ZAVOD ZA ELEKTRONIKU, MIKROELEKTRONIKU, RAČUNALNE I INTELIGENTNE SUSTAVE
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
SVEUČILIŠTE U ZAGREBU

ISPITNO OKRUŽENJE
DEFINICIJA XML-a I PARSER

Jure Rastić

SEMINARSKI RAD

Zagreb, 2006.

Sadržaj

1. Uvod

Internet se razvio u sveprisutni medij za razmjenu informacija, te obavljanje posla u globalnim razmjerima koji unosi temeljite promjene u dosadašnji način života, rada, zabave i učenja. Računalne mreže postale su nezamjenjivim segmentom infrastrukture. Mrežne aplikacije postaju sve raširenije i složenije, te je njihova stabilnost i robusnost postala kritična točka o kojoj može ovisiti nečiji život, posao, te općenito – budućnost.

Kako bi se kreirala mrežna aplikacija koja je sigurna, stabilna, te zadovoljava postavljene zahtjeve nad njom, potrebno je veliko iskustvo u programiranju, poznavanju računalnih mreža, te mogućih sigurnosnih propusta pojedinog pristupa dizajnu arhitekture i sl. Jedini realan način za kreiranje takve aplikacije je neprestano ispitivanje ideje dizajna aplikacije kao i samo testiranje onog što je napravljeno.

IEEE/ANSI definicija testiranja:

Testiranje programske opreme je proces izvođenja sustava ili komponente pod strogo definiranim uvjetima, pri čemu se promatraju i bilježe rezultati izvođenja te ocjenjuju svojstva sustava ili komponente.

To je proces analize programa s ciljem otkrivanja razlika između zahtijevanog i stvarnog ponašanja programa.

U svijetu razvoja programske podrške općenito postoje mnoga okruženja za ispitivanje ispravnosti same funkcije određenog dijela aplikacije (*unit* testovi i sl.) i postoje formalne metode za definiranje ispravnosti rada određenog odsječka koda, dok na području ispitivanja mrežnih aplikacija je situacija bitno lošija. Najčešći općeniti problemi (nevezani uz tip mrežne aplikacije) koji se javljaju kod testiranja ovih aplikacija su sljedeći:

- **priprema sklopovske i programske podrške**
  Kako bi se aplikacija ispitala potrebno je stvoriti odgovarajuće uvjete u kojima se aplikacija treba izvoditi. Potrebno je postaviti računala, osigurati potrebnu programsku podršku, kreirati zadovoljavajuću topologiju mreže itd.
- **postavljanje aplikacije**
  Razvijanu aplikaciju je potrebno distribuirati na prethodno postavljena računala, te izvršiti dodatne inicijalizacijske korake.
- **pokretanje aplikacije**
  Jednom postavljenu aplikaciju treba pokrenuti na svim računalima koji su domaćini te aplikacije. Problem postavljanja i pokretanja aplikacije je jako izražen kod testova u kojima je potreban velik broj domaćina.
- **nadziranje rada**
  Pokrenut skup aplikacija je potrebno nadzirati kako bi se utvrdila ispravnost rada. Ovaj problem postaje nepremostiv za nadziranje od strane čovjeka kod situacija u kojima se testira velik broj aplikacija koje se paralelno izvršavaju što je čest slučaj.
- **pribavljanje rezultata izvođenja**
  Nakon sprovedenog testa potrebno je prikupiti sve podatke koji su nastali izvođenjem ispitivanih aplikacija na jedno mjesto kako bi se vršila daljna analiza.

- **analiza rezultata**
  Podatke koji su generirani od strane ispitivanih aplikacija potrebno je na kraju ili tijekom testa analizirati kako bi se došlo do dodatnih informacija o radu ispitivane aplikacije.
- **rad s novim verzijama**
  Izvorni kod razvijane aplikacije se često mijenja, te je potrebno nakon svake izmjene neke testove ponoviti kako bi se osiguralo da izmjena u kodu nije uzrokovala nove probleme osim što je stare probleme rješila.

Razvojni timovi uobičajno kreiraju vlastiti sustav za ispitivanje razvijane mrežne aplikacije koji je usko vezan za mogućnosti i problematiku te aplikacije što ga kasnije čini neuporabljivim za daljne korištenje.

Javlja se potreba za testnim okruženjem koje omogućuje jednostavnu provedbu automatizacije izvršavanja testa mrežne aplikacije tijekom razvoja. Potrebno je ostvariti programsku podršku koja bi automatizirala izvođenje procesa ispitivanja mrežne aplikacije rješavajući prethodno navedene probleme.

