XML Schema določa v XML zapisan besednjak, ki opisuje vsebine dokumentov XML. Nastal je predvsem zato, da odpravi pomanjkljivosti DTD. Je obsežen in kompleksen standard, zato je nastal v dveh delih. Prvi del se ukvarja z opisom strukture dokumentov XML in načini za opis vsebinskih modelov posameznih elementov ter omejitev na atributih XML. Drugi del specifikacije definira tipski sistem XML in s tem določa način tipizacije dokumentov XML.

XML Schema temelji na drugačnih, mnogo bolj uporabnih konceptih kot DTD. Določa pravila za zapis XML sheme, ki vsebuje slovar deklaracij XML elementov in definicij tipov XML elementov. Vsak deklariran XML element je v shemi podrobno opredeljen. Koncept tipa elementa je podoben kot pri programskih jezikih. Tip ima elementa ime in lastnosti. Vsak XML element je določenega tipa. Tipe iz ene sheme smemo uporabiti v drugih (Kovačič, 2005).

XML tipi elementov so vgrajeni, enostavni in kompleksni. Slednji so lahko sestavljeni iz elementov ali atributov. Vgrajeni tipi se delijo na primitivne in izpeljane vgrajene podatkovne tipe.

XML Schemo uporabimo tako, da jo priredimo XML dokumentu. V shemi morajo biti deklarirani vsi uporabljeni elementi. Povezava je lahko eksplicitna s pomočjo v XML dokumentu določenih imenskih prostorov, ki določajo njeno lokacijo.

```
<?xml version="1.0"?>
  <TRSegment Dolžina="0.49" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation=" besednjak.xsd">
     <Coor ID="0">394980.368</Coor>
     <Coor ID="1">91399.208</Coor>
     <Coor ID="2">394980.496</Coor>
     <Coor ID="3">91399.690</Coor>
  </TRSegment>
```

Slika 14 Sklicevanje na zunanji zapis besednjaka

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
 <xs:element name="TRSegment">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="Coor"/>
   </xs:sequence>
   <xs:attribute name="Dolžina" use="required" type="xs:decimal"/>
  </xs:complexType>
 </xs:element>
 <xs:element name="Coor">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:decimal">
     <xs:attribute name="ID" use="required" type="xs:integer"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Slika 15 Zapis besednjaka z uporabo XML Scheme

5 VMESNIKA ZA PROCESIRANJE IN RAZČLENJEVANJE XML DOKUMENTOV

XML dokument vsebuje zgolj podatke, in sam po sebi še ni uporaben. Z načinom pregledovanja se opredeli način dostopa in obdelave podatkov, zapisanih v XML obliki. V praksi sta se uveljavila dva tipa razčlenjevalnikov: DOM (Document Object Model) in SAX (Simple API for XML).

5.1 DOM razčlenjevalnik

Objektni model dokumenta – DOM je programski vmesnik, ki se lahko uporablja za dostop do katerih koli komponent dokumenta XML, vključno z njegovimi elementi, atributi, navodili za obdelavo, komentarji in deklaracijami entitet. Programski vmesnik objektnega modela DOM omogoča aplikacijam, da gredo skozi drevo in urejajo njegova vozlišča. Vsako vozlišče je določeno kot posebna vrsta vozlišča, odvisno od oštevilčenih konstant predmetnega modela DOM v XML-ju, ki tudi določa veljavna nadrejena in podrejena vozlišča za vsako vrsto vozlišča. Pri večini dokumentov XML so najbolj pogosta vrsta vozlišč elementi, atributi in besedilo. Atributi zavzemajo posebno mesto v predmetnem modelu, ker se jih ne obravnava kot podrejena vozlišča nadrejenih, ampak kot lastnosti elementov. Objektni model dokumenta je standard, neodvisen od platforme in programskega jezika.

Temelji na drevesni strukturi XML dokumenta. Delovanje vmesnika DOM je enostavno in si sledi v naslednjih fazah:

1. Preberi dokument.
2. Razčleni dokument na posamezne enote (besedilo, komentar, element, atribut, navodilo za procesiranje).
3. V spominu strukturiraj logično drevo vozlišč iz točke 2.

Uporaba vmesnika DOM za izgradnjo in manipulacijo z XML dokumenti ima kar nekaj prednosti. Vmesnik DOM zagotavlja pravilno sintakso. Razvijalec se ukvarja z drevesno podobo dokumenta in ne več z besedilno predstavitvijo. Ločitev predstavitve od same sintakse se omogoči z drevesno strukturo, ki predstavlja logični pogled na vsebino – kakšne informacije so v dokumentu in na kakšen način so povezane med seboj. Transformacija XML dokumenta z uporabo DOM-a pomeni spreminjanje drevesne strukture, kar je mnogo bolj enostavno in hitrejše kot delo z datotekami. Prenos informacij iz hierarhičnih in relacijskih baz v XML-u z uporabo DOM-a je zelo enostaven, saj je način, predstavitve povezav med elementi, ki jih podpira DOM, zelo podoben predstavitvam v omenjenih bazah (Anderson s sod., 2000).

DOM prebere celotno vsebino dokumenta XML v spomin in ga tam procesira, zato ni primeren za delo z večjimi datotekami. Aplikacije DOM-a se lahko uporabljajo na obeh straneh sodelujočih v procesu, na strani odjemalca ali pa na strani strežnika. Na strani odjemalcev se DOM uporablja za prikazovanje podatkov XML in za oblikovanje podatkov, ki jih uporabnik vnese na strani odjemalca in se potem prenesejo na stran strežnika. Na strani strežnikov pa se DOM uporablja za arhiviranje informacij in za izmenjavo podatkov med posameznimi notranjimi in zunanjimi procesi ali med bazami podatkov.

5.2 Enostavni vmesnik uporabniškega programa za XML

Za razliko od vmesnika DOM temelji enostavni vmesnik uporabniškega programa za XML SAX na dogodkih. To pomeni, da razčlenjevalnik bere dokument in sporoči programu, ko najde določen simbol – na primer, začetek ali konec elementa – določene znakovne podatke, ipd.

SAX ne prebere celotnega XML dokumenta v spomin, zato je primeren za delo tudi z velikimi datotekami. Poleg tega omogoča, da hranimo podatke v kompleksnih podatkovnih strukturah, ki jih definiramo v svoji aplikaciji, in jih spreminjamo, ko vmesnik sproži določen dogodek. Pogosto potrebujemo iz XML dokumenta samo majhen del informacije, ki jo nosi,

na primer število določenih elementov. V tem primeru ni nobene potrebe, da preberemo dokument v celoti, saj SAX omogoča, da ignoriramo podatke, ki nas ne zanimajo. SAX je enostaven za uporabo in pomeni tudi najhitrejši način dostopa do določene informacije, ki zahteva samo enkraten prehod skozi dokument. To, kar so v določenih primerih prednosti vmesnika SAX, so v drugih njegove slabosti. Ker se dokument ne nahaja v spominu, ni možno naključno dostopati do njegovih posameznih delov, kar postane problematično zlasti pri dokumentih, ki vsebujejo navzkrižne reference. Tudi implementacija kompleksnih iskanj v SAX je zahtevna, saj moramo sami definirati strukture, ki hranijo ustrezne podatke za nadaljnje poizvedbe. Poleg tega SAX ne nudi informacij o strukturi podatkov v DTD-ju niti ne nekaterih informacij o načinu zapisa podatkov v dokumentu. SAX ni primeren za spreminjanje, je pa primeren za izdelavo in branje dokumentov (Rojko, 2005).

Aplikacija, ki jo uporablja SAX, je sestavljena iz razčlenjevalnika, ki je prilagojen standardu SAX in glavnega programa, programske kode, ki procesira vsebino dokumenta in jo imenujemo upravljavec dokumenta. V aplikaciji izdelamo razčlenjevalnik in upravljavec dokumenta, povemo razčlenjevalniku, kateri upravljavec dokumentov naj uporablja in začne procesirati vhodni dokument. Naloga razčlenjevalnika je, da obvesti upravljavca dokumenta na pomembne dogodke znotraj dokumenta. Na zahtevo aplikacije procesira določene dogodke in izvede aktivnosti.

Ločimo več tipov dogodkov. Dogodke dokumenta tvorita metodi, ki označujeta njegov začetek in konec, prav tako imamo par metod, ki označujejo začetek in konec elementov. Nadalje vsebuje SAX še metodo, ki vrne vrednost danega atributa ter metode za znakovne podatke, navodila za procesiranje in odkrivanje napak. Napake, ki sprožijo dogodke, delimo v tri skupine: prve so napake pri odpiranju dokumenta in drugih virov, druge so napake, ki jih odkrije razčlenjevalnik in so povezane z dobro oblikovanostjo in veljavnostjo dokumentov, tretje pa so napake, ki jih javi sama aplikacija (Čavlek, 1999).

Obravnavanje vseh dogodkov v večjem dokumentu, z veliko različnimi elementi pogosto postane nepregledno, saj je težko razločevati med posameznimi dogodki, v vsakem odzivu na dogodek pa se izvaja vrsta med seboj neodvisnih aktivnosti. Zato obstajata dve metodi, ki sta namenjeni reševanju kompleksnejših problemov v vmesniku SAX. Prvo imenujemo filter

skozi cev, kjer celotno procesiranje razdelimo na več vmesnih stopenj, od katerih ima vsaka svoj vhodni dokument in vrača kot rezultat neki izhodni dokument. Druga metoda temelji na zbirki pravil oblike »Če nastopi ta in ta dogodek, potem izvedi to opravilo.« in je v bistvu nadgradnja programiranja, ki temelji na dogodkih.

5.3 Primerjava DOM in SAX razčlenjevalnika

Prednosti in slabosti SAX razčlenjevalnika (Jurič, 2001):

		Razlaga
Prednosti	Manjša poraba pomnilnika	Za razliko od DOM modela SAX razčlenjevalnik obdeluje XML dokument postopoma, kar pomeni, da se v pomnilnik naloži le manjši del vsebine dokumenta. Za to je primeren tudi za obdelavo večjih količin podatkov, predvsem če ni potrebno spreminjati vsebine dokumenta.
	Delni pregled dokumenta	SAX razčlenjevalnik dopušča prekinitev obdelave dokumenta kadarkoli, zato je uporaben za iskanje posameznih podatkov, ki se nahajajo v dokumentu. Ta lastnost je tudi uporabna pri odkrivanju napak, saj lahko izvajanje obdelave ob prvi napaki prekinemo.
	Pridobitev delov dokumenta	SAX razčlenjevalnik uporabimo, kadar želimo iz dokumenta izluščiti posamezne informacije. V ta namen namreč ni potrebno v pomnilnik naložiti celotnega XML dokumenta.
	Pretvorbe strukture	SAX razčlenjevalnik je uporaben pri določenih pretvorbah XML dokumenta, zlasti ko nas zanimajo podatki na višjih nivojih drevesa.
Slabost	Odkrivanje napak	V kolikor želimo odkriti vse napake v XML dokumentu, moramo pregledati celoten dokument, saj se preverjanje ustreznosti XML dokumenta izvaja sproti med branjem posameznih delov, naključno dostopanje do delov dokumenta pa ni mogoče. V določenih primerih to predstavlja slabost.

Prednosti in slabosti DOM razčlenjevalnika (Jurič, 2001):

		Razlaga
Prednosti	XSL pretvorbe	DOM razčenjevalnik deluje bolje pri XSL (eXstensible Stylesheet Language) pretvorbah, kjer XML dokument preoblikujemo na osnovi XSLT (eXstensible Style Sheet Language Transformations) šablone.
	XPath poizvedbe	Pri uporabi zahtevnih XPath (XML Path Language) poizvedb, razultat katerih so kompleksni XML elementi, je potrebno uporabiti DOM razčlenjevalnik, saj se drevesna struktura ohrani tudi v rezultatu poizvedbe.
	Spremembe vsebine XML dokumenta	DOM razčlenjevalnik je namenjen izdelavi ali popravljanju, kot tudi branju XML dokumenta v pomnilnik. Na drugi strani je SAX razčlenjevalnik namenjen predvsem učinkovitem branju XML dokumentov.
Slabosti	Velika poraba pomnilnika	DOM razčlenjevalnik preoblikuje XML dokument v drevesno strukturo v pomnilniku. Poraba pa je znatno večja od dejanske velikosti dokumenta.

6 OBLIKOVANJE XML DOKUMENTOV

Standard XML (eXstensible Markup Language) loči podatke od pravil, kako se naj ti podatki prikazujejo. XML podatki so sicer končnemu uporabniku berljivi, vendar nepregledni, saj so namenjeni predvsem računalniški obdelavi. V kolikor pa iste podatke zapišemo in predstavimo v uporabniku prijaznejši obliki, dodamo podatkom novo uporabno vrednost. Zmožnost preoblikovanja XML podatkov v druge oblike dodatno prispeva k uporabnosti XML tehnologije.

Za predstavitev podatkov v XML-u se uporabljajo stilni jeziki, ki temeljijo na deklarativnem programiranju in transformirajo dokument XML v zbirko elementov, ki se slušno ali vidno zaznavajo. Deklarativno programiranje pomeni, da aplikacija dobi podatek kaj želimo, ne pa tudi, kako naj pride do rezultata. Stilne jezike zaznamujeta dve skupini pravil: navodila za delovanje in pravila za ujemanje po vzorcu oziroma predlogi (Anderson s sod., 2000).

V uporabi sta dva mehanizma, ki izpisujeta podatke:

- kaskadne stilne predloge CSS (Cascading Style Sheets);
- razširljiv jezik za stil XSL (Exstensible Stylesheet Language).

Pri predstavitvi dokumenta XML v spletni aplikaciji sta dve možnosti prikaza: na strani odjemalca (uporabnika) ali pa na strani strežnika. V primeru da uporabnik uporablja spletni brskalnik, ki ne podpira XML-a, mora strežnik preoblikovati dokument v predstavitveno obliko in šele nato ga pošlje odjemalcu. V nasprotnem primeru, ko brskalnik uporabnika podpira XML, pa spletni strežnik enostavno pošlje dokument XML s pripadajočo stilno predlogo brskalniku uporabnika, ki nato dokument sam preoblikuje v predstavitveno obliko.

6.1 Kaskadne stilne predloge

Kaskadne stilne predloge CSS (Cascading Style Sheets) temeljijo na modelu škatel. Škatla je definirana kot pravokotno območje z osnovnimi značilnostmi, kot so dimenzije in odmiki od

robov. Škatle so zložene ena v drugo. Tak model omogoča predstavitev v obliki drevesa pravokotnih območij. Drevo vseh škatel je vsebovano v krovnem pravokotniku, ki je odvisen od predstavitvenega medija. Lahko je nepretrgan kot v brskalniku, kjer se lahko pomikamo po zaslonu gor in dol, lahko pa je fiksnih dimenzij kot v primeru tiskanih medijev. Elemente v dokumentu XML povezujemo s škatlami preko pravil v jeziku CSS. Pravila sestavljajo množice lastnosti, ki so povezane z določenim tipom elementa ali elementov. Vsako pravilo je sestavljeno iz proceduralnega dela, ki vsebuje zbirko lastnosti ter iz vzorcev za ujemanje, ki ga imenujemo selektor. Lastnosti opisujejo, kako naj bo škatla predstavljena, selektor pa določa ime elementa (Andersons sod., 2000).

Model CSS je odvisen od strukture elementa, ki ga predstavlja. Drevo komponent dokumenta XML in drevo objektov za obliko in predstavitev sta pogosto enaka. Najpomembnejši predstavitveni objekti v CSS-u so:

- plavajoči objekt,
- tabela,
- vrstični objekt,
- seznam,
- blok.

Smer izpisa določa, v kakšnem vrstnem redu so bloki zloženi eden na drugega V primeru, da je smer izpisa od vrha proti dnu in od leve proti desni, potem so bloki v krovnem pravokotniku nanizani od vrha proti dnu. Vrstični objekti so vključeni v blokih. Smer pisanja določa, kako se vrstični objekti izpišejo eden za drugim v smeri pisanja in znotraj območja, ki jim je na voljo. Poleg tega omogoča CSS tudi postavljanje posameznih objektov na absolutne pozicije ali pa postavljanje posebnih plavajočih pravokotnikov, okrog katerih se razporedi vsebina ostalih objektov (Jurič, 2001).