Ovaj seminar obrađuje ideju o izvedbi takvog ispitnog okruženja na način da se kreira format XML datoteke (XML shema) kojom bi se odredili akteri testa, subjekti ispitivanja, te postupci za analizu izvođenja mrežne aplikacije. Na taj način se stvara jasna slika o mogućnostima ispitnog okruženja i podloga za dokumentaciju o izradi definicije testa putem ovog XML-a. Ujedno su time definirani zahtjevi pri implementaciji ispitnog okruženja.

2. Teorijska razrada

U ovom poglavlju bit će zadane potrebe definiranja elemenata testa, te samo ukratko opisan sustav na kojem bi se test trebao izvršavati.

2.1. Definicija potreba

Pri automatizaciji ispitnog procesa kreira se skup datoteka i skripti koje omogućuju izvršavanje testa:

- skripte za pokretanje
- skripte za analizu
- skripte za detekciju greški
- datoteke za konfiguriranje
- datoteke za rad aplikacije
- različite verzije razvijane aplikacije

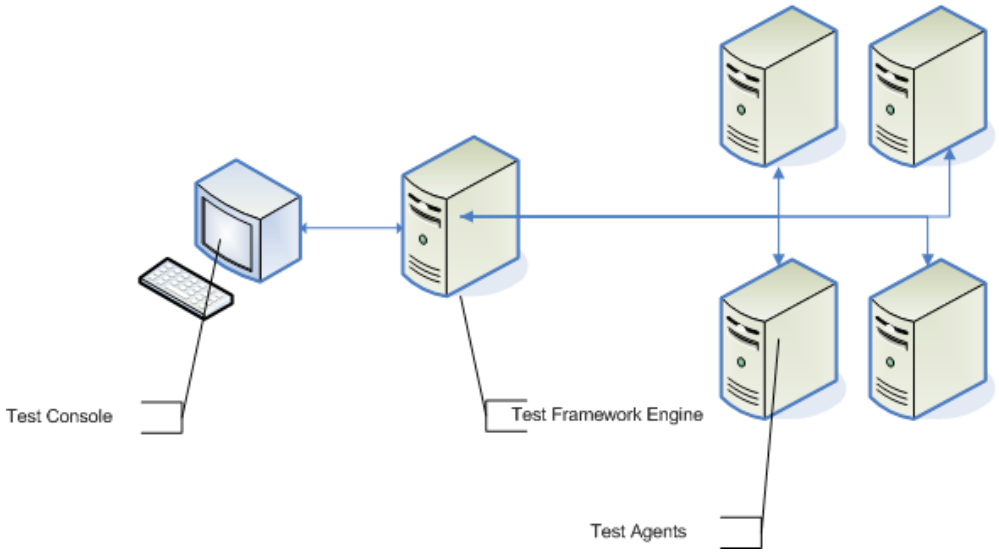
Definiraju se uvjeti koje aplikacija mora zadovoljiti, te načini za detekciju i dojavu greške pri izvršavanju. Potrebno je odrediti topologiju mreže i sklopovsku opremu na kojoj će se sam test izvršavati.

Često se primjenjuje pristup razbijanja cjelokupnog testa na manje dijelove koji mogu biti međusobno zavisni ili nezavisni kako bi se jednostavnije vršila analiza rezultata izvođenja testa.

Kreirani XML mora zadovoljiti sve potrebe za definicijama navedenih stavki što je moguće jednostavnije i jasnije.

2.2. Ispitno okruženje

Na slici 2.1 prikazana je arhitektura ispitnog okruženja.



Slika 2.1: Arhitektura ispitnog okruženja

Ukratko opisane uloge triju komponenti:

- **testna konzola (eng. *Test Console – TC*)**
  Konzola putem koje korisnik vrši interakciju s ispitnim okruženjem. Korisnik može unijeti test, pokrenuti izvršavanje testa, pregledati status izvođenja, zaustaviti test, te prikupiti rezultate testa.
- **pokretač ispitnog okruženja (eng. *Test Framework Engine – TFE*)**
  Centralni dio sustava. Prihvaća i daje izvještaj o testu, vrši komunikaciju s agentima i upravlja njima. Prilikom pokretanja sustava otkriva dostupne agente i prijavljuje ih u okolinu.
- **testni agenti (eng. *Test Agents – TA*)**
  Testni agenti izvršavaju same dijelove testa, tj. pokreću mrežnu aplikaciju koja se testira i vrše nadzor. Podaci o izvršavanju se spremaju u dnevnik za kasniju analizu.

2.3. Sustav upravljan događajima

Pristup izvođenju testa u ispitnom okruženju se zasniva na samoj prirodi testiranja mrežnih aplikacija. Mrežne aplikacije komuniciraju međusobno i izvode potrebne akcije za ostvarivanje svog zadatka. Potrebno je analizirati komunikaciju i rezultate izvršenih zadataka kako bi se utvrdila ispravnost rada aplikacije. Komunikacija i izvršavanje zadataka često nemaju prethodno definiran redoslijed ili je vremenski prezahtjevno formalno ga definirati.

Stoga je potrebno ostvariti paralelan skup neovisnih procesa za:

- očitavanje komunikacije
- ispitivanje uspjeha zadatka
- sinkronizaciju zavisnih procesa

Aplikacija koja se izvršava mora dojaviti ispitnom okruženju o relevantnom događaju koji se zbio. Ispitno okruženje odgovara na događaj prethodno definiranim akcijama. Akcija može biti pokretanje druge aplikacije, promjena konfiguracije aplikacije i sl.

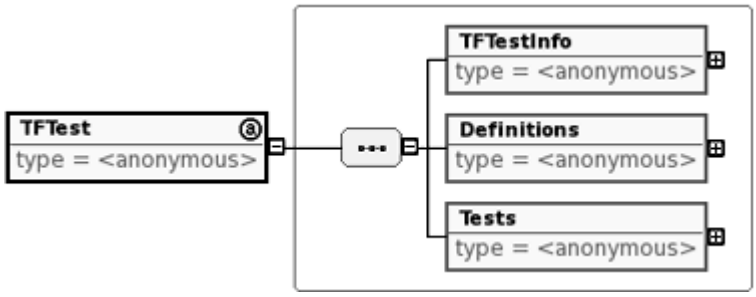
Ovakva paradigma programiranja se naziva programiranje pokretano događajima (eng. *event-driven programming*). Za razliku od tradicionalnih pristupa izrade programa putem niza slijednih poziva rutina i naredbi, te povremenih poziva funkcija za grananje toka izvođenja, kod programiranja pokretanog događajima kontrola toka izvođenja programa je vezana na događaje koji se zbivaju u sustavu, te oni upravljaju tijekom izvođenja.

Ideja je da se korištenjem signala u ispitnom okruženju ostvari sustav upravljan događajima na način da se propagiraju poruke o događajima na određenom agentu, te nakon toga poduzmu definirane akcije na svim agentima koji očekuju ovaj događaj. Dakle, potrebno je definirati samo akcije za određene događaje unutar ispitnog okruženja. Ovim se ostvaruje jednostavnost definiranja testa, te široko područje primjene u ispitivanju mrežnih aplikacija.

Iznesene ideje su bile uvod za definiranje formata XML datoteke koji će biti obrađen u idućem poglavlju.

3. Elementi XML-a

Cjelokupan XML je podijeljen u osnovne dijelove koji su organizirani na način da se što bolje razdvoje logički dijelovi samog testa, te su prikazani slikom 3.1.

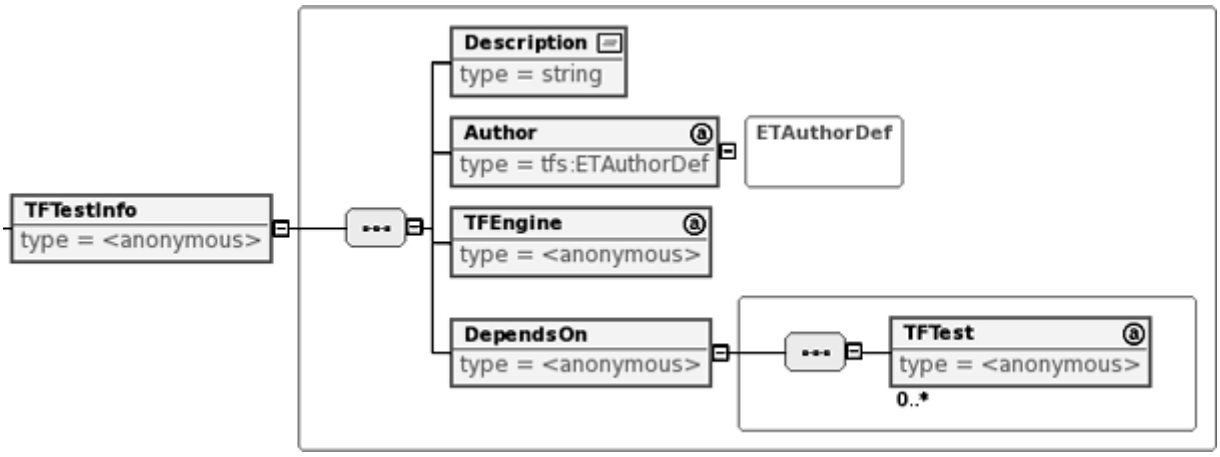


Slika 3.1. Osnovni elementi XML-a

Elementi XML-a opisani su detaljno u idućim potpoglavljima.

3.1. Preambula o testu

U ovom dijelu XML-a dani su opći podaci o testu kao što su autor testa, kratak opis i sl. Prikaz odgovarajućih elemenata dan je slikom 3.2.