Z jezikom CSS lahko podatke prikažemo na več načinov:

- izpis na tiskalnik,
- izpis na ekran,
- posredovanje na govorno napravo,

- posredovanje na televizijo,
- posredovanje na napravo za slepe.

Vidimo, da lahko isti XML dokument s CSS-om prikažemo na različne načine, ne da bi posegali v podatke same. Zapis podatkov v dokumentu XML z jezikom CSS je neodvisen od aplikacije ali naprave, ki te podatke prikazuje.

6.2 Razširljiv jezik za stil

Jezik XSL (eXstensible Stylesheet Language) opisuje kako prikazati XML dokument. Zahteva izvorni XML dokument, ki vsebuje informacije, ki jih bo XSL prikazal in dokument, ki določa pravila, ki jih bo stil XSL uporabil za prikaz. V primerjavi s CSS-om je XSL bolj prilagodljiv in zato tudi uporaben.

Razširljiv jezik za stil omogoča:

- filtriranje podatkov v XML dokumentu,
- pretvorbo slovnice XML dokumenta v drugo obliko slovnice,
- naslavljanje določenega dela dokumenta,
- oblikovanje XML dokumenta glede na določeno vrednost,
- sortiranje podatkov.

XSL jezik sestavljajo trije deli (W3C, 2008):

- XSLT (eXstensible Stylesheet Language Transformations) – jezik za opisovanje transformacij med XML dokumenti,
- XPath (XML Path Language) – jezik za naslavljanje XML dokumentov in
- XSL-FO (XSL Formatting Objects) – jezik za oblikovanje dokumentov.

6.2.1 Jezik za oblikovanje izpisa XML dokumentov

Transformacije XML dokumentov s pomočjo razširljivega jezika za stil XSL omogočajo veliko bolj prožen način prikazovanja XML dokumentov. Ena od možnosti uporabe razširljivega jezika za stil je, da ga uporabljamo za ustvarjanje drugih oblik predstavitvenih objektov, kot so predstavitve, ki temeljijo na XML-u in jih lahko izvedemo s pretvorbami. Za uporabnika, ki nima spletnega brskalnika, ki podpira XML, je najbolj priročna transformacija v HTML (Hyper Text Markup Language), saj omogoča, da mu strežnik, ki preoblikuje dokument, pošlje verzijo HTML, ki pa jo brskalnik podpira.

XSL-FO je jezik za oblikovanje izpisa XML dokumentov. Je specifikacija, ki se ukvarja z oblikovanimi objekti, in je del jezika XSL. XSL-FO omogoča transformacijo XSL dokumenta v obliko, ki ne temelji na XML osnovi, a je primerna za predstavitev podatkov. Vrste transformacij, ki jih XSL-FO podpira (Anderson s sod., 2000):

- pretvorba XML dokumenta v obliko dokumenta z obogatenim tekstom RTF (Rich Text Format),
- pretvorba XML dokumenta v obliko prenosljivega formata dokumenta PDF (Portable Document Format),
- pretvorba XML dokumenta v obliko dokumenta, napisanega z matematičnim besedilom, napisanega v jeziku TeX,
- transformacija XML dokumenta v drevo oblikovanih objektov XSL-FO – XSL deluje kot zbirka pravil na določenem vzorcu ali predlogi, ki iz dokumenta izbere določene elemente in jih prevede v obliko, kot je navedena v navodilih predloge,
- pretvorba v katerokoli od poljubnih oblik, saj nam razširljivost jezika omogoča poljubno kreiranje glede na pravila, ki jih moramo upoštevati.

Spodnja slika prikazuje osnovno zgradbo XSL-FO dokumenta. XSL-FO so XML dokumenti, zato se dokument začne z deklaracijo. Sledi korenski element <fo:root>, ki vsebuje tudi deklaracijo imenskega prostora. Elemet <fo:simple-page-master> določa enolično predlogo za stran, zato mora imeti enolično oznako, ki jo definira atribut master-name. Poljubno število

elementov <fo:page-sequence> opisuje vsebino strani. Vrednost atributa master-reference določa povezavo med vsebino strani in enolično predlogo.

(vir: http://www.w3schools.com/xslfo/xslfo_documents.asp)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
 <fo:simple-page-master master-name="A4">
  <!—predloga strani -->
 </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference="A4">
 <!—vsebina strani -->
</fo:page-sequence>
</fo:root>
```

6.2.2 Jezik za opisovanje pretvorb med dokumenti

Jezik XSLT (eXstensible Stylesheet Language Transformations) skupaj z XPath (XML Path Language) predstavlja jezik za opisovanje pretvorb med dokumenti. Oba skupaj omogočata implementacijo drevesno orientiranih jezikov, ki preslikujejo XML dokumente iz enega slovarja v poljuben drug slovar.

Vsebinsko ločimo tri tipe preoblikavanj (Anderson s sod., 2000):

- izdelavo dinamičnih dokumentov – filtriranje, sortiranje, ali preurejanje delov dokumenta XML,
- preoblikovanje v predstavitveni jezik – dokument XML se preoblikuje v brezžični označevalni jezik WML (Wireless Markup Language) za predstavitev na mobilnih telefonih ali pa v obliko HTML, ki je primerna za prikaz v spletnih brskalnikih,
- prevajanje – transformacija strukture, kjer se transformira en slovar XML-a v drugega.

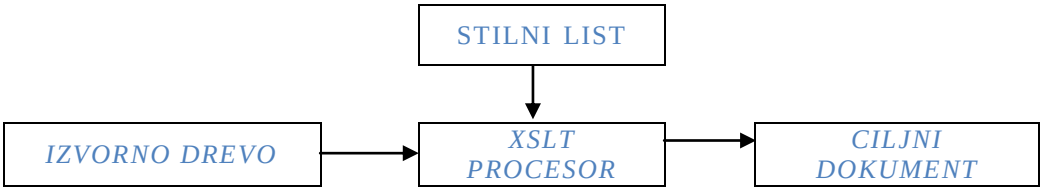
Jezik XSLT, ki pri tem uporablja jezik XPath, določa, kako procesor XSLT izdela zaželeni rezultat transformacije na osnovi vhodnih podatkov. Jezik XPath določa kaj je potrebno transformirati, jezik XSLT pa poskrbi za dejansko izvedbo transformacije. Pri tem se

uporabljajo stilni listi. Stilni list je XML dokument, ki uporablja elemente iz slovarja XSLT za opis transformacij. Obsega množico pravil za transformiranje izvornega XML dokumenta v ciljni XML dokument (Mohorič in Zrnec, 2008).

Transformacija se izvaja nad drevesno strukturiranim izvornim dokumentom in pri tem gradi ciljno drevo, ki predstavlja rezultat preslikave.

Zato, da se lahko izvede transformacija, potrebuje XSLT procesor dve sestavini:

- izvorno drevo,
- stilni list (več stilnih listov), s transformacijskimi pravili.



Slika 16 Proces transformacije izvornega drevesa v ciljni dokument

Stilni list je XML dokument, ki uporablja elemente iz slovarja XSLT za opisovanje transformacij. Element vsakega stilnega lista je element <xsl:stylesheet>, katerega vsebina je množica pravil za opis transformacije, ki se naj izvede. Vsako pravilo v stilnem listu vsebuje pridruženi vzorec XPath, ki se ujema z vozlišči v izvornem dokumentu, nad katerim naj se to pravilo aplicira.

Vsako pravilo, poimenovano model, je predstavljeno z elementom <xsl:template> z atributom match="vzorec" in njegovo pridruženo vrednostjo Xpath. Vsako pravilo je model. Znakovni elementi in atributi, vsebovani v telesu pravila, delujejo kot načrt za konstruiranje dela drevesa, ki predstavlja rezultat preslikave – ciljno drevo. Procesor XSLT konstruira vsebino modela pravila v ciljno drevo, kadarkoli naleti na izvorno vozlišče, ki se ujema z vzorcem pravila.

Ko se ustrezni model izdela, se izvedejo naslednje operacije (Mohorič in Zrnec, 2008):

- Ciljni elementi in atributi v modelu se kreirajo v ciljnem drevesu. Ciljni elementi in atributi, ki niso iz imenskega prostora XSLT, se obravnavajo kot literali in se nespremenjeni pojavijo v ciljnem drevesu.
- Vsi modeli z atributno vrednostjo, vsebovani znotraj literalnih atributivnih vrednosti, se nadomestijo z vrednostjo njihovega izraza XPath.
- Vsi modeli v imenskem prostoru XSLT se obdelujejo v zaporedju dokumentov.

Osnovna operacija je, na kratko, naslednja: ko se vozlišče v izvornem drevesu ujema z vzorcem pravila, se vsebina tega pravila kreira v ciljnem drevesu. S pomočjo izvornega drevesa in stilnega lista izvede procesor XSLT transformacijo, opisano s pravili v stilnem listu.

V okviru obdelave liste vozlišč izvornega drevesa se ustvarjajo fragmenti ciljnega drevesa. Pri obdelavi tekočega vozlišča se upoštevajo vsa možna pravila, ki se z vozliščem ujemajo, nato pa se izbere in aplicira najustreznejše pravilo.

Obdelava se prične z listo vozlišč, ki obsega le koren dokumenta. Procesor poišče model, ki se ujema s tem korenskim vozliščem (tipično pravilo, ki se ujema z match="/") in kreira vsebino modela v ciljnem drevesu ob upoštevanju treh osnovnih procesnih korakov. Če model vsebuje elemente iz imenskega prostora XSLT, ki izberejo naslednja vozlišča za obdelavo, potem se postopek rekruzivno nadaljuje, dokler niso obdelana vsa vozlišča. Ko je obdelava zaključena, predstavlja ciljno drevo ciljni dokument, izdelan s transformacijo. Pri tvorjenju ciljnega drevesa, ki predstavlja rezultat preslikave, so elementi izvornega drevesa lahko filtrirani in preurejeni, lahko pa se doda tudi poljubna struktura.

6.3 Prikaz prostorskih podatkov z uporabo SVG in XSLT jezika

Kmalu po predstavitvi SVG (Scalable Vector Graphics) standarda so se začeli pojavljati interesi rabe formata za uporabo v spletni kartografiji. Kljub veliko različnim možnostim uporabe XSLT jezika, predstavlja raba v namen kartografije pomembno skupino.

Projekti različnih namenov in ciljev se ukvarjajo s pretvorbo med GML (Geography Markup Language), katerega glavni namen je shranjevanje prostorskih podatkov, in SVG formatom. Ker oba jezika temeljita na XML tehnologiji, obstaja možnost transformacije med jezikoma z uporabo XSLT jezika. Z razvojem tehnike transformacij z uporabo XSLT jezika se je aktivno ukvarjalo podjetje SchemaSoft. Britanska kartografska agencija Ordnance Survey je eden izmed prvih ponudnikov podatkov v GML formatu. Zato ni čudno, da je postala zagovornik transformacije z uporabo XSLT jezika ponujenih podatkov v SVG format (Zaslavsky, 2003).

Jezik XSLT omogoča kartiranje podatkov zapisanih v XML standardu iz raznolikih podatkovnih virov. Vseeno pa različni zapisi entitet zahtevajo izdelavo novih pravil za izvedbo transformacije, za katere je potreben čas in znanje. Z uporabo nižjega programskega jezika za potrebe računalniško podprtega kartiranja pridobimo na prilagodljivosti, ki omogoča neposredno kontrolo nad procesom transformacije podatkov. Vendar pa nižji programski jeziki, v tem primeru jezik XSLT, zahtevajo več znanja in časa (Adams, 2005).

6.3.1 Transformacija XML dokumenta kabelske kanalizacije

Transformacija izvornega drevesa XML dokumenta kabelske kanalizacije A_6001_NG_TKPIS.xml v ciljni SVG dokument, je bila narejena s XSLT procesorjem SAXON.

```
C:\saxon\bin>Transform.exe -o index.svg A_6001_NG_TKPIS.xml xml2svg.xslt
C:\saxon\bin>
```

Slika 17 Zaporedje argumentov, ki jih potrebuje XSLT procesor

Spodnja slika prikazuje pravilo – del stilnega lista xml2svg.xslt, ki transformira trasne elemente XML dokumenta kabelske kanalizacije v SVG zapis.

```
<xsl:template match="TRObjekti">
  <g id="JA">
    <xsl:apply-templates select="TRObjekt[@OznakaSTR='JA']"/>
  </g>
  <g id="PJ">
    <xsl:apply-templates select="TRObjekt[@OznakaSTR='PJ']"/>
  </g>
  <g id="OS">
    <xsl:apply-templates select="TRObjekt[@OznakaSTR='OS']"/>
  </g>
  <g id="OM">
    <xsl:apply-templates select="TRObjekt[@OznakaSTR='OM']"/>
  </g>
  <g id="TS">
    <xsl:apply-templates select="TRObjekt[@OznakaSTR='TS']"/>
  </g>
</xsl:template>
```

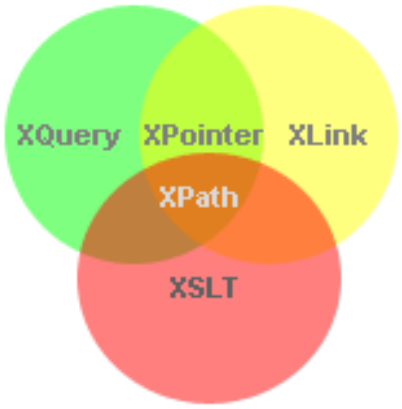
Slika 18 Stilni list

Stilni list xml2svg.xslt vključuje pravila oziroma modele, ki transformirajo podatke iz izvornih dokumentov: XML dokumenta kabelske kanalizacije in XML dokumenta, ki vsebuje podatke o hišnih številkah – EHIŠ (Evidenca hišnih številk). Rezultat transformacije je bil v tej diplomski nalogi SVG dokument index.svg, ki skupaj z navigacijskimi orodji tvori interaktivno karto.

7 ISKANJE IN POVPRAŠEVANJE PO XML DOKUMENTIH

7.1 XPath

XPath (XML Path Language) je jezik, ki se uporablja za iskanje in pridobivanje informacij v vozliščih XML dokumenta. Pomen jezika XPath prikazuje spodnja slika (vir: http://www.w3schools.com/xpath/default.asp).



XPath poizvedbe so oblikovane kot izrazi. Ti izrazi se uporabljajo za naslavljanje različnih delov XML dokumenta, za upravljanje nizov, števil in logičnih vrednosti ter za ujemanje niza vozlišč v dokumentu. XPath prikazuje XML dokument kot drevo različnih vrst vozlišč. XPath izrazi v XML dokumentu določajo vozlišča na osnovi njihovega tipa, imena in vrednosti ter relacije vozlišč do drugih vozlišč v dokumentu (W3C, 2008)

7.1.1 Vozlišča

Med vrste vozlišč v jeziku XPath spadajo:

- dokumentni element (korenski element),
- element,

- atribut,
- tekst,
- imenski prostor,
- procesna inštrukcija,
- komentar.

7.1.2 Relacije med vozlišči

Za delo z vozlišči je pomembno razumevanje relacij (odnosov) med vozlišči.Na spodnji sliki so razložene relacije med vozlišči. Vsak element in atribut ima enega starša (starš elementov <TRObjekt> je element <TRObjekti>). Elementi lahko imajo nič, enega ali več otrok (elementi <Coor> so otroci elementa <SITGEOMETRY>). V isto družino spadajo elementi z istim staršem (elementi <Coor> spadajo v isto družino). Prednik je staršev starš (prednika elementa <TRObjekt> sta elementa <TRObjekti> in <TKOmrezje>). Potomci so otroci otrok (elemeta <SITGEOMETRY> in <Coor> sta potomca elementa <TRSegment>).

```
<TKOmrezje Vrsta-dok="DOKUMENTACIJA KANALIZACIJE">
  <TRObjekti>
    <TRObjekt Poz_x="394982.368" Poz_y="91394.208"></TRObjekt>
    <TRObjekt Poz_x="394982.311" Poz_y="91393.208"></TRObjekt>
  </TRObjekti>
  <TRSegment Dolzina="1.20">
    <SITGEOMETRY VertNumber="2">
      <Coor ID="0">395303.533</Coor>
      <Coor ID="1">91313.373</Coor>
      <Coor ID="2">395302.933</Coor>
      <Coor ID="3">91314.423</Coor>
    </SITGEOMETRY>
  </TRSegment>
</TKOmrezje>
```

Slika 19 Relacije med vozlišči

Vozlišča brez otroka ali starša so nesestavljene (atomarne) vrednosti. Predmeti so nesestavljene vrednosti ali vozlišča.