Slika 3.2. Informacije o testu

U sljedećem isječku XML-a prikazan je primjer korištenja ovog dijela:

```
<tfs:TFTest
  TFTestID="HttpTest"
  version="0.2" date="2006-08-29"
  xmlns:tfs="TFSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="TFSchema tfsch.xsd">
  <TFTestInfo>
     <Description>Simple test that tests thttpd and wget</Description>
     <Author
        name="Jure"
        surname="Rastic"
        url="http://osl.zemris.fer.hr/"
        email="jure.rastic@gmail.com" />

     <TFEngine version="0.1"/>

     <DependsOn />
  </TFTestInfo>
```

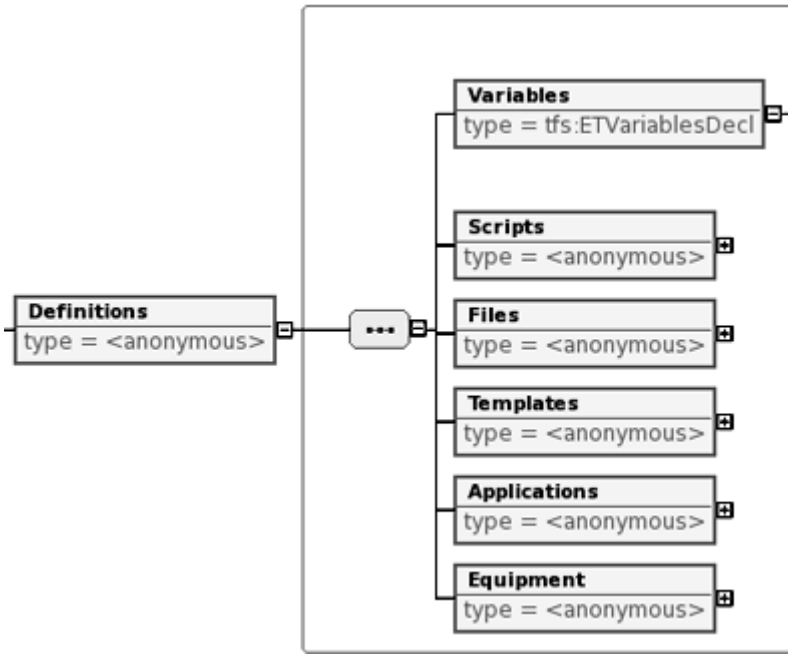
Za glavni element (eng. *root element*) *TFTest* se definira identifikator, verzija testa, datum kreiranja, te lokacija XML sheme.

U *TFTestInfo* elementu se nalazi opis testa i autor. Atribut *email* autora može poslužiti za dojavu o statusu testa.

TFEngine element definira potrebnu verziju ispitnog okruženja za izvršavanje testa, dok *DependsOn* element sadrži listu testova o kojima ovisi ovaj test.

3.2. Definicija testne okoline

U ovom dijelu XML-a obavljena je podjela definicija svih potrebnih elemenata koji će se koristiti kroz izvođenje testa. Prikaz podelemenata je dan slikom 3.3.



Slika 3.3. Elementi definicije

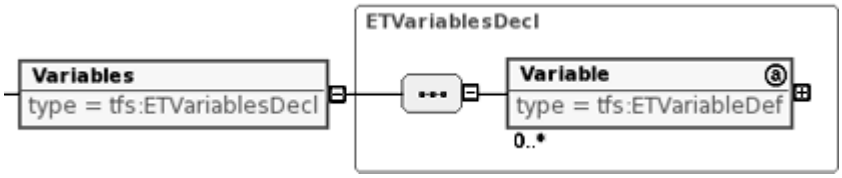
Svaki element će biti opisan detaljno u idućim potpoglavljima.

3.2.1. Varijable

Kako bi se ostvarila mogućnost privremenog spremanja vrijednosti rezultata neke akcije, potrebno je kreirati varijable koje će čuvati tu vrijednost. Vrijednost varijable u određenom trenutku se može kasnije iskoristiti za odlučivanje o akciji koju treba pokrenuti ili za kasniju analizu.

Varijable koje su definirane u ovom dijelu XML-a su **javne varijable** koje mogu biti pročitane i promijenjene od strane bilo kojeg agenta ne navodeći putanju varijable. O privatnim varijablama biti će više riječi u poglavlju 3.3.2.

Grafički prikaz podelemenata je dan slikom 3.4.