```
394985.368  91394.208
```

Slika 20 Primer atomarne vrednosti

7.1.3 Izbiranje vozlišč

Jezik Xpath za izbiro vozlišč v XML dokumentu uporablja izraze za pot. Vozlišče se lahko izbere z uporabo poti, oziroma korakov. Spodnja preglednica prikazuje glavne ukaze za izbiro vozlišč.

(vir: http://www.w3schools.com/xpath/xpath_syntax.asp, 2008)

Ukaz	Opis
vozlišče	Izbere vsa podrejena vozlišča v vozlišču.
/	Izbere vsa vozlišča od korenskega elementa naprej.
//	Izbere vsa vozlišča dokumenta od trenutnega vozlišča navzdol.
.	Izbere trenutno vozlišče.
..	Izbere nadrejeno vozlišče trenutnega vozlišča.
@	Naslavlja atribut.
*	Izbere vse elemente vozlišča.
@*	Izbere vse atribute vozlišča.
node()	Izbere vsa vozlišča ne glede na tip.

7.1.4 Izrazi za pot

Izraz za pot je lahko absolutni ali relativni. Absolutni izraz za pot se prične z znakom /. Izraz za pot je lahko sestavljen iz več korakov, ki so med seboj ločeni z znakom / (poševnico). Vsak korak za pot je sestavljen iz: (W3Schools, 2008)

- osi (),

- testa vozlišča (položaj vozlišča znotraj osi),
- nič ali več filtrov.

Spodnja slika prikazuje splošno sintakso izraza za pot.

```
os::test_vozlišča[filter]
```

Slika 21 Splošna sintaksa izraza za pot

Preglednica 1 Izrazi za pot

Izraz za pot	Rezultat
TRSegment	Izbere vsa vozlišča, ki otroci elementa »TRSegment«.
/TKOmrezje	Izbere korenski element »TKOmrezje«. Če se izraz za pot prične z znakom »/«, to predstavlja absolutno naslavljanje elementa.
TRSegmenti/TRSegment	Izbere vse elemente z imenom »TRSegment«, ki so otroci vozlišča »TRSegmenti«.
//TRSegment	Izbere vse elemente z imenom »TRSegment«, ne glede na njihov položaj v drevesu dokumenta. Če se izraz za pot prične z znakom »//«, to predstavlja relativno naslavljanje elementa.
TRSegment//Coor	Izbere vse elemente z imenom »Coor«, ki so otroci vozlišča »TRSegment«, ne glede na to, kje znotraj elementa »TRSegment« se nahajajo.
//@Poz_x	Izbere vse atribute z imenom »Poz_x«.
//SITGEOMETRY/Coor[1]	Izbere prvi element z imenom »Coor«, ki je otrok elementa »SITGEOMETRY«.
//SITGEOMETRY/Coor[last()]	Izbere zadnji element z imenom »Coor«, ki je otrok elementa »SITGEOMETRY«.
//TRObjekt[@OznakaSTR= 'JA']	Izbere vse elemente z imenom »TRObjekt«, ki imajo atribut z imenom »OznakaSTR«, katerega vrednost je »JA«.

7.1.5 Osi

Os v jeziku XPath določa relacijo nabora vozlišč v drevesni strukturi glede na trenutno vozlišče (vir: http://www.w3schools.com/xpath/xpath_axes.asp, 2008).

Ime osi	Rezultat
Ancestor	Izbere vse starše trenutnega vozlišča.
Ancestor-or-self	Izbere vse starše trenutnega vozlišča in tudi trenutno vozlišče.
Attribute	Izbere vse atribute trenutnega vozlišča.
Child	Izbere vse otroke trenutnega vozlišča.
Descendant	Izbere vse potomce trenutnega vozlišča.
Descendant-or-self	Izbere vse potomce trenutnega vozlišča in tudi trenutno vozlišče.
Following	V XML dokumentu izbere vse kar sledi končni oznaki trenutnega vozlišča.
Following-sibling	Izbere vse družine za trenutnim vozliščem.
Namespace	Izbere vsa vozlišča imenskih prostorov trenutnega vozlišča.
Parent	Izbere starša trenutnega vozlišča.
Preceding	V XML dokumentu izbere vse kar je pred začetno oznako trenutnega vozlišča.
Preceding-sibling	Izbere vse družine pred trenutnim vozliščem.
Self	Izbere trenutno vozlišče.

7.1.6 Funkcije

Poleg izrazov za pot ima XPath nekaj funkcij, ki se jih lahko uporabi za pridobivanje podatkov iz XML dokumenta.

Preglednica 2 Vrste funkcij, ki jih podpira jezik XPath

Vrste funkcij	Opis
vozliščne	Delo z različnimi vrstami vozlišč.
pozicijske	Uporablja se jih za vozlišča, ki pripadajo naboru vozlišč. Te funkcije določajo, ali se ujemajo s položajem vozlišča.
številske	Uporablja se jih za vračanje številskih vrednosti.
logične	Uporablja se jih za ocenjevanje XPathovih izrazov za resnično ali neresnično.
nizne	Uporablja se jih za upravljanje in razčlenjevanje niznega besedila

7.2 XQuery

Z naraščajočo količino XML dokumentov narašča tudi potreba po pregledovanju in iskanju vsebine teh dokumentov. Z obstoječimi poizvedovalnimi jeziki, ki jih poznajo relacijske podatkovne baze, si tu ne moremo kaj prida pomagati.

V ta namen so se razvili poizvedovalni jeziki, namenjeni iskanju po podatkih zapisanih v XML obliki, med katerimi prevladuje poizvedovalni jezik XQuery, razvit v W3C. XQuery se je razvil kot derivat več poizvedovalnih jezikov: Quilt, XQL (XML Query Language), XML-QL (A Query Language for XML), SQL (Structured Query Language), OQL (Object Query Language), Lorel and YATL (Yet Another Transformation Language) (Lehti, 2001).

XQuery je pravzaprav poizvedovalni jezik sestavljen iz treh jezikov. Prva plast je površinska, vidna plast, s katero se sreča uporabnik ob pisanju poizvedbe. Druga plast je v XML preoblikovana prva plast. Zapisana v jeziku XQueryX (XML Syntax for XQuery), ki je XML zapis za XQuery. Ta oblika zapisa ni najbolj berljiva, je pa primerna za obdelavo s programsko opremo, saj je zapisana v XML obliki. Tretjo, zadnjo plast, predstavlja algebraični jezik, ki opisuje notranje delovanje poizvedovalnega jezika (Poljšak, 2005).

XPath si kot veja iz katere se je razvil XQuery, deli s poizvedovalnim jezikom XQuery skupen podatkovni model in se uporablja za naslavljanje delov dokumenta.

7.2.1 Gradniki izrazov

Osnovni gradniki XQuery izrazov so (Jurič, 2001):

- izrazi za pot,
- konstruktorji elementov
- FLWR (For, Let, Where, Return) izrazi,
- operatorji in funkcije,
- pogojni izrazi.

7.2.2 Izrazi za pot

Z izrazi za pot naslavljamo določene dele drevesno organiziranega XML dokumenta. V ta namen se uporablja naslovni jezik XPath. Tako tehnologiji XQuery kot XSLT, temeljita na osnovah naslovnega jezika XPath.

Z Xpath izrazi lahko naslavljamo vozlišča (eno ali več) v dokumentu, posamezne elemente ali atribute. XPath ukazi vračajo množico vozlišč, niz znakov, številko ali logično vrednost. Spodnji primer predstavlja izraz za pot, ki naslavlja element <RAZDALJA>.

```
doc('A_6001_NG_TKPIS.xml') /TKOmrezje/TRSegmenti/RAZDALJA
```

Slika 22 XPath izraz za pot

7.2.3 Konstruktorji izrazov

S konstruktorji izrazov lahko ustvarimo nov XML element ali atribut. Konstruktor sestavlja začetna oznaka, končna oznaka in seznam izrazov, ki sestavlja vsebino elementa.

Konstruktorje delimo na neposredne in izračunane. Konstruktor lahko uporabimo za izdelavo elementa, atributa ali sestavljenega elementa. Vsebino elementa ali atributa pa lahko določamo tudi dinamično, pri čemer govorimo o izračunanemu konstruktorju.

Primer na spodnji sliki prikazuje konstruktor, ki v korenski element <Seznam_jaskov> gnezdi elemente <Jasek>.

```
(:seznam jaskov:)
<Seznam_jaskov>
 {
  for $x in doc("A_6001_NG_TKPIS.xml")/TKOmrezje/TRObjekti/TRObjekt[@OznakaSTR = "JA"]
  order by $x/@OZNAKA
  return
   <Jasek>
    { $x/@OZNAKA } { string($x/@Poz_y) }, { string($x/@Poz_x) }, { string($x/@Poz_z) }
   </Jasek>
 }
</Seznam_jaskov>
```

Slika 23 Konstruktor elementa <Jasek>

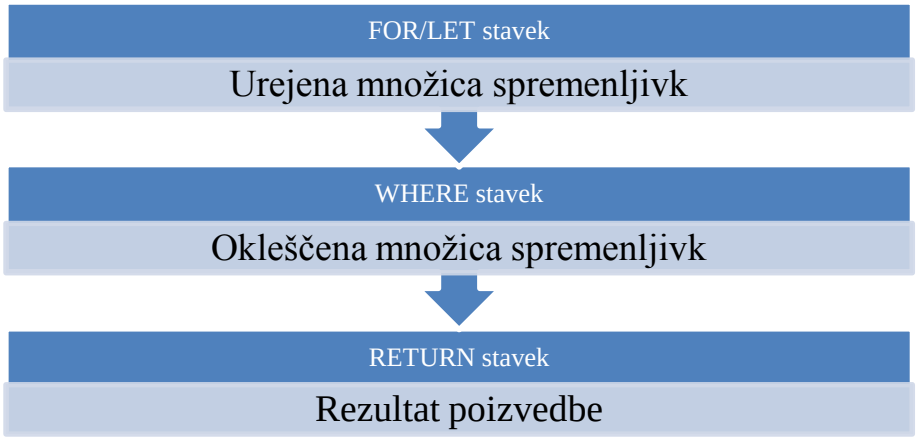
7.2.4 FLWR izrazi

Namesto izrazov za pot se lahko uporablja tudi drugo izrazje. Zelo razširjen je izraz FLWR, ki je podoben SQL (Standard Query Language) poizvedbi SELECT-FROM-WHERE. FLWR predstavlja telo poizvedbe, v katerem povežemo več spremenljivk v končni rezultat.

Stavek FOR uporabimo za ponavljanje ukazov v zankah. S stavkom LET pa spremenljivkam prirejamo vrednosti, ki so lahko matematični izrazi ali vrednosti iz dokumenta. Stavki LET se ponovijo v vsaki ponovitvi zanke. Množico spremenljivk, ki jih dobimo pri FOR in LET

stavkih nato še oklestimo z pogojem WHERE in sortiramo s stavkom ORDER BY. Samo tiste vrednosti, ki ustrezajo WHERE pogoju sestavljajo rezultat poizvedbe, ki ga vrne rezultat RETURN.

Spodnja slika prikazuje predpisano zaporedje FLWR izrazov (Jurič, 2001).



Primer na spodnji sliki prikazuje enostaven primer FLWR izraza s konstruktorjem, ki vrne prvih dvajset vrednosti za spremenljivko $x.

```
for $x in doc('A_6001_NG_TKPIS.xml')/TKOmrezje/TRObjekti/TRObjekt[@OznakaSTR = 'JA']
let $y := count($x)
where $y < 20
return
 <Jasek>
  { $x/@OZNAKA } { string($x/@Poz_y) }, { string($x/@Poz_x) }, { string($x/@Poz_z) }
 </Jasek>
```

Slika 24 FLWR izraz s konstruktorjem

7.2.5 Pogojni izrazi

Pogojne izraze uporabimo tam, kjer je vsebina rezultata odvisna od določenega pogoja. Pogojne izraze lahko poljubno gnezdimo in uporabimo povsod, kjer potrebujemo logičen izračun vrednosti nekega izraza.

Pogojni izraz mora biti vedno zapisan v oklepajih. Obvezna je tudi uporaba izraza else, za katerega pa ni nujno, da vrne vrednost. Lahko je oblike else().
Slika v spodnjem primeru prikazuje pogojni izraz s konstruktrjem <Jasek>. Poizvedba vrne rezultat za vsako vrednost atributa OznakaSTR, katere vrednost je enaka JA.

```
for $x in doc("diploma/A_6001_NG_TKPIS.xml")/TKOmrezje/TRObjekti/TRObjekt
return
 if($x/@OznakaSTR = "JA") then <Jasek>{ $x/@OZNAKA } </Jasek>
 else ()
```

Slika 25 Pogojni izraz s konstruktorjem elementa

7.2.6 Funkcije

XQuery ponuja množico vnaprej določenih ukazov in funkcij za delo z vozlišči, nizi, števili, datumski podatki in logičnimi izrazi. URI imenskega prostora vgrajenih XQuery funkcij je: http://www.w3.org/2005/02/xpath-functions. Ime predpone privzetega imenskega prostora je fn:. Uporaba predpone pri klicanju privzetih funkcij ni nujna.

Funkcije lahko uporabnik deklarira tudi sam. Uporabniško določene funkcije so lahko definirane v sami poizvedbi ali v ločeni knjižnici. Uporabniško določena funkcija se deklarira s ključno besedo declare. Pri uporabniško določeni funkciji je obvezna uporaba predpone. Podatkovni tipi parametrov (spremenljivk) so večinoma enaki podatkovnim tipom definiranimi v XSD (XML Schema Datatypes) (W3C, 2008).

7.2.7 Xquery Update dodatek

XQuery Update dodatek razširja XQuery jezik in omogoča spreminjanje podatkov v XML dokumentih. XQuery Update dodatek omogoča naslednje operacije (prav tam):

- dodajanje vozlišča,
- brisanje vozlišča,
- spreminjanja lastnosti vozlišča.

Izrazi, ki jih XQuery Update omogoča (W3C, 2008):

- insert (vstavi nič ali več vozlišč),
- delete (zbriše nič ali več vozlišč),
- replace (zamenja vozlišče z nič ali več vozlišči, ali pa zamenja vrednost vozlišča),
- rename (preimenuje ime vozlišča) in
- transform (kopira, popravi in vrne izbrana vozlišča, pri tem ohrani obstoječa vozlišča).

Spodnja slika prikazuje XQuery Update poizvedbo, ki spremeni vrednost atributa OznakaSTR JA v vrednost Komunalni Jasek.

```
update replace
doc('diploma/A_6001_NG_TKPIS.xml')/TKOmrezje/TRObjekti/TRObjekt[@OznakaSTR='JA']/@OznakaSTR
with "Komunalni Jasek"
```

Slika 26 XQuery Update poizvedba

8 POVEZOVANJE XML DOKUMENTOV

Povezovanje XML dokumentov opisujeta dva standarda: XLink (XML Linking Language) in XPointer (XML Pointer Language). Standarda omogočata boljše povezovanje dokumentov, kot to omogoča HTML (Hyper Text Markup Language). Vsak element v XML dokumentu lahko deluje kot povezava. Povezave so lahko dvosmerne, hkrati pa lahko naredimo tudi povezave na spreminjajoče se reference iz dokumentov, ki se ne smejo spreminjati. XLink določa običajen način povezav v XML dokumentih. XPointer predpisuje sintakso za sklicevanje na posamezne dele XML dokumenta (tudi na podatke) in omogoča izdelavo povezav na katerikoli element znotraj XML dokumenta (WC3, 2008).

Na spodnji sliki je prikazana uporaba povezave znotraj SVG dokumenta na kartografski simbol za trasni objekt.

```
<symbol id="SimbolObjekt" overflow="visible" preserveAspectRatio="xMidYMid meet">
        <circle style="stroke:blue; stroke-width:0.3" r="0.5" cy="0" cx="0"/>
 </symbol>

<image height="3000" width="2250" xlink:href="B232461A.jpg" y="-93999.75" x="394250.25"
xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad" preserveAspectRatio="xMidYMid
meet"/>

<use transform="translate(395585.262 -91054.639)" onclick="ObjectClick('OM:
ID:600119236_362DC')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
```

Slika 27 Sklicevanje na XML element

9 SVG

SVG (Scalable Vector Graphics) je standardni zapis za opisovanje 2D-vektorske in delno tudi rastrske grafike v jeziku XML. SVG je potrdila krovna organizacija W3C (World Wide Web Consortium), kot spletni standard, septembra 2001. Omogoča zapis treh različnih tipov objektov (vektorske grafične objekte, sestavljene iz linij ali krivulj, rastrske slike in besedila).

Kot javni standard, ki podpira animacijo, vektorsko in rastrsko grafiko, je uporaben v vseh spletnih aplikacijah z dinamično vsebino, ki zahtevajo visoko kvalitetno grafiko. Možni primeri uporabe so naslednji (Neuman in Winter, 2001):

- splošno oblikovanje spletnih aplikacij,
- tehnične risbe,
- vizualizacija podatkov v znanstvene namene,
- spletno kartiranje in spletne GIS aplikacije, navigacija, sledenje,
- lokacijske storitve (SVG Mobile),
- spletni katalogi in spletne trgovine,
- spletni sistemi za učenje,
- multimedijske vsebine in razvedrilo,
- uporabniški vmesniki spletnih strani in spletnih aplikacij,
- SVG kot prosti standard za izmenjavo slikovnih vsebin med različno grafično programsko opremo in različnimi arhitekturami.

Kmalu po predstavitvi je SVG postal zanimiv tudi za posredovanje prostorskih podatkov v spletnem brskalniku, saj je imel precej prednosti pred večino obstoječih načinov za posredovanje prostorskih podatkov na internetu. V grobem jih lahko razdelimo v dve skupini (Šumrada, 2001):

- uporabnik pošlje zahtevo preko spletnega brskalnika do strežnika, na katerem se v določenem času izvede zahtevana prostorska operacija, nato pa se rezultat pošlje nazaj do uporabnika (metoda on the fly);

- uporabnik izbere določeno operacijo, katere rezultati so bili že predhodno pripravljeni s posebnim programskim orodjem, ki je podatke iz nekega GIS-a izvozilo v obliko, primerno za prikaz na internetu.

9.1 SVG dokument

SVG dokument se začne z prologom, ki lahko vsebuje dodatna atributa encoding in standalone. Atribut standalone določa, odvisnost SVG dokumenta od DTD ali drugega XML dokumenta. Sledi korenski element <svg>, kjer se definirajo lastnosti SVG dokumenta.

Spodnja slika prikazuje prolog in korenski element datoteke index.svg, ki predstavlja rezultat transformacije XML dokumenta kabelske kanalizacije. Korenski element <svg> vsebuje podatke o lastnostih SVG dokumenta.

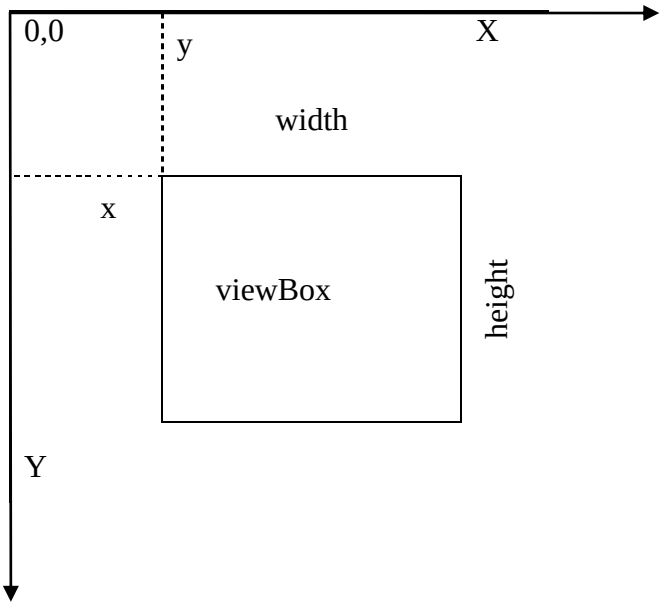
```
<?xml version="1.0" encoding="UTF-8"?>
<?AdobeSVGViewer save="snapshot"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:batik="http://xml.apache.org/batik/ext"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:attrib="http://www.carto.net/attrib"
viewBox="0 0 1024 768" height="100%" onload="init(evt)" zoomAndPan="disable"
   width="100%" preserveAspectRatio="xMidYMid meet" version="1.1"
contentScriptType="text/ecmascript" contentStyleType="text/css"/>
```

Slika 28 Prolog in korenski element SVG dokumenta index.svg

Element <svg> se lahko v SVG dokumentu uporabi večkrat. Pomembna atributa korenskega elementa sta width in height, s katerima določimo velikost SVG dokumenta znotraj brskalnika. Velikost SVG dokumenta se lahko zapiše v različnih merskih enotah (em, px, mm, pt, pc, cm, in). V korenskem elementu se definira tudi koordinatni sistem (izhodišče in velikost) v merskih enotah. Atribut zoomAndPan z vrednostjo disable preprečuje premikanje in povečavo oziroma pomanjšavo grafičnih elementov SVG pregledovalniku.

9.2 Koordinatni sistem in vidno okno

Izhodišče koordinatnega sistema je v SVG dokumentu levo zgoraj. Za prikaz prostorskih podatkov je potrebno prezrcaliti os Y. Za prikaz prostorskih podatkov se lahko uporabijo negativne vrednosti ali pa se uporabi ustrezna translacija v zapisu korenskega elementa.



Slika 29 Koordinatni sistem in vidno okno

Z atributom ViewBox se določa vidno polje - pravokotnik znotraj katerega je prikazana grafična vsebina SVG dokumenta. V kombinaciji z atributom <transform> lahko znotraj SVG dokumenta definiramo več različnih koordinatnih sistemov, ki omogočajo enostavno realizacijo spletnih kart. Atributu ViewBox se določijo naslednje vrednosti (v tem vrstnem redu): x in y vrednost levega zgornjega kota, širina in višina.

```
  <svg viewBox="394919.936 -91676.771 858.391 806.558" id="mainMap" cursor="crosshair"
height="700" width="1000" y="30" x="0" preserveAspectRatio="xMidYMid meet"
zoomAndPan="magnify" version="1.1" contentScriptType="text/ecmascript"
      contentStyleType="text/css"/>
```

Slika 30 Zapis koordinatnega sistema in vidnega okna vsebine interkativne karte

9.3 Osnovni geometrijski elementi

SVG omogoča rabo treh različnih vrst grafičnih elementov: vektorske elemente, rastrske slike in besedila. SVG pregledovalniki omogočajo upodabljanje visoko kvalitetne grafike z uporabo tehniko glajenja robov. Pri upodabljanju se uporablja DOM, ki omogoča hiter dostop do elementov in atributov, in dovoljuje spreminjanje strukture dokumenta znotraj drevesa.

Osnovni geometrijski elementi, ki jih lahko vsebuje SVG dokument so:

- krog,
- štirikotnik,
- elipsa,
- linija,
- polilinija,
- path,
- poligon in
- besedilo.

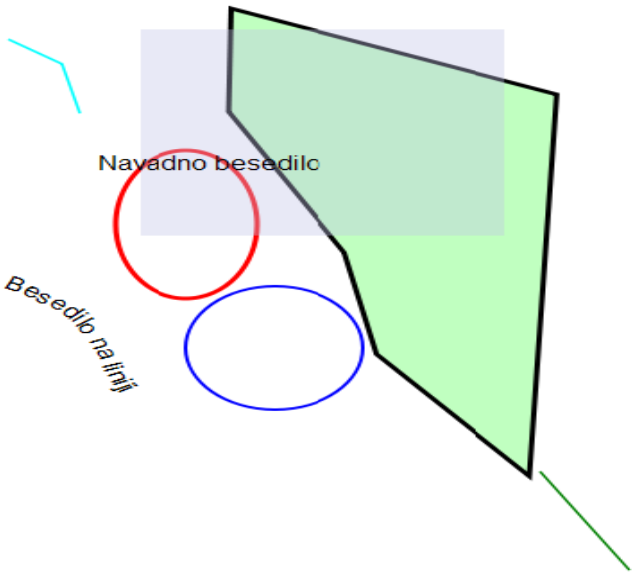
Izmed vseh osnovnih geometrijskih elementov je za kartografijo najbolj uporaben element path, ki lahko vsebuje: linije, kvadratne in kubične Bezierove krivulje, krožne loke.

Vsak objekt ali skupina mora imeti opredeljeno grafično predstavitev. Možnosti je veliko: barva polnila, barva obrobe, velikost in tip linije, barvni prehodi, prosojnost, filtri, vzorci, znaki. SVG pri tem podpira opredelitev takšnih stilov v CSS datoteki, s čimer lahko povsem ločimo zapis koordinatnih vrednosti od grafične predstavitve (Preložnik, 2002).

Uporaba znakov je smiselna iz mnogih razlogov: z zapisom znotraj <defs> sekcije in večkratno uporabo zmanjšamo velikost dokumenta, enostavno urejanje, ohrani se preglednost, saj se znak opredeli samo enkrat na začetku, nato pa se lahko kliče večkrat na poljubnih lokacijah.

Spodnji primer prikazuje zapis SVG dokumenta z osnovnimi geometrijskimi elementi. Na sliki pa je isti dokument kot ga prikaže v brskalnik (vir: Preložnik, 2002:361).

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="500" height="500" xmlns:xlink="http://www.w3.org/1999/xlink">
  <defs>
    <path id="linija" d="M46.5 251.1 C116.788 283.54 87.3688 324.251 128.5 355.1 C163.172 381.104 270.983 375.259 310.5 363.1 C351.568 350.463 383.031 316.799 413.5 289.1" style="fill:none;stroke:rgb(0,0,0);stroke-width:1"/>
  </defs>
  <ellipse cx="150" cy="200" rx="40" ry="60" fill="none" stroke="red" stroke-width="3"/>
  <line x1="350" y1="400" x2="400" y2="480" stroke="green" stroke-width="1.5"/>
  <polyline points="50,50 80,70 90,110" style="fill:white;stroke:cyan;stroke-width:1.6"/>
  <polygon points="344.011,403.9 257.66,305.014 239.554,222.841 174.095,108.635 175.487,25.0696 359.331,94.7075" style="fill:rgb(192,255,192);stroke:rgb(0,0,0);stroke-width:3"/>
  <rect x="124" y="42" width="206" height="167" style="fill:rgb(193,194,229);stroke:rgb(0,0,0);stroke-width:0;opacity:0.37"/>
  <text x="100px" y="156px" style="fill:rgb(0,0,0);font-size:24;font-family:Arial">Navadno besedilo</text>
  <text style="fill:rgb(0,0,0);font-size:24">
    <textPath xlink:href="#linija">Besedilo na liniji</textPath>
  </text>
  <circle cx="200" cy="300" r="50" fill="none" stroke="blue" stroke-width="2"/>
</svg>
```



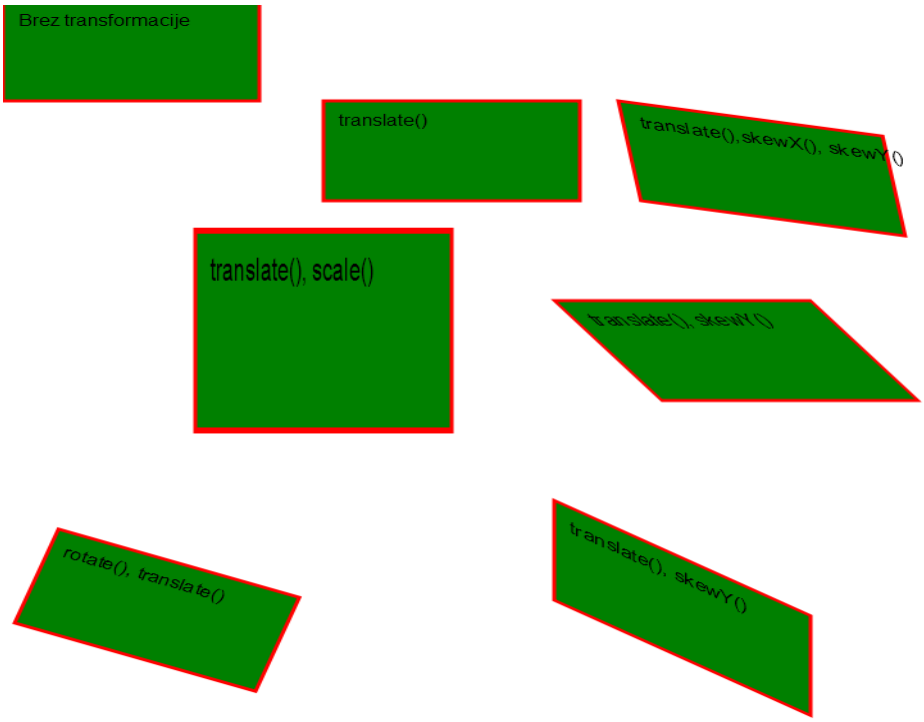
9.4 Grupiranje in transformacija

Element <g> omogoča grupiranje – združevanje elementov v skupine. S tem dosežemo večslojnost, ki je uporabljena v okoljih GIS in CAD. Vsi elementi ene skupine imajo v SVG dokumentu iste parametre in se lahko uporabljajo kot en sloj.

```
<g stroke="red" stroke-width="0.2" stroke-linecap="round">
        <line x1="395166.597" y1="-91463.641" x2="395180.841" y2="-91459.333"/>
        <line x1="395180.841" y1="-91459.333" x2="395181.816" y2="-91457.937"/>
        <line x1="395181.816" y1="-91457.937" x2="395185.001" y2="-91440.708"/>
        <line x1="395185.001" y1="-91440.708" x2="395184.773" y2="-91432.695"/>
      </g>
```

Slika 31 SVG grafični objekti združeni v skupino

Atribut transform omogoča: spremembo merila, rotacijo, premik in spreminjanje nagiba osi. Spodnji primer prikazuje različne možnosti transformacij.



Slika 32 Možnosti transformacij SVG elementa

9.5 Animacija

SVG omogoča animacijo skoraj vsakega elementa, predvsem pa atributov. Sintaksa in specifikacija, ki omogoča animacijo je prevzeta iz SMIL (Sychronized Multimedia Integration Language). Še posebej uporabna je možnost, ki omogoča premikanje grafičnih objektov po elemenu path. Grafični objekt, ki ga premikamo, lahko orientriamo glede na smer trenutnega elementa.

9.6 Interaktivnost-JavaScript

Osnovne interaktivne funkcije so že del nekaterih SVG pregledovalnikov, vendar pa je zelo enostavno doseči se veliko večjo stopnjo interaktivnosti. Z uporabo funkcij JavaScript lahko dosežemo npr. vklop in izklop določenih skupin (slojev), premikanje, povečavo, prehod na začetno sliko, prikaz dodatnih informacij ob kliku na določen objekt. Možno je izdelati poseben grafični uporabniški vmesnik s funkcijami pregledovalnikov GIS-ov, in vse to brez čakanja na nove podatke iz strežnikov (Preložnik, 2002).

JavaScript je nastal v podjetju Netscape kot del njihovega internetnega brskalnika. Zaradi večje uveljavitve na tržišču so ponudili jezik v standardizacijo leta 1996, standard pa je ECMA (European Computer Manufacturers Association) sprejela že naslednje leto. ECMAScript je edini standardizirani skriptni jezik, ki je trenutno največkrat namenjen internetnim brskalnikom. Jedro JavaScripta je implementacija standarda ECMA-262 z nekaterimi dodatki. Standard definira sintakso, nekatere osnovne objekte, njihove metode in atribute, in nekatere globalne funkcije.

JavaScript je interpretiran, interpreter je del internetnega brskalnika. Programi so zapisani tam, kjer jih interpreter kar najlažje najde - v dokumentih ali pa so v dokumentu povezave na datoteke z izvorno kodo. V dokumentu je interpreterju treba povedati, kateri podatki so izvorna koda in katere podatke naj brskalnik ne prikaže kot navadno besedilo. Če so programi zapisani v zunanjih datotekah, se na njih sklicujemo podobno kot na ostale zunanje elemente dokumenta.

Standard je prilagodljiv, saj določa le minimalni nabor funkcionalnosti, ki jih mora skriptni jezik podpirati, dovoljeno pa je poljubno razširjanje – lahko dodajamo lastne objekte, dodatne metode in lastnosti objektov, lahko celo razširjamo sintakso (standardna sintaksa seveda mora ostati nespremenjena).