Slika 3.4. Element varijabla

Primjer korištenja ovog elementa je dan sljedećim isječkom XML koda:

```
<Variables>
  <Variable VarID="V_SERVER_IP" type="string" value="10.0.0.2" />
  <Variable VarID="V_SERVER_PORT" type="string" value="8080" />
  <Variable VarID="V_FILE_NAME" type="string" value="image.iso" />
  <Variable VarID="V_HTDOCS_FOLDER" type="string" value="./htdocs/" />
  <Variable VarID="V_file_sum" type="string" value="unknown" />
</Variables>
```

Za varijablu se definiraju:

- identifikator - *VarID*
  koji će kasnije poslužiti za referenciranje varijable
- tip varijable - *type*
  mogući tipovi su *integer*, *decimal*, *string*, *boolean*
- početne vrijednosti varijable - *value*
  varijabla ima zadanu vrijednost prije prvog pridjeljivanja

U isječku XML koda je prikazana deklaracija 5 varijabli od kojih su prve 4 definirane, a zadnja je postavljena na vrijednost *unknown*.

U ovom dokumentu će se koristiti ime varijable pisano velikim slovima ukoliko varijabla neće mijenjati vrijednost, a malim slovima ukoliko varijabla mijenja vrijednost tijekom izvršavanja testa.

3.2.2. Skripte

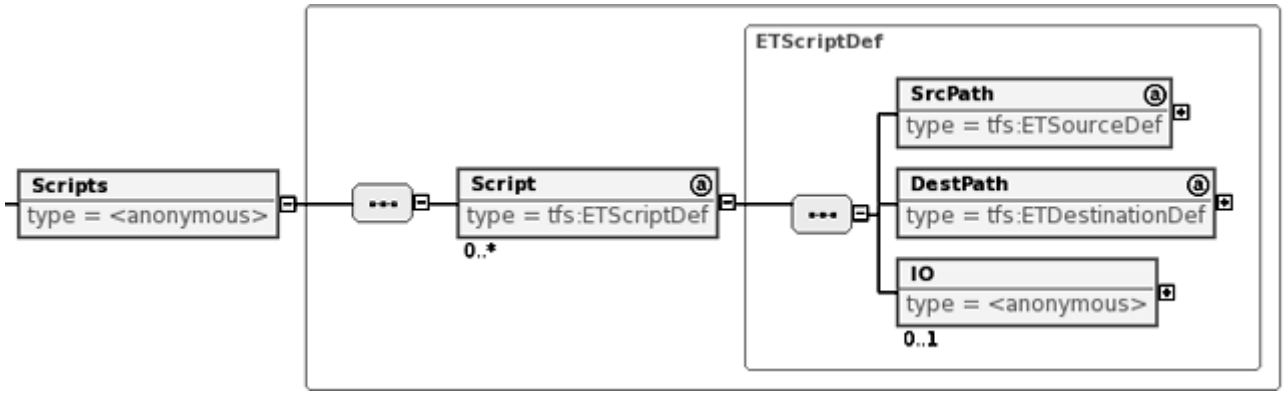
Kroz razmatranje ideje o izvršavanju samog testa pojavili su se problemi jednoznačnog definiranja pokretanja aplikacije, izvršavanja potrebnih akcija, te izvođenja operacija nad varijablama. Najjednostavniji pristup je preslikati ručni način rješavanja ovog problema kako bi se ostvarilo široko područje primjene.

Moguće je definirati tri tipa skripti:

- izvršne skripte
- računske skripte
- skripte za parsiranje

Ovim skriptama je moguće izvesti sve potrebne operacije kroz izvršavanje samog testa, a povezivanje skripti na raspodijeljenom sustavu se vrši pomoću slanja signala o određenoj vrsti događaja izvršavane skripte. Detaljnije o načinu upotrebe samih skripti pogledati u poglavlju 3.3.2.

Grafički prikaz podelemenata je dan slikom 3.5.



Slika 3.5. Elementi za definiranje skripte

Deklaracija skripte je prikazana sljedećim isječkom XML koda:

```
<Script ScriptID="thttpd_starter" type="execute" lang="shell">
  <SrcPath type="local" path="scripts/thttpd_starter.sh" />
  <DestPath path="scripts/thttpd_starter.sh" />
</Script>
```

Za samu skriptu se definira:

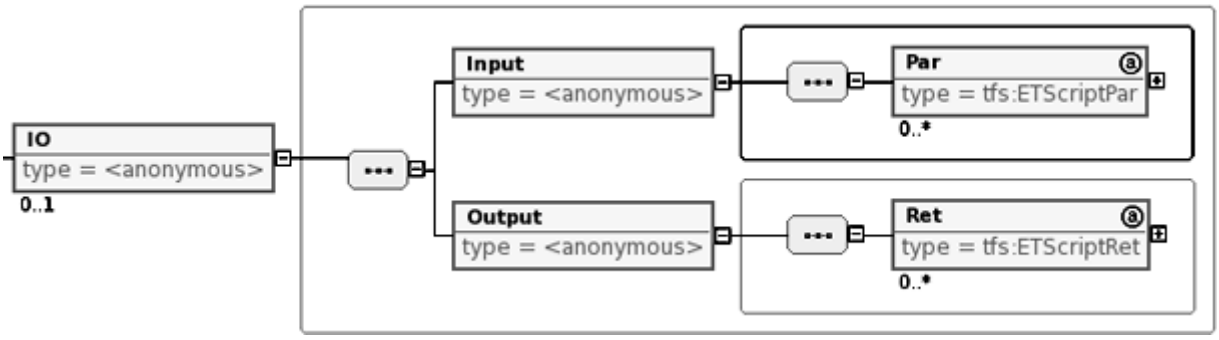
- identifikator skripte – *ScriptID*
  jedinstveni identifikator za ovu skriptu koji se koristi dalje u testu za referenciranje skripte
- tip skripte – *type*
  prilikom implementacije ispitnog okruženja biti će potrebno razlikovati skripte koje služe za obavljanje operacije nad varijablama i skripti koje parsiraju nekakav tok podataka. Prillikom pokretanja skripte primjenit će određen način pokretanja ovisno o ovom tipu.
- skriptni jezik – *lang*
  moguće je koristiti više skriptnih jezika, te se ovim atributom izriče jezik kojeg skripta koristi

Podelementi *SrcPath* i *DestPath* definiraju lokaciju odakle treba dohvatiti skriptu, te gdje je treba pozicionirati prilikom izvršavanja testa. Kod *SrcPath* elementa *type* atribut se koristi za definiranje tipa izvora datoteke.

Dopušteni tipovi su:

- lokalno – *local*
  datoteka skripte se nalazi na danoj putanji relativno naspram putanje XML datoteke
- na web poslužitelju – *http*
  datoteka se nalazi na web poslužitelju
- na ftp poslužitelju – *ftp*
  datoteka se nalazi na ftp poslužitelju

IO element se koristi kako bi se definirali ulazi i izlazi skripte, te je ovaj dio prikazan slikom 3.6.



Slika 3.6. Element ulaza i izlaza skripte (IO)

Sljedećim isječkom koda će biti prikazana tipična uporaba ovog elementa.

```
<Script ScriptID="S_inc_counter" type="calculate" lang="python">
  <SrcPath type="local" path="scripts/inc_counter.py" />
  <DestPath path="scripts/inc_counter.py" />
  <IO>
    <Input>
      <Par ParID="cur_count" index="1"/>
    </Input>
    <Output>
      <Ret RetID="inc_count" index="1"/>
    </Output>
  </IO>
</Script>
```

Isječkom koda je definirana računska skripta (atribut *type* ima vrijednost *calculate*). Skripta ima svoje ulazne i izlazne parametre definirane *IO* elementom. U *Input* podelementu nalazi se lista ulaznih parametara, dok u *Output* elementu se nalazi lista povratnih vrijednosti. Za pojedini ulaz i izlaz se definira identifikator parametra (*ParID*, *RetID*), te indeks pozicije atributom *index* na kojem se parametar nalazi (prilikom poziva same skripte, te dohvata rezultata).

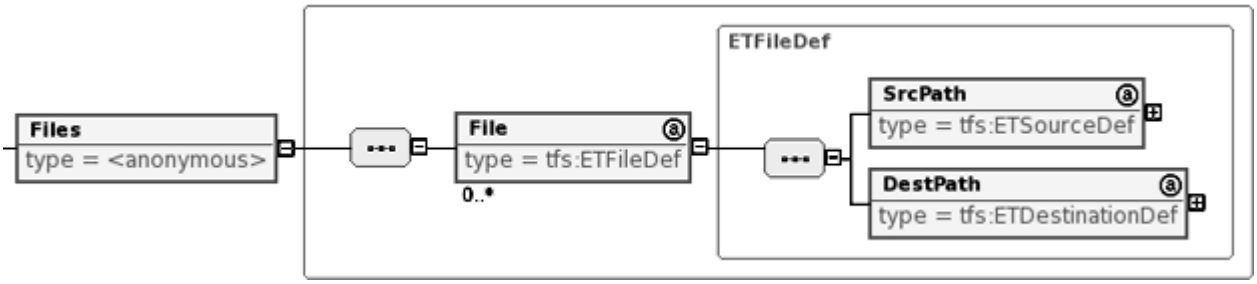
3.2.3. Datoteke

Prilikom testiranja mrežnih aplikacija često se koriste dodatne datoteke koje nisu same po sebi dio razvijane aplikacije, već se koriste kako bi se prilagodilo ponašanje aplikacije (konfiguracijske datoteke) ili kako bi se ostvarila nekakva primjena mrežne aplikacije (npr. posluživanje dane datoteke pri testiranju poslužiteljski orijentirane aplikacije).