Uporabniški vmesnik JavaScript aplikacije skrbi za vmesnik med človekom in strojem. Uporablja gradnike, ki so preslikani v programski jezik oziroma vnaprej definirane objekte programskega jezika. Večina uporabniških vmesnikov, napisanih v JavaScriptu je dogodkovno orientiranih. Uporabniški vmesniki od programov zahtevajo, da se odzivajo na različne dogodke.

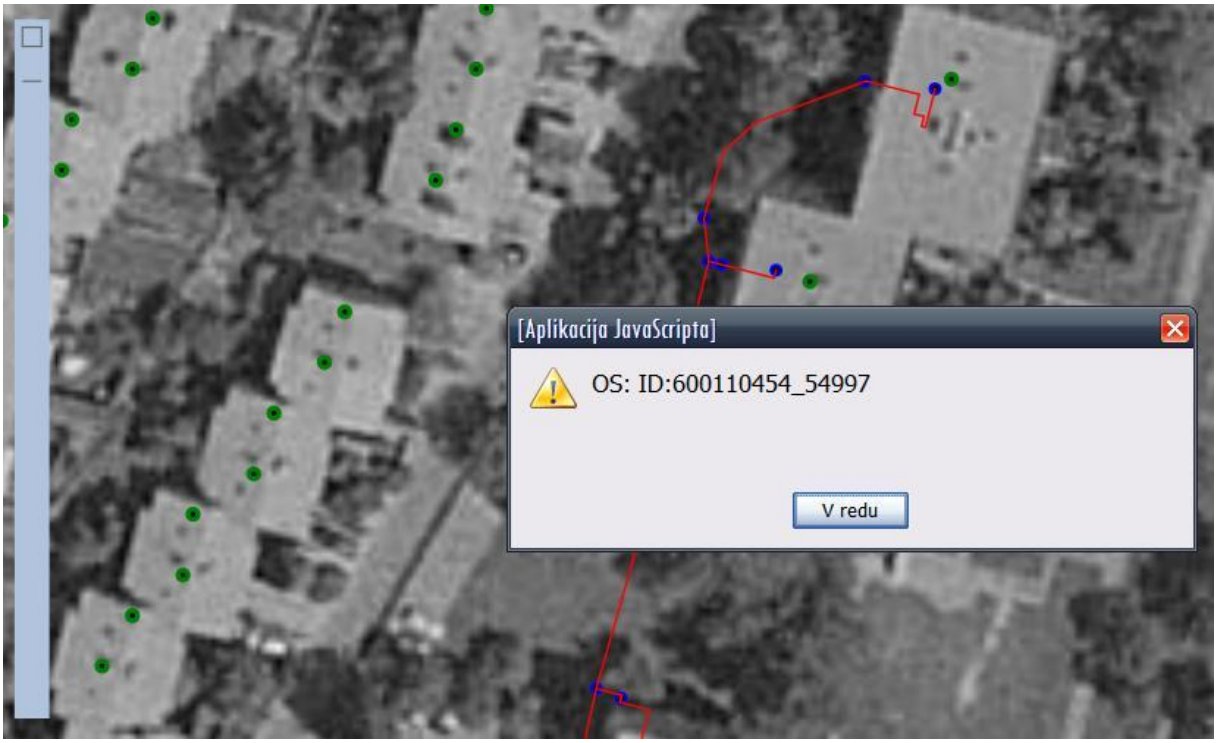
Večina dogodkov je uporabniško proženih (pritisk gumba, premik miške), nekateri pa se sprožijo samodejno (recimo dogodek OnLoad, ki je sprožen, ko je stran naložena). Seveda je vrsta dogodka, ob katerem se bo prožila neka funkcija, odvisna od elementa uporabniškega vmesnika. Nekateri dogodki so takšni, da jih po privzeti vrednosti niti ne zaznavamo (npr. premike miške). Posamezni dogodki so predstavljeni s pomočjo objekta `event`. Objekt vsebuje lastnosti, ki opisujejo neki dogodek in je poslan vsaki funkciji, ki je registrirana za obravnavo dogodkov (Čavlek, 1999).

Spodnji primer prikazuje enostavno funkcijo, ki ob pritisku na gumb v obliki opozorila izpiše informacije.

```
function ObjectClick(text) {
alert(text);
}
```

Slika 33 JavaScript funkcija, ki izpiše opozorilo

Spodnja slika prikazuje obvestilo, ki se prikaže ob kliku na trasni objekt.



Slika 34 Izpis podatkov z uporabo JavaScript

Za grafični uporabniški vmesnik v izdelani spletni karti so uporabljeni elementi spletnega centra za kartografijo (www.carto.net), ki omogočajo naslednje lastnosti:

- povečavo oziroma pomanjšavo,
- premikanje,
- določanje središča karte,
- prehod na začetno sliko,
- prikaz dodatnih informacij ob kliku na objekt,
- vklop in izklop slojev.

V aplikaciji izdelani v tej diplomski nalogi, interaktivnost omogoča navigacijsko ogrodje Map Navigation Tools. Podatki, ki so prirejeni velikosti obravnavanega območja, so zapisani v datoteki MyMapApp.js. Ostale datoteke, ki omogočajo interaktivnost in uporabniški grafični vmesnik, niso bile spremenjene.

Spodnja slika predstavlja del SVG dokumenta index.svg, in predstavlja povezavo na datoteke, ki omogočajo interaktivnost in uporabniški grafični vmesnik.

```
  <script xlink:href="helper_functions.js" type="text/ecmascript" xlink:type="simple"
xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="mapApp.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="timer.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="slider.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="button.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="navigation.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="checkbox_and_radiobutton.js" type="text/ecmascript" xlink:type="simple"
xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="Window.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
  <script xlink:href="myMapApp.js" type="text/ecmascript" xlink:type="simple" xlink:show="other"
xlink:actuate="onLoad"/>
```

Slika 35 Sklicevanje na JavaScript datoteke

9.7 Prednosti in slabosti SVG zapisa

Glavne prednosti zapisa SVG-ja so (Williams in Neuman, 2006):

- standardiziran vektorski zapis podatkov (manjši od ekvivalentne rastrske slike),
- dobra ločljivost in zato primeren tisk,
- enostavno premikanje in povečava po sliki brez vmesnega nalaganja in čakanja,
- enostavna vdelava v HTML-jeve dokumente,
- možnost izvedbe interaktivnosti s pomočjo skripta (JavaScirpt),
- možnost iskanja po besedilnih objektih (te objekte lahko indeksirajo tudi internetni iskalniki zaradi načina zapisa v standardu XML),
- združevanje objektov v skupine (ti objekti ene skupine imeli skupne parametre in se bodo uporabljali kot en sloj),

- izvedba transformacij posameznih objektov ali skupin,
- uporaba barv polnila in obrob objektov, znakov, filtrov, prosojnosti,
- podpora različnih grafičnih objektov, sestavljenih tudi iz krivulj,
- možnost animacij,
- odprta koda, del XML,
- možnost uporabe kompresiranih datotek SVG, končnica SVGZ (Compressed Scalable Vector Graphic File).

Obstaja pa tudi nekaj slabosti pri uporabi SVG-ja za spletno kartografijo (Preložnik, 2002):

- podatki so dostopni vsakomur, ni učinkovitega sistema za njihovo zaščito. Pri tem nastopi problem avtorskih pravic in dovoljenj za uporabo;
- za prikaz datotek SVG je v nekaterih spletnih brskalnik potreben še dodatek za brskalnik;
- SVG ne omogoča topološke kontrole, saj je samo standard za grafiko za razliko od formata GML (Geography Markup Language), ki je standard za zapis in prenos prostorskih podatkov;
- SVG ne omogoča prikaza bolj kompleksnih znakov, recimo dvojne črte za linijske objekte (to se lahko uredi samo z dvakratnim zapisom istega objekta z drugačnimi grafičnimi stili);
- neučinkovito delo z velikimi datotekami – nezmožnost odpiranja posameznih delov dokumenta glede na uporabnikove zahteve.

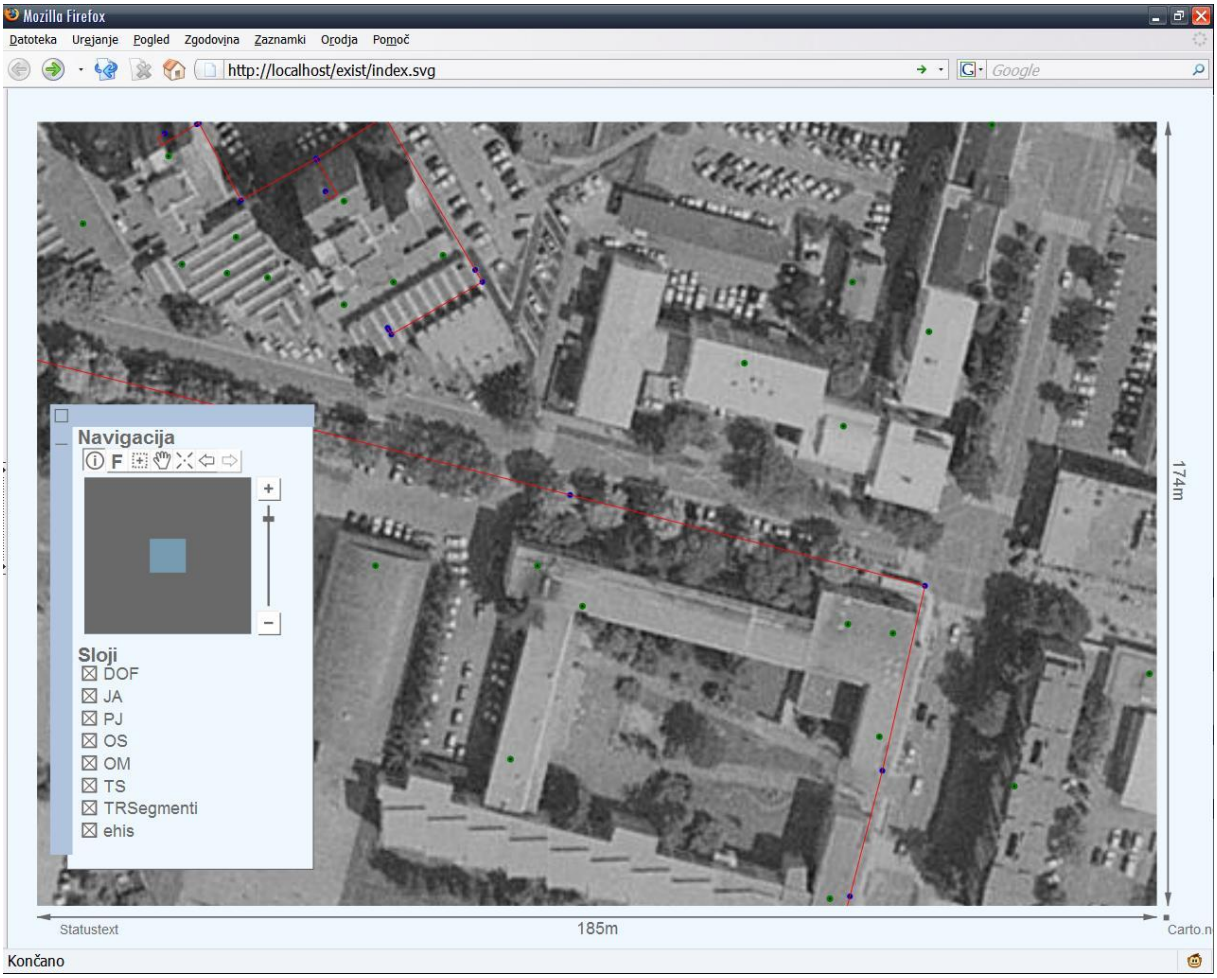
Osnovni problem je pretvorba prostorskih podatkov, ki so shranjeni v različnih CAD (Computer Aided Design) in GIS (Geografski informacijski sistem) zapisih ali v relacijskih bazah podatkov v format SVG. Možnosti pretvorbe podatkov lahko v grobem razdelimo v tri glavne skupine (Pavlicko, 2002):

- neposredne pretvorbe (s programi, moduli ali skriptami, ki neposredno pretvorijo prostorske podatke v SVG zapis),
- posredne pretvorbe podatkov zapisanih v XML formatu z uporabo XSLT jezika,

- posredne pretvorbe ne GIS-ovskih vektorskih formatov (potrebna vmesna pretvorba v standarden vektorski zapis).

9.8 SVG dokument kabelske kanalizacije

Rezultat praktičnega dela diplomske naloge predstavlja SVG dokument index.svg, ki skupaj z navigacijskimi orodji tvori interaktivno karto. Interaktivna karta prikazuje trasne objekte in trasne segmente. Kot kartografska podlaga je uporabljen DOF5 (Digitalni orto foto 1:5000). Dodani pa so tudi točkovni simboli, ki predstavljajo centroide iz EHIŠ (Evidence hišnih številk). Spodnja slika prikazuje SVG dokument index.svg, kot ga prikazuje spletni brskalnik.



Slika 36 Zaslonsko okno interaktivne karte

10 VREDNOTENJE REZULTATOV

Standardiziran XML (eXstensible Markup Language) zapis podatkov omogoča enostavno obdelavo podatkov z računalniki. Standardna orodja, ki omogočajo obdelavo podatkov postopke zelo olajšajo. Prikaz prostorskih podatkov zapisanih v standardizirani obliki, ki ne vključuje predstavitve, je možen z uporabo transformacijskih jezikov, ki preslikujejo podatke v različne besednjake.

Raba XML standarda v spletni kartografiji podpira izdelavo različno oblikovanih predstavitev prostorskih podatkov iz istega metapodatkovnega jedra in omogoča univerzalen standard za opisovanje prostorskih pojavov, ki je neodvisen od ponudnika podatkov in platforme.

Rezultat diplomske naloge – interaktivna spletna karta kabelske kanalizacije, je bila izdelana z uporabo transformacijskega jezika XSLT (eXstensible Stylesheet Language Transformations). Grafični elementi so kot rezultat trasformacije zapisani v SVG (Scalable Vector Graphics) zapisu, ki ga z uporabo vključka upodabljajo grafično nezmogljivi brskalniki. Interaktivnost preko grafičnega uporabniškega orodja omogočajo navigacijska orodja zapisana v JavaScript jeziku. Hitrost upodabljanja prostorskih podatkov na strani uporabnika omejuje rastrski podatkovni sloj DOF5 (digitalni orto foto 1:5000). Ker je izbrano območje, ki vsebuje podatke o kabelski kanalizaciji, relativno majhno, velikost SVG dokumenta pri procesiranju ne predstavlja omejitve.

11 ZAKLJUČEK

Hiter razvoj računalniške in telekomunikacijske tehnike omogoča reševanje problemov, ki se še pred leti niso dali preprosto rešiti. Standarden zapis podatkov v nevtralni obliki prinaša mnoge prednosti kot so prilagodljivost, berljivost in prenosljivost.

Zapisovanje podatkov z uporabo XML standarda je enostavno. Potrebno je le poznavanje sintaktičnih pravil in besednjaka. Kar predstavlja največjo prednost XML standarda, predstavlja hkrati tudi njegovo največjo slabost. Nastalo je mnogo XML jezikov, ki so definirani bodisi v obliki DTD (Document Type Definition) bodisi z XML Schema. Hitra rast novih XML jezikov na področjih, na katerih so že uveljavljeni standardni jeziki, namesto rabe in razširitve obstoječih, povzroča inflacijo standardizacije in jo lahko izniči.

Najbolj obetavna smer rabe XML standarda v spletni kartografiji je povezana z aplikacijami, ki upodabljajo vektorsko grafiko. Razvitih je že kar nekaj standardnih zapisov, ki omogočajo prikazovanje prostorskih podatkov na internetu.

Z uporabo XSLT (eXstensible Stylesheet Language Transformations) jezika ali XQuery (XML Query Language) jezika je preoblikovanje prostorskih podatkov, ki so zapisani v nevtralni obliki brez predstavitvenih elementov, dokaj enostavno opravilo.

Prihodnost rabe SVG zapisa v kartografiji zaradi omejene podpore v spletnih brskalnikih, verjetno predstavlja SVG Mobile zapis, ki se uporablja za upodobitev vektorskih vsebin na mobilnih napravah.

Rezultat diplomske naloge predstavlja interaktivni SVG zapis prostorskih podatkov, in XQuery peskovnik, ki omogoča spreminjanje prostorskih podatkov zapisanih v nevtralni obliki ločeno od predstavitvenih elementov.

VIRI

Uporabljeni viri

Anderson, R. Birbeck, M. Kay, M. 2000. Professional XML. Birmingham, Chicago Wrox Press: 1269 str.

Adams, T. 2005. Using SVG and XSLT to Display Visually Geo-referenced XML. V: Geroimenko, V. Chen C. (ur.). Visualizing Information Using SVG and X3D. NetLibrary Inc: 340 str.

Čavlek, J. 1999. Povezovanje komponent Java s pomočjo jezika Javascript. Diplomska naloga. Maribor, Univerza v Mariboru, Fakulteta za elektrotehniko računalništvo in informatiko: 82 str.

Jurič B. M. 2001. Osnove jezika XML. Maribor: Fakulteta za elektrotehniko, računalništvo in informatiko.

Kovačič, I. 2005. Uvod v podatkovni standard XML. Študijsko gradivo. Ljubljana, Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo.

Lehti P. 2001. Design and Implementation od a Data Manipulation Processor for an XML Cuery Language. Diplomsko delo. Darmstadt, Technische Universitat Darmstadt.

Mohorič, T. Zrnec A. 2001. Transformacijski spletni jeziki.
http://www.ltfe.org/pdf/Transformacijski_spletni_jeziki.pdf.pdf (15.4.2008)

Neuman, A. Winter A.M. 2001. SVG – Scalable Vector Graphics, Ein zukunftiger Eckstein der WWW – Infrastruktur.
http://www.carto.net/papers/svg/articles/paper_ugra_zurich_2001.pdf (28.5.2008)

Neuman, A. Winter M. A. SVG Window Object.
http://www.carto.net/papers/svg/gui/Window/index.shtml (13.3.2008)

Pavlicko, P. 2002. Topographic Maps With SVG.
http://www.svgopen.org/2004/papers/TopographicMapsWithSVG/ (22.5.2008)

Poljšak, T. 2005. Analiza usklajenosti sistemov za upravljanje podatkovnih baz s tehnologijami elektronskega poslovanja. Ljubljana, Univerza v Ljubljani, Ekonomska fakulteta: str 45-49.

Preložnik, U. 2002. SVG kot način za prikazovanje visoko ločljivih interaktivnih spletnih kart. Geodetski vestnik letnik 45, št. 4, 357-363.

Rojko, G. 2005. Smotrnost uporabe XML standarda v PS Mercator d.d. Diplomsko delo. Ljubljana, Univerza v Ljubljani, Ekonomska fakulteta.

Sturm, J. 2000. Developing XML Solutions. Microsoft Press.

Šumrada, R. Prehod od osrednje k porazdeljeni uporabi GIS-ov. Geodetski vestnik letnik 45, št. 4, str. 560-571.

W3C, Exstensible Markup Language (XML),
http://www.w3.org/TR/REC-xml (13.4.2008)

W3C, XML Linking Language (XLink),
http://www.w3.org/TR/xlink/ (13.4.2008)

W3C, XML Path Language (XPath) 2.0,
http://www.w3.org/TR/xpath20/ (13.4.2008)

W3C, XSL Transformations (XSLT),
http://www.w3.org/TR/xslt/ (13.4.2008)

XQuery Update Facility.
http://www.w3.org/TR/xquery-update/ (13.4.2008)

W3Schools. XSL –FO Documents.
http://www.w3schools.com/xslfo/xslfo_documents.asp (16.5.2008)

Zaslavsky, I. 2003. Online Cartography with XML. Maps and the Internet 1. 171-172.

Williams, J. Neuman, A. Navigation Tools for SVG Maps.
http://www.carto.net/papers/svg/navigationTools/ (12.3.2008)

Williams, J. Neuman, A. Manipulating SVG Documents Using ECMAScript (Javascript) and the DOM.
http://www.carto.net/papers/svg/manipulating_svg_with_dom_ecmascript/ (12.3.2008)

Ostali viri

Held, G. Using XSLT to create SVG content
http://www.carto.net/papers/svg/samples/xslt/ (11.3.2008)

Mangano, S. 2006. XSLT Cookbook, Second Edition. Sebastopol, O'Reilly

Froumentin, M. Hardy, V. Using XSLT and SVG together: a survey of case studies.
http://www.svgopen.org/2002/papers/froumentin_hardy__xslt/index.html (17.4.2008)

Jolif, C. Comparsion between XML to SVG Transformation Mechanisms, The GraphML use case.
http://www.svgopen.org/2005/papers/ComparisonXML2SVGTransformationMechanisms/index.html (17.4.2008)

Rindahl, B. Real Time Flood Warning Decision Support Systems Utilizing Client-Side XML/XSLT/SVG.
http://www.svgopen.org/2005/papers/RealTimeFloodWarning/index.html (17.4.2008)

XLink and Xpointer Tutorial
http://www.w3schools.com/xlink/default.asp (15.5.2008)

XPath Tutorial.
http://www.w3schools.com/xpath/default.asp (3.3.2008)

W3Schools, XSLT Tutorial.

http://www.w3schools.com/xsl/default.asp (13.3.2008)

W3C, World Wide Web Consortium, XML Query Use Cases.

http://www.w3.org/TR/xquery-use-cases/ (15.4.2008)

PRILOGE

PRILOGA A: DEL XML DOKUMENTA A _6001_NG_TKPIS.XML

PRILOGA B: VSEBINA DATOTEKE XML2SVG.XSLT

PRILOGA C: DEL SVG DOKUMENTA INDEX.SVG

PRILOGA D: VSEBINA DOKUMENTA MYMAPAPP.JS

Anželak, A. 2008. Interaktivna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

PRILOGA A: DEL XML DOKUMENTA A_6001_NG_TKPIS.XML


```
<TKOmrezje Datum-izvoza="22.02.2008" Datum-kreiranja="22.02.08" Del-kabla="0" FL="6001_PSX NOVA GORICA" Firma-izv="TELEKOM" Izdelal="MALNARIČ"
Izmeril="STERLE" Izvorni-TKI-ID="" Kabel="NG" Kabel-kanal="NG" Kabel-smer="" Komentar="" OE="NG" PID="600111743" Pozic-izv-TKI-papir="20|150" Projekt="ITD-NG
(600111743)" Rok-projekta="22.05.2008" Skrbnik-email="" TRSDelNaziv="" Vrsta-dok="DOKUMENTACIJA KANALIZACIJE" Zacet-smer-izr="" Zap-stev-sh="">
  <TRObjekti>
    <TRObjekt ACID="" ACVRSTA="A" DatumSpr="" Dimenzija="0.000" FL-prip="6001_PSX NOVA GORICA" Kanalizacija-prip="" Lastnik="" LetoIzgradnje="0" Natancnost="2"
OE-prip="NG" OZNAKA="" OznakaSTR="TS" Poz_x="394980.368" Poz_y="91399.208" Poz_z="93.150" PrevObjectHandle="" Projekt="600111743" RELDist="0" RELSmer="0"
Rotation="0" Simb_x="394980.368" Simb_y="91399.208" Simb_z="93.150" Status="1" Status-dok="" TROBJ-TDID="600110454_551F2" TROBJ-TKISID="600110454_551F2"
TipSpremembe="0" VirPodloge="2" VisinaObjekta="0.000"></TRObjekt>
.
.
.
    <TRObjekt ACID="{D1E55C8E-9257-4D91-ACE0-BB3D419D7CF7}" ACVRSTA="A" DatumSpr="" Dimenzija="200.000" FL-prip="6001_PSX NOVA GORICA" Kanalizacija-
prip="NOVA GORICA" Lastnik="" LetoIzgradnje="200" Natancnost="2" OE-prip="NG" OZNAKA="466" OznakaSTR="JA" Poz_x="395412.670" Poz_y="91216.453" Poz_z="95.530"
PrevObjectHandle="" Projekt="600111743" RELDist="0" RELSmer="0" Rotation="1.26438334990603" Simb_x="395412.670" Simb_y="91216.453" Simb_z="95.530" Status="1"
Status-dok="" TROBJ-TDID="600119236_36DC2" TROBJ-TKISID="600119236_36DC2" TipSpremembe="0" VirPodloge="4" VisinaObjekta="180.000"></TRObjekt>
  </TRObjekti>
  <TRSegmenti>
    <TRSegment ACID="" ACTIPLINIJE="5" ACVRSTA="S" DatumSpr="" Dolzina="0.49" FL-prip="" IDTipTrase="1" IZV-TROBJ-TDID="600110454_551F2" IZV-TROBJ-
TKISID="600110454_551F2" Kanalizacija-prip="" Lastnik="" Natancnost="0" NullStatus="0" OE-prip="" PON-TROBJ-TDID="600117370_67206" PON-TROBJ-
TKISID="600117370_67206" Projekt="600111743" SirinaTrase="0.00" Status="0" Status-dok="" Stevilo_cevi="" TRODS-TDID="600110454_551F4" TRODS-
TKISID="600110454_551F4" TipSpremembe="0" VirPodloge="0">
      <SITGEOMETRY VertNumber="3">
        <Coor ID="0">394980.368</Coor>
        <Coor ID="1">91399.208</Coor>
        <Coor ID="2">394980.496</Coor>
        <Coor ID="3">91399.690</Coor>
        <Coor ID="4">394980.496</Coor>
        <Coor ID="5">91399.691</Coor>
      </SITGEOMETRY>
      <SITGEOMETRYZ VertNumber="3">
        <CoorZ ID="0">93.150</CoorZ>
        <CoorZ ID="1">93.150</CoorZ>
        <CoorZ ID="2">95.150</CoorZ>
      </SITGEOMETRYZ>
    </TRSegment>
.
.
.
    <TRSegment ACID="{F61FDD7F-5FEB-445C-8340-76D93442C491}" ACTIPLINIJE="5" ACVRSTA="S" DatumSpr="" Dolzina="54.60" FL-prip="6001_PSX NOVA GORICA"
IDTipTrase="1" IZV-TROBJ-TDID="600119236_359EC" IZV-TROBJ-TKISID="600119236_359EC" Kanalizacija-prip="NOVA GORICA" Lastnik="" Natancnost="2" NullStatus="0"
OE-prip="NG" PON-TROBJ-TDID="600119236_35A62" PON-TROBJ-TKISID="600119236_35A62" Projekt="600111743" SirinaTrase="0.00" Status="1" Status-dok=""
Stevilo_cevi="" TRODS-TDID="600119236_35A65" TRODS-TKISID="600119236_35A65" TipSpremembe="0" VirPodloge="2">
      <SITGEOMETRY VertNumber="2">
        <Coor ID="0">395561.518</Coor>
        <Coor ID="1">91147.116</Coor>
        <Coor ID="2">395547.907</Coor>
        <Coor ID="3">91094.236</Coor>
      </SITGEOMETRY>
      <SITGEOMETRYZ VertNumber="2">
        <CoorZ ID="0">97.440</CoorZ>
        <CoorZ ID="1">97.390</CoorZ>
      </SITGEOMETRYZ>
    </TRSegment>
    <RAZDALJA>191347</RAZDALJA>
  </TRSegmenti>
</TKOmrezje>
```

PRILOGA B: VSEBINA DATOTEKE XML2SVG.XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet version="2.0" xmlns="http://www.w3.org/2000/svg" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:fn="http://www.w3.org/2005/xpath-functions" xmlns:functx="http://www.functx.com"
            xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xsl:output method="xml" indent="yes" omit-xml-declaration="no" encoding="UTF-8" media-type="image/svg+xml"/>
   <xsl:template match="TKOmrezje">
      <xsl:text disable-output-escaping="yes"><![CDATA[<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" [
      <!ATTLIST svg
   xmlns:attrib CDATA #IMPLIED
   xmlns:batik CDATA #IMPLIED
>
<!ATTLIST g
   batik:static CDATA #IMPLIED
>
<!ATTLIST image
   batik:static CDATA #IMPLIED
>
<!ATTLIST path
   batik:static CDATA #IMPLIED
>
]>
]]></xsl:text>
      <xsl:variable name="du" select="number(1000)"/>
      <xsl:variable name="minY" select="min(//TRObjekt/@Poz_y)-50"/>
      <xsl:variable name="maxX" select="max(//TRObjekt/@Poz_x)+50"/>
      <xsl:variable name="minX" select="(min(//TRObjekt/@Poz_x))-50"/>
      <xsl:variable name="maxY" select="(max(//TRObjekt/@Poz_y))+50"/>
      <xsl:variable name="dx" select="format-number (($maxX - $minX)*1.2, '#.000') "/>
      <xsl:variable name="dy" select="format-number (($maxY - $minY)*1.2, '#.000') "/>
      <xsl:processing-instruction name="AdobeSVGViewer">save="snapshot"</xsl:processing-instruction>
      <svg width="100%" height="100%" viewBox="0 0 1024 768" onload="init(evt)" zoomAndPan="disable" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:attrib="http://www.carto.net/attrib"
          xmlns:batik="http://xml.apache.org/batik/ext">
         <script type="text/ecmascript" xlink:href="helper_functions.js"/>
         <script type="text/ecmascript" xlink:href="mapApp.js"/>
         <script type="text/ecmascript" xlink:href="timer.js"/>
         <script type="text/ecmascript" xlink:href="slider.js"/>
         <script type="text/ecmascript" xlink:href="button.js"/>
         <script type="text/ecmascript" xlink:href="navigation.js"/>
         <script type="text/ecmascript" xlink:href="checkbox_and_radiobutton.js"/>
         <script type="text/ecmascript" xlink:href="Window.js"/>
         <script type="text/ecmascript" xlink:href="myMapApp.js"/>
         <defs>
            <symbol id="buttonRect" overflow="visible">
               <rect x="-70" y="-10" width="140" height="20" fill="gainsboro" stroke="dimgray" stroke-width="1"/>
            </symbol>
            <symbol id="checkBoxRect" overflow="visible">
               <rect x="-6" y="-6" width="12" height="12" fill="white" stroke="dimgray" stroke-width="1.5"/>
            </symbol>
            <symbol id="checkBoxCross" overflow="visible" fill="none" stroke="dimgray" stroke-width="1" pointer-events="none">
               <line x1="-5" y1="-5" x2="5" y2="5"/>
               <line x1="-5" y1="5" x2="5" y2="-5"/>
            </symbol>
            <symbol id="magnifyerFull" overflow="visible">
               <text y="7.5" font-family="Arial,Helvetica" fill="dimgray" font-size="18" font-weight="bold" text-anchor="middle" pointer-events="none">F</text>
            </symbol>
            <symbol id="infoBut" overflow="visible">
               <circle fill="none" stroke="dimgray" stroke-width="1.5" r="7.5"/>
               <text y="5" font-family="Arial,Helvetica" font-size="13" font-weight="bold" fill="dimgray" text-anchor="middle" pointer-events="none">i</text>
            </symbol>
            <symbol id="magnifyerManual" overflow="visible" fill="none" stroke="dimgray" stroke-width="1.5">
               <rect x="-6" y="-6" width="12" height="12" stroke-dasharray="1.5,1.5"/>
               <line x1="-3" y1="0" x2="3" y2="0"/>
               <line x1="0" y1="-3" x2="0" y2="3"/>
            </symbol>
            <symbol id="magnifyerZoomIn" overflow="visible" fill="none" stroke="dimgray" stroke-width="2">
               <line x1="-4" y1="0" x2="4" y2="0"/>
               <line x1="0" y1="-4" x2="0" y2="4"/>
            </symbol>
            <symbol id="magnifyerZoomOut" overflow="visible">
               <line x1="-4" y1="0" x2="4" y2="0" fill="none" stroke="dimgray" stroke-width="2"/>
            </symbol>
            <symbol id="symbPan" overflow="visible">
               <path transform="scale(1.2)" fill="none" stroke="dimgray" stroke-width="1"
                  d="M-2 6 C -2.2 2.5 -8.0 -0 -5.7 -1.9 C -4.3 -2.5 -3.3 -0.5 -2.5 0.7 C -3.2 -2.1 -5.5 -5.2 -3.6 -5.8 C -2.1 -6.3 -1.6 -3.6 -1.1 -1.9 C -0.9 -4.2 -1.6 -6.4 -0.2 -6.6 C 1.4 -6.8 0.9 -3
1.1 -1.9 C 1.5 -3.5 1.2 -6.1 2.5 -6.1 C 3.9 -6.1 3.5 -3.2 3.6 -1.6 C 4 -2.9 4.1 -4.3 5.3 -4.4 C 7.3 -3.5 4 2.2 3 6z"/>
            </symbol>
            <symbol id="symbArrow" overflow="visible">
               <polyline fill="none" stroke="dimgray" stroke-width="1" points="-3,-6 3,-6 3,1 5,1 0,7 -5,1 -3,1 -3,-5"/>
            </symbol>
            <symbol id="symbArrowLeft" overflow="visible">
               <use xlink:href="#symbArrow" transform="rotate(90)"/>
            </symbol>
            <symbol id="symbArrowRight" overflow="visible">
               <use xlink:href="#symbArrow" transform="rotate(-90)"/>
            </symbol>
            <symbol id="symbRecenter" overflow="visible">
               <circle fill="dimgray" cx="0" cy="0" r="1" pointer-events="none"/>
```

```
            <g fill="none" stroke="dimgray" stroke-width="1.5" pointer-events="none">
              <line x1="-7" y1="-7" x2="-3" y2="-3"/>
              <line x1="7" y1="7" x2="3" y2="3"/>
              <line x1="-7" y1="7" x2="-3" y2="3"/>
              <line x1="7" y1="-7" x2="3" y2="-3"/>
            </g>
          </symbol>
          <symbol id="sliderSymbol" overflow="visible" pointer-events="none">
            <line x1="0" y1="-5" x2="0" y2="5" fill="none" stroke="dimgray" stroke-width="5"/>
          </symbol>
          <symbol id="myDragCrossSymbol" overflow="visible" stroke-width="2000" fill="none" stroke="darkblue" pointer-events="none">
            <line x1="-7000" y1="0" x2="-2500" y2="0"/>
            <line x1="7000" y1="0" x2="2500" y2="0"/>
            <line x1="0" y1="-3300" x2="0" y2="-7800"/>
            <line x1="0" y1="3300" x2="0" y2="7800"/>
          </symbol>
          <marker id="myStartArrow" overflow="visible" orient="auto">
            <polyline fill="dimgray" points="-0.5,0 8,-2 8,2"/>
          </marker>
          <marker id="myEndArrow" overflow="visible" orient="auto">
            <polyline fill="dimgray" points="0.5,0 -8,-2 -8,2"/>
          </marker>
          <symbol id="SimbolObjekt" overflow="visible">
            <circle cx="0" cy="0" r="0.5" style="stroke:blue; stroke-width:0.3"/>
          </symbol>
        </defs>
        <rect x="-500" y="-500" width="3000" height="3000" stroke="none" fill="aliceblue"/>
        <svg id="mainMap" x="0" y="30" viewBox="{$minX} -{$maxY} {$dx} {$dy}" width="1000" height="700" cursor="crosshair">
          <g id="mainMapGroup" transform="translate(0,0)">
            <g id="DOF" pointer-events="none">
              <image height="3000" width="2250" xlink:href="B232461A.jpg" pointer-events="none" y="-93999.75" x="394250.25"/>
              <image height="3000" width="2250" xlink:href="B233461A.jpg" pointer-events="none" y="-90999.75" x="394250.25"/>
            </g>
            <xsl:apply-templates select="TRObjekti"/>
            <g id="TRSegmenti" stroke="red" stroke-width="0.2" stroke-linecap="round">
              <xsl:apply-templates select="TRSegmenti"/>
            </g>
            <g id="ehis" style="stroke:green; stroke-width:0.5">
              <xsl:variable name="dok" select="document('hs_opis_ng.xml')"/>
              <xsl:for-each select="$dok/dataroot/hs_opis">
                <xsl:if test="(functx:between-inclusive(x, $minX, $maxX)) and (functx:between-inclusive(y, $minY, $maxY))">
                  <circle cx="{x}" cy="-{y}" r="0.5" onclick="ObjectClick('{na_ime}, {ul_ime},{hshd}')"/>
                </xsl:if>
              </xsl:for-each>
            </g>
          </g>
        </svg>
        <g>
          <g>
            <line stroke="dimgray" stroke-width="1.5" marker-start="url(#myStartArrow)" marker-end="url(#myEndArrow)" x1="1010" y1="30" x2="1010" y2="730"/>
            <rect fill="aliceblue" x="1015" y="330" width="12" height="70"/>
            <text id="myScaleTextH" font-family="Arial,Helvetica" fill="dimgray" font-size="15" text-anchor="middle" transform="translate(1015,350),rotate(90)" pointer-
events="none">573 m</text>
            <line stroke="dimgray" stroke-width="1.5" marker-start="url(#myStartArrow)" marker-end="url(#myEndArrow)" x1="0" y1="740" x2="1000" y2="740"/>
            <rect fill="aliceblue" x="370" y="755" width="70" height="12"/>
            <text id="myScaleTextW" font-family="Arial,Helvetica" fill="dimgray" font-size="15" text-anchor="middle" transform="translate(500,755)" pointer-events="none">616
m</text>
          </g>
        </g>
        <g id="NavWindow">
          <svg viewBox="394919.936 -91676.771 858.391 806.558" id="referenceMap" height="140" width="200" y="50" x="-10">
            <rect stroke="none" height="806.558" fill="dimgray" width="858.391" y="-91676.771" x="394919.936"/>
          </svg>
          <g id="mapZoomSlider"/>
          <text fill="dimgray" font-family="Arial,Helvetica" font-weight="bold" font-size="18" x="10" y="20" pointer-events="none">Navigacija</text>
          <g id="zoomIn" cursor="pointer"/>
          <g id="zoomOut" cursor="pointer"/>
          <g id="infoButton" cursor="pointer"/>
          <g id="zoomFull" cursor="pointer"/>
          <g id="zoomManual" cursor="pointer"/>
          <g id="panManual" cursor="pointer"/>
          <g id="recenterMap" cursor="pointer"/>
          <g id="backwardExtent" cursor="pointer"/>
          <g id="forwardExtent" cursor="pointer"/>
          <g id="highlightMap" cursor="pointer"/>
          <g transform="translate(20 225)" id="checkBoxes">
            <text font-family="Arial,Helvetica" fill="dimgray" font-size="18" font-weight="bold" x="-10" y="-10" pointer-events="none">Sloji</text>
          </g>
        </g>
        <text id="statusText" font-family="Arial,Helvetica" fill="dimgray" font-size="12px" x="20" y="755">Statustext</text>
        <rect fill="dimgray" x="1006" y="738" width="5" height="5" onclick="showExtent()"/>
        <a xlink:href="http://www.carto.net/" target="_top">
          <text font-family="Arial,Helvetica" fill="dimgray" font-size="12px" x="1010" y="755">Carto.net</text>
        </a>
      </svg>
    </xsl:template>
    <xsl:template match="TRObjekti">
      <g id="JA">
        <xsl:apply-templates select="TRObjekt[@OznakaSTR='JA']"/>
      </g>
      <g id="PJ">
        <xsl:apply-templates select="TRObjekt[@OznakaSTR='PJ']"/>
      </g>
      <g id="OS">
        <xsl:apply-templates select="TRObjekt[@OznakaSTR='OS']"/>
```

```
      </g>
      <g id="OM">
        <xsl:apply-templates select="TRObjekt[@OznakaSTR='OM']"/>
      </g>
      <g id="TS">
        <xsl:apply-templates select="TRObjekt[@OznakaSTR='TS']"/>
      </g>
    </xsl:template>
    <xsl:template match="TRObjekt">
      <xsl:for-each select=".">
        <use xlink:href="#SimbolObjekt" transform="translate({@Poz_x} -{@Poz_y})" onclick="ObjectClick('{@OznakaSTR}:{@OZNAKA} ID:{@TROBJ-TDID}')"/>
      </xsl:for-each>
    </xsl:template>
    <xsl:template match="TRSegmenti">
      <xsl:for-each select="TRSegment">
        <xsl:for-each-group select="SITGEOMETRY" group-by="Coor/@ID">
          <xsl:variable name="max" select="Coor/@ID"/>
          <xsl:if test="(current-grouping-key() mod 2=0) and (position() &lt; $max) ">
            <line x1="{Coor[@ID=current-grouping-key()]}" y1="-{Coor[@ID=current-grouping-key()+1]}" x2="{Coor[@ID=current-grouping-key()+2]}" y2="-{Coor[@ID=current-
grouping-key()+3]}"/>
          </xsl:if>
        </xsl:for-each-group>
      </xsl:for-each>
    </xsl:template>
    <xsl:function name="functx:between-inclusive" as="xs:boolean">
      <xsl:param name="value" as="xs:anyAtomicType?"/>
      <xsl:param name="minValue" as="xs:anyAtomicType"/>
      <xsl:param name="maxValue" as="xs:anyAtomicType"/>
      <xsl:sequence select=" $value &gt;= $minValue and $value &lt;= $maxValue "/>
    </xsl:function>
</xsl:stylesheet>
```

Anželak, A. 2008. Interaktivna karta kabelske kanalizacije
Dipl. nal. – VSŠ. Ljubljana, UL, FGG, Oddelek za geodezijo, Prostorska smer.

PRILOGA C: DEL SVG DOKUMENTA INDEX.SVG

```
<?xml version="1.0" encoding="UTF-8"?>
<?AdobeSVGViewer save="snapshot"?>
<svg xmlns="http://www.w3.org/2000/svg" xmlns:batik="http://xml.apache.org/batik/ext" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:attrib="http://www.carto.net/attrib"
viewBox="0 0 1024 768" height="100%" onload="init(evt)" zoomAndPan="disable"
    width="100%" preserveAspectRatio="xMidYMid meet" version="1.1" contentScriptType="text/ecmascript" contentStyleType="text/css">
  <script xlink:href="helper_functions.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="mapApp.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="timer.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="slider.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="button.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="navigation.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="checkbox_and_radiobutton.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="Window.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <script xlink:href="myMapApp.js" type="text/ecmascript" xlink:type="simple" xlink:show="other" xlink:actuate="onLoad"/>
  <defs>
    <symbol id="buttonRect" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <rect stroke="dimgray" height="20" fill="gainsboro" width="140" y="-10" stroke-width="1" x="-70"/>
    </symbol>
    <symbol id="checkBoxRect" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <rect stroke="dimgray" height="12" fill="white" width="12" y="-6" stroke-width="1.5" x="-6"/>
    </symbol>
    <symbol id="checkBoxCross" stroke="dimgray" fill="none" overflow="visible" pointer-events="none" stroke-width="1" preserveAspectRatio="xMidYMid meet">
      <line y1="-5" y2="5" x2="5" x1="-5"/>
      <line y1="5" y2="-5" x2="5" x1="-5"/>
    </symbol>
    <symbol id="magnifyerFull" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <text font-size="18" fill="dimgray" text-anchor="middle" font-family="Arial,Helvetica" font-weight="bold" pointer-events="none" y="7.5">F</text>
    </symbol>
    <symbol id="infoBut" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <circle stroke="dimgray" r="7.5" fill="none" stroke-width="1.5"/>
      <text fill="dimgray" font-size="13" text-anchor="middle" font-family="Arial,Helvetica" font-weight="bold" pointer-events="none" y="5">i</text>
    </symbol>
    <symbol id="magnifyerManual" stroke="dimgray" fill="none" overflow="visible" stroke-width="1.5" preserveAspectRatio="xMidYMid meet">
      <rect height="12" width="12" stroke-dasharray="1.5,1.5" y="-6" x="-6"/>
      <line y1="0" y2="0" x2="3" x1="-3"/>
      <line y1="-3" y2="3" x2="0" x1="0"/>
    </symbol>
    <symbol id="magnifyerZoomIn" stroke="dimgray" fill="none" overflow="visible" stroke-width="2" preserveAspectRatio="xMidYMid meet">
      <line y1="0" y2="0" x2="4" x1="-4"/>
      <line y1="-4" y2="4" x2="0" x1="0"/>
    </symbol>
    <symbol id="magnifyerZoomOut" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <line stroke="dimgray" fill="none" y1="0" y2="0" x2="4" stroke-width="2" x1="-4"/>
    </symbol>
    <symbol id="symbPan" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <path stroke="dimgray" transform="scale(1.2)"
          d="M-2 6 C -2.2 2.5 -8.0 -0 -5.7 -1.9 C -4.3 -2.5 -3.3 -0.5 -2.5 0.7 C -3.2 -2.1 -5.5 -5.2 -3.6 -5.8 C -2.1 -6.3 -1.6 -3.6 -1.1 -1.9 C -0.9 -4.2 -1.6 -6.4 -0.2 -6.6 C 1.4 -6.8 0.9 -3 1.1 -1.9 C 1.5 -3.5 1.2 -6.1 2.5 -6.1 C 3.9 -6.1 3.5 -3.2 3.6 -1.6 C 4 -2.9 4.1 -4.3 5.3 -4.4 C 7.3 -3.5 4 2.2 3 6z"
          fill="none" stroke-width="1"/>
    </symbol>
    <symbol id="symbArrow" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <polyline stroke="dimgray" fill="none" points="-3,-6 3,-6 3,1 5,1 0,7 -5,1 -3,1 -3,-5" stroke-width="1"/>
    </symbol>
    <symbol id="symbArrowLeft" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <use transform="rotate(90)" xlink:href="#symbArrow" xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"/>
    </symbol>
    <symbol id="symbArrowRight" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <use transform="rotate(-90)" xlink:href="#symbArrow" xlink:type="simple" xlink:show="embed" xlink:actuate="onLoad"/>
    </symbol>
    <symbol id="symbRecenter" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <circle r="1" fill="dimgray" cy="0" cx="0" pointer-events="none"/>
      <g stroke="dimgray" fill="none" pointer-events="none" stroke-width="1.5">
        <line y1="-7" y2="-3" x2="-3" x1="-7"/>
        <line y1="7" y2="3" x2="3" x1="7"/>
        <line y1="7" y2="3" x2="-3" x1="-7"/>
        <line y1="-7" y2="-3" x2="3" x1="7"/>
      </g>
    </symbol>
    <symbol id="sliderSymbol" overflow="visible" pointer-events="none" preserveAspectRatio="xMidYMid meet">
      <line stroke="dimgray" fill="none" y1="-5" y2="5" x2="0" stroke-width="5" x1="0"/>
    </symbol>
    <symbol id="myDragCrossSymbol" stroke="darkblue" fill="none" overflow="visible" pointer-events="none" stroke-width="2000" preserveAspectRatio="xMidYMid meet">
      <line y1="0" y2="0" x2="-2500" x1="-7000"/>
      <line y1="0" y2="0" x2="2500" x1="7000"/>
      <line y1="-3300" y2="-7800" x2="0" x1="0"/>
      <line y1="3300" y2="7800" x2="0" x1="0"/>
    </symbol>
    <marker id="myStartArrow" orient="auto" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <polyline fill="dimgray" points="-0.5,0 8,-2 8,2"/>
    </marker>
    <marker id="myEndArrow" orient="auto" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <polyline fill="dimgray" points="0.5,0 -8,-2 -8,2"/>
    </marker>
    <symbol id="SimbolObjekt" overflow="visible" preserveAspectRatio="xMidYMid meet">
      <circle style="stroke:blue; stroke-width:0.3" r="0.5" cy="0" cx="0"/>
    </symbol>
  </defs>
  <rect stroke="none" height="3000" fill="aliceblue" width="3000" y="-500" x="-500"/>
```

```
<svg viewBox="394919.936 -91676.771 858.391 806.558" id="mainMap" cursor="crosshair" height="700" width="1000" y="30" x="0" preserveAspectRatio="xMidYMid meet"
zoomAndPan="magnify" version="1.1" contentScriptType="text/ecmascript"
    contentStyleType="text/css">
  <g id="mainMapGroup" transform="translate(0,0)">
    <g id="DOF" pointer-events="none">
      <image height="3000" width="2250" xlink:href="B232461A.jpg" pointer-events="none" y="-93999.75" x="394250.25" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad" preserveAspectRatio="xMidYMid meet"/>
      <image height="3000" width="2250" xlink:href="B233461A.jpg" pointer-events="none" y="-90999.75" x="394250.25" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad" preserveAspectRatio="xMidYMid meet"/>
    </g>
    <g id="JA">
      <use transform="translate(395220.207 -91450.564)" onclick="ObjectClick('JA:323 ID:600110454_54903')" xlink:href="#SimbolObjekt" xlink:type="simple"
xlink:show="embed" xlink:actuate="onLoad"/>
.
.
.
      <use transform="translate(395561.518 -91147.116)" onclick="ObjectClick('JA:102 ID:600119236_359EC')" xlink:href="#SimbolObjekt" xlink:type="simple"
xlink:show="embed" xlink:actuate="onLoad"/>
    </g>
    <g id="PJ">
      <use transform="translate(395303.533 -91313.373)" onclick="ObjectClick('PJ: ID:600117370_62AD3')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
.
.
.
      <use transform="translate(395251.234 -91376.463)" onclick="ObjectClick('PJ:313 ID:600117370_61F41')" xlink:href="#SimbolObjekt" xlink:type="simple"
xlink:show="embed" xlink:actuate="onLoad"/>
    </g>
    <g id="OS">
      <use transform="translate(395161.669 -91608.119)" onclick="ObjectClick('OS: ID:600110454_54997')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
.
.
.
      <use transform="translate(394980.496 -91399.691)" onclick="ObjectClick('OS: ID:600117370_67206')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
    </g>
    <g id="OM">
      <use transform="translate(395585.262 -91054.639)" onclick="ObjectClick('OM: ID:600119236_362DC')" xlink:href="#SimbolObjekt" xlink:type="simple"
xlink:show="embed" xlink:actuate="onLoad"/>
    </g>
    <g id="TS">
      <use transform="translate(394980.368 -91399.208)" onclick="ObjectClick('TS: ID:600110454_551F2')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
.
.
.
      <use transform="translate(395155.010 -91608.998)" onclick="ObjectClick('TS: ID:600110454_5494D')" xlink:href="#SimbolObjekt" xlink:type="simple" xlink:show="embed"
xlink:actuate="onLoad"/>
    </g>
    <g id="TRSegmenti" stroke="red" stroke-linecap="round" stroke-width="0.2">
      <line y1="-91399.690" y2="-91399.691" x2="394980.496" x1="394980.496"/>
      .
      .
      .
      <line y1="-91094.236" y2="-91063.815" x2="395540.020" x1="395547.907"/>
    </g>
    <g id="ehis" style="stroke:green; stroke-width:0.5">
      <circle r="0.5" onclick="ObjectClick('NOVA GORICA, BAZOVIŠKA ULICA, 8 ')" cy="-91199" cx="394980"/>
      .
      .
      .
      <circle r="0.5" onclick="ObjectClick('NOVA GORICA, BEVKOV TRG, 8 ')" cy="-91280" cx="395490"/>
    </g>
  </g>
</svg>
<g>
  <g>
    <line stroke="dimgray" y1="30" y2="730" marker-end="url(#myEndArrow)" x2="1010" marker-start="url(#myStartArrow)" x1="1010" stroke-width="1.5"/>
    <rect height="70" width="12" fill="aliceblue" y="330" x="1015"/>
    <text id="myScaleTextH" transform="translate(1015,350),rotate(90)" font-size="15" fill="dimgray" text-anchor="middle" font-family="Arial,Helvetica" pointer-
events="none">573 m</text>
    <line stroke="dimgray" y1="740" y2="740" marker-end="url(#myEndArrow)" x2="1000" marker-start="url(#myStartArrow)" x1="0" stroke-width="1.5"/>
    <rect height="12" width="70" fill="aliceblue" y="755" x="370"/>
    <text id="myScaleTextW" transform="translate(500,755)" font-size="15" fill="dimgray" text-anchor="middle" font-family="Arial,Helvetica" pointer-events="none">616 m</text>
  </g>
</g>
<g id="NavWindow">
  <svg id="referenceMap" viewBox="394919.936 -91676.771 858.391 806.558" height="140" width="200" y="50" x="-10" preserveAspectRatio="xMidYMid meet"
zoomAndPan="magnify" version="1.1" contentScriptType="text/ecmascript" contentStyleType="text/css">
    <rect stroke="none" height="806.558" width="858.391" fill="dimgray" y="-91676.771" x="394919.936"/>
  </svg>
  <g id="mapZoomSlider"/>
  <text font-size="18" fill="dimgray" font-family="Arial,Helvetica" font-weight="bold" pointer-events="none" y="20" x="10">Navigacija</text>
  <g id="zoomIn" cursor="pointer"/>
  <g id="zoomOut" cursor="pointer"/>
  <g id="infoButton" cursor="pointer"/>
  <g id="zoomFull" cursor="pointer"/>
  <g id="zoomManual" cursor="pointer"/>
  <g id="panManual" cursor="pointer"/>
  <g id="recenterMap" cursor="pointer"/>
  <g id="backwardExtent" cursor="pointer"/>
  <g id="forwardExtent" cursor="pointer"/>
  <g id="highlightMap" cursor="pointer"/>
  <g id="checkBoxes" transform="translate(20 225)">
```

```
        <text font-size="18" fill="dimgray" font-family="Arial,Helvetica" font-weight="bold" pointer-events="none" y="-10" x="-10">Sloji</text>
      </g>
    </g>
    <text id="statusText" font-size="12px" fill="dimgray" font-family="Arial,Helvetica" y="755" x="20">Statustext</text>
    <rect height="5" width="5" fill="dimgray" onclick="showExtent()" y="738" x="1006"/>
    <a target="_top" xlink:href="http://www.carto.net/" xlink:type="simple" xlink:show="replace" xlink:actuate="onRequest">
      <text font-size="12px" fill="dimgray" font-family="Arial,Helvetica" y="755" x="1010">Carto.net</text>
    </a>
</svg>
```

PRILOGA D: VSEBINA DOKUMENTY MYMAPAPP.JS

```
//global variables for map application and navigation
var myMapApp = new mapApp(false,undefined);
var myMainMap;

function init(evt) {
  //start a new window array in myMapApp
  myMapApp.Windows = new Array();
  //first a few styles
  var winPlaceholderStyles = {"fill":"none","stroke":"dimgray","stroke-width":1.5};
  var windowStyles = {"fill":"aliceblue","stroke":"dimgray","stroke-width":1};
  var titlebarStyles = {"fill":"gainsboro","stroke":"dimgray","stroke-width":1};
  var statusbarStyles = {"fill":"aliceblue","stroke":"dimgray","stroke-width":1};
  var titletextStyles = {"font-family":"Arial,Helvetica","font-size":14,"fill":"dimgray"};
  var statustextStyles = {"font-family":"Arial,Helvetica","font-size":10,"fill":"dimgray"};
  var buttonStyles = {"fill":"none","stroke":"dimgray","stroke-width":1};
  var titlebarHeight = 17;
  var statusbarHeight = 13;
  myMapApp.Windows["navWindow"] = new
Window("navWindow","Windows",220,400,800,85,true,0,80,1024,700,true,winPlaceholderStyles,windowStyles,3,false,false,"","",false,true,true,titlebarStyles,ti
tlebarHeight,statusbarStyles,statusbarHeight,titletextStyles,statustextStyles,buttonStyles,buttonTextChange);
  myMapApp.Windows["navWindow"].appendContent("NavWindow",true);
 //dynamic layer array that allow loading from database
  var dynamicLayers = new Array();
  //initialize digiLayers (layers that allow digitizing)
  var digiLayers = new Array();
  //define some styles for the map object
  var zoomRectStyles = {"fill":"none","stroke":"crimson","stroke-width":0.002,"stroke-dasharray":"0.012,0.002"};
  var highlightStyles = {"stroke":"crimson","stroke-width":0.002};
  var dragRectStyles = {"fill":"lightskyblue","fill-opacity":0.5};
  //initialize myMainMap object, you need to adopt the parameters here
  myMainMap = new
map("mainMap",858.391,80,0.6,0,2170,"m",1,false,"coordY","coordX",dynamicLayers,digiLayers,"",zoomRectStyles,highlightStyles,dragRectStyles,"reference
Map","myDragCrossSymbol",4750);
  //create zoom slider
  //zoom slider styles
  var sliderStyles={"stroke":"dimgray","stroke-width":2};
  myMapApp.zoomSlider = new
slider("mapZoomSlider","mapZoomSlider",180,75,myMainMap.minWidth,180,165,myMainMap.maxWidth,myMainMap.maxWidth,sliderStyles,10,"sliderSym
bol",myMapApp.refMapDragger,true);
  //now initialize buttons
  myMapApp.buttons = new Array();
  //button styles, adopt the style settings to match your needs
  var buttonTextStyles = {"font-family":"Arial,Helvetica","fill":"dimgray","font-size":10};
  var buttonStyles = {"fill":"white"};
  var shadeLightStyles = {"fill":"rgb(235,235,235)"};
  var shadeDarkStyles = {"fill":"dimgray"};
  //button instance creation
  myMapApp.buttons["infoButton"] = new
switchbutton("infoButton","infoButton",zoomImageSwitchButtons,"rect",undefined,"infoBut",15,25,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,shad
eDarkStyles,1);
  myMapApp.buttons["infoButton"].setSwitchValue(true,false);
  //statusChange("Mode: Infomode");
  myMapApp.buttons["zoomFull"] = new
button("zoomFull","zoomFull",zoomImageButtons,"rect",undefined,"magnifyerFull",35,25,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,shadeDarkStyl
es,1);
  myMapApp.buttons["zoomManual"] = new
switchbutton("zoomManual","zoomManual",zoomImageSwitchButtons,"rect",undefined,"magnifyerManual",55,25,20,20,buttonTextStyles,buttonStyles,shadeLi
ghtStyles,shadeDarkStyles,1);
  myMapApp.buttons["panManual"] = new
switchbutton("panManual","panManual",zoomImageSwitchButtons,"rect",undefined,"symbPan",75,25,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,sh
adeDarkStyles,1);
  myMapApp.buttons["recenterMap"] = new
switchbutton("recenterMap","recenterMap",zoomImageSwitchButtons,"rect",undefined,"symbRecenter",95,25,20,20,buttonTextStyles,buttonStyles,shadeLightS
tyles,shadeDarkStyles,1);
  myMapApp.buttons["backwardExtent"] = new
button("backwardExtent","backwardExtent",zoomImageButtons,"rect",undefined,"symbArrowLeft",115,25,20,20,buttonTextStyles,buttonStyles,shadeLightStyle
s,shadeDarkStyles,1);
  myMapApp.buttons["forwardExtent"] = new
button("forwardExtent","forwardExtent",zoomImageButtons,"rect",undefined,"symbArrowRight",135,25,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,
shadeDarkStyles,1);
  myMapApp.buttons["zoomIn"] = new
button("zoomIn","zoomIn",zoomImageButtons,"rect",undefined,"magnifyerZoomIn",170,50,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,shadeDarkSt
yles,1);
  myMapApp.buttons["zoomOut"] = new
button("zoomOut","zoomOut",zoomImageButtons,"rect",undefined,"magnifyerZoomOut",170,170,20,20,buttonTextStyles,buttonStyles,shadeLightStyles,shade
```

```
DarkStyles,1);

        myMainMap.checkButtons();
         //create checkbox array
        myMapApp.checkBoxes = new Array();
        //labeltext styles
        var labeltextStyles = {"font-family":"Arial,Helvetica","fill":"dimgray","font-size":15};
        //create individual checkboxes
        myMapApp.checkBoxes["DOF"] = new
checkBox("DOF","checkBoxes",0,0,"checkBoxRect","checkBoxCross",true,"DOF",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["JA"] = new
checkBox("JA","checkBoxes",0,20,"checkBoxRect","checkBoxCross",true,"JA",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["PJ"] = new
checkBox("PJ","checkBoxes",0,40,"checkBoxRect","checkBoxCross",true,"PJ",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["OS"] = new
checkBox("OS","checkBoxes",0,60,"checkBoxRect","checkBoxCross",true,"OS",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["OM"] = new
checkBox("OM","checkBoxes",0,80,"checkBoxRect","checkBoxCross",true,"OM",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["TS"] = new
checkBox("TS","checkBoxes",0,100,"checkBoxRect","checkBoxCross",true,"TS",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["TRSegmenti"] = new
checkBox("TRSegmenti","checkBoxes",0,120,"checkBoxRect","checkBoxCross",true,"TRSegmenti",labeltextStyles,12,6,undefined,toggleMapLayer);
        myMapApp.checkBoxes["ehis"] = new
checkBox("ehis","checkBoxes",0,140,"checkBoxRect","checkBoxCross",true,"ehis",labeltextStyles,12,6,undefined,toggleMapLayer);
        loadProjectSpecific();
    }

    function buttonTextChange(id,status) {
      if (status == "closed" || status == "opened") {
        var buttonText = document.getElementById("buttonText"+id);
        butText = buttonText.getAttributeNS(attribNS,"buttonText");
        if (myMapApp.Windows[id].closed) {
          buttonText.firstChild.nodeValue = "Open " + butText;
        }
        else {
          buttonText.firstChild.nodeValue = "Close " + butText;
        }
      }
      if (id == "navWindow") {
        if (status == "created" || status == "resized") {
          addOrUpdateWindowDecoration(id);
        }
      }
    }

    function addOrUpdateWindowDecoration(id) {
      var myWindow = myMapApp.Windows[id];
      var group = document.createElementNS(svgNS,"g");
      var rect1 = document.createElementNS(svgNS,"rect");
      rect1.setAttributeNS(null,"x",-15);
      rect1.setAttributeNS(null,"y",-15);
      rect1.setAttributeNS(null,"width",20);
      rect1.setAttributeNS(null,"height",(myWindow.height+2));
      rect1.setAttributeNS(null,"fill","lightsteelblue");
      rect1.setAttributeNS(null,"stroke","none");
      group.appendChild(rect1);
      var rect2 = document.createElementNS(svgNS,"rect");
      rect2.setAttributeNS(null,"x",-15);
      rect2.setAttributeNS(null,"y",-15);
      rect2.setAttributeNS(null,"width",(myWindow.width + 16));
      rect2.setAttributeNS(null,"height",20);
      rect2.setAttributeNS(null,"fill","lightsteelblue");
      rect2.setAttributeNS(null,"stroke","none");
      rect2.setAttributeNS(null,"id","decoGroupMinimized"+myWindow.id);
      group.appendChild(rect2);
      var text = document.createElementNS(svgNS,"text");
      text.setAttributeNS(null,"x",0);
      text.setAttributeNS(null,"y",myWindow.height-20);
      text.setAttributeNS(null,"font-family","Arial,Helvetica");
      text.setAttributeNS(null,"font-size",12);
      text.setAttributeNS(null,"fill","dimgray");
      text.setAttributeNS(null,"transform","rotate(-90,0,"+(myWindow.height-20)+")");
      text.setAttributeNS(null,"pointer-events","none");
      var textNode = document.createTextNode(myWindow.titleText);
      text.appendChild(textNode);
      group.appendChild(text);
      //move buttons
      var x = -5;
      var curY = -5;
      if (myWindow.closeButton) {
        myWindow.closeButtonInstance.setAttributeNS(null,"x",x);
```

```
            myWindow.closeButtonInstance.setAttributeNS(null,"y",curY);
            curY += 20;
         }
         if (myWindow.maximizeButton) {
            myWindow.maximizeButtonInstance.setAttributeNS(null,"x",x);
            myWindow.maximizeButtonInstance.setAttributeNS(null,"y",curY);
            curY += 20;
         }
         if (myWindow.minimizeButton) {
            myWindow.minimizeButtonInstance.setAttributeNS(null,"x",x);
            myWindow.minimizeButtonInstance.setAttributeNS(null,"y",curY);
         }
         myWindow.addWindowDecoration(group,true,"top");
      }
   function loadProjectSpecific() {
         //adopt width and height of map extent
         document.getElementById("myScaleTextW").firstChild.nodeValue =
formatNumberString(myMainMap.curWidth.toFixed(myMainMap.nrDecimals),",") + myMainMap.units;
         document.getElementById("myScaleTextH").firstChild.nodeValue =
formatNumberString(myMainMap.curHeight.toFixed(myMainMap.nrDecimals),",") + myMainMap.units;
      }

      //this function toggles the visibility of a map layer
      function toggleMapLayer(id,checkStatus,labelText) {
         var mapLayer = document.getElementById(id);
         var visibleStatus = "hidden";
         if (checkStatus) {
            visibleStatus = "visible";
         }
         mapLayer.setAttributeNS(null,"visibility",visibleStatus);
      }

      function ObjectClick(text) {
         //show an alert message
         alert(text);
```