Ove datoteke je potrebno također definirati kako bi se mogle koristiti u daljnim koracima testa. Moguće je rad s ovim datotekama (definiranje i dohvat) izvesti na druge načine kao što je izrada

skripte koja će ih dinamički generirati, no predlaže se uporaba ovog elementa XML-a kako bi se ostvarila što veća jasnoća pri opisu testa.

Grafički prikaz podelemenata je dan slikom 3.7.



Slika 3.7. Element za definiranje datoteke

Sljedećim isječkom koda će biti prikazana tipična definicija ovog elementa.

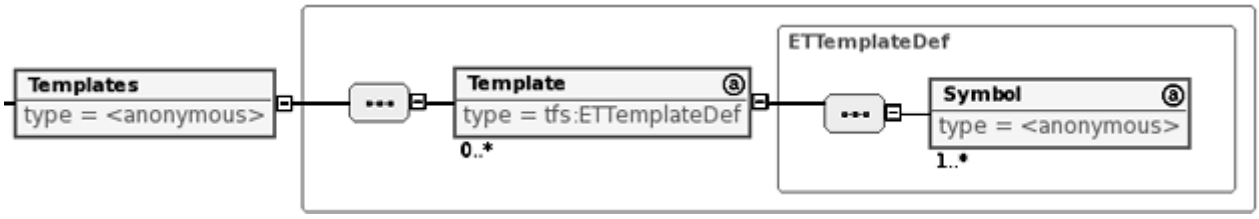
```
      <File FileID="F_served_file">
        <SrcPath type="local" path="files/image.iso" />
        <DestPath path="htdocs/image.iso" />
      </File>
```

Atributi elementa, te njegovi podelementi imaju isto značenje kao i kod definiranje skripte.

3.2.4. Predlošci

U prethodno spomenutim konfiguracijskim datotekama često je potrebno mijenjati određene stavke konfiguracijske datoteke za aplikaciju kako bi se ispitala željena funkcionalnost. Da bi se omogučio što jednostavniji i jasniji pristup mijenjanju parametara aplikacije, kreiran je ovaj element koji definira stavke u datoteci koje se mogu izmjeniti.

Grafički prikaz podelemenata je dan slikom 3.8.



Slika 3.8. Element za definiranje predloška

Sljedećim isječkom koda će biti prikazana tipična uporaba ovog elementa.

```
    <Templates>
      <Template TplID="T_thttp_conf">
        <Symbol string="$ip" value="none" SymID="IP" />
        <Symbol string="$port" value="8080" SymID="PORT" />
        <Symbol string="$htdocs" value="./htdocs/" SymID="HTDOCS_FOLDER" />
      </Template>
    </Templates>
```

Za predložak potrebno je definirati identifikator predloška atributom *TplID*, te listu simbola u datoteci nad kojom će se primjeniti predložak podelementima *Symbol*.

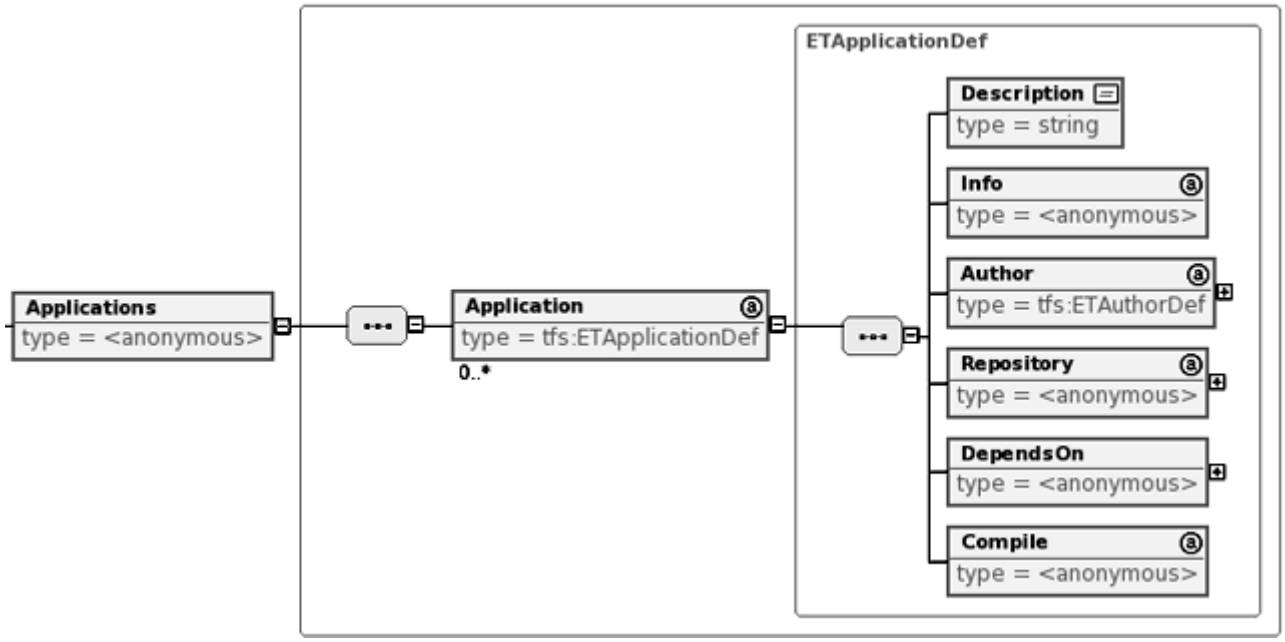
Za svaki simbol potrebno je definirati:

- identifikator simbola – *SymID*
  jedinstveni identifikator koji se koristi za referenciranje simbola
- znakovni niz kojeg treba zamijeniti – *string*
  datoteka na koju se primjenjuje predložak sadrži ovaj niz, te će biti zamijenjen danom vrijednošću
- početnu vrijednost simbola – *value*
  ukoliko primjenom predloška na ciljnu datoteku nije zadana vrijednost na koju treba postaviti traženi niz, tada će se unijeti ovdje navedena vrijednost.

3.2.5. Aplikacije

U ovom dijelu se definiraju sve aplikacije koje su sudionici testa bilo da su u ulozi subjekta ispitivanja ili pomoćne aplikacije za izvršavanje testa. Prilikom definiranja aplikacije potrebno je odrediti način dohvata izvornog koda ili izvršnog paketa za instalaciju. Dodatni parametri definiranja će biti dodatno objašnjeni.

Grafički prikaz podelemenata je dan slikom 3.9.



Slika 3.9. Element za definiranje aplikacija

Sljedećim isječkom koda je prikazana tipična uporaba ovog elementa:

```
<Application AppID="wget">
  <Description>internet download client</Description>
  <Info name="wget" version="1.10.2"
    url="http://www.gnu.org/software/wget/" />
  <Author name="Hrvoje" surname="Niksic"/>
  <Repository type="source" >
    <Source type="http"
      path="http://ftp.gnu.org/pub/gnu/wget/wget-1.10.2.tar.gz" />
  </Repository>
  <DependsOn>
    <CompileTime />
    <RunTime />
  </DependsOn>
  <Compile type="beforeDistribution" />
</Application>
```

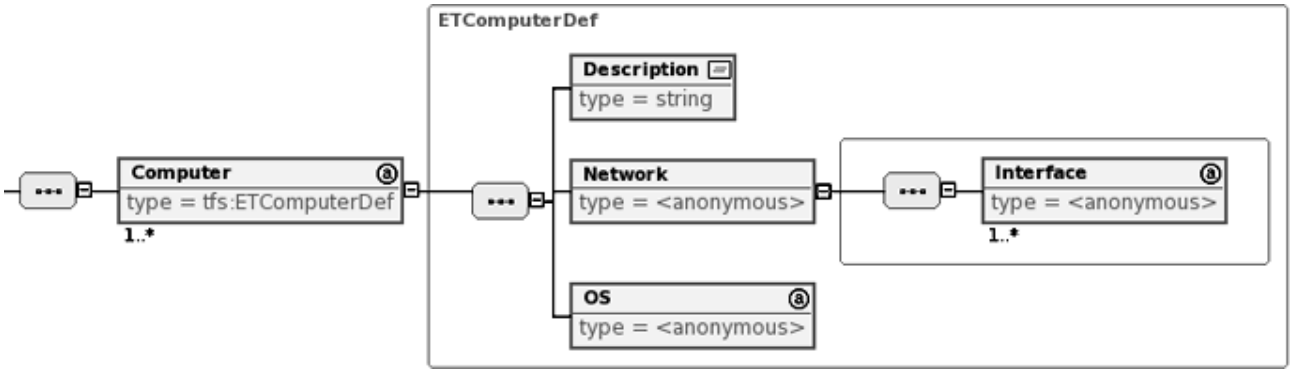
Ovim osječkom koda je definirana *wget* aplikacija. Za samu aplikaciju općenito je potrebno definirati:

- identifikator aplikacije – atribut *AppID*
  Koristi se za daljne referenciranje aplikacije.
- opis aplikacije – element *Description*
  Kratak opis aplikacije.
- općenite informacije o aplikaciji – element *Info*
  Naziv aplikacije, verzija, internet stranice aplikacije.
- podaci o autoru aplikacije – element *Author*
  Mogući atributi su ime, prezime, e-mail adresa, te internet stranice autora.
- način dohvata aplikacije – element *Repository*
  Definira se tip distribucije aplikacije atributom *type* koji može poprimiti vrijednost *source* ili *binary*. Podelementom *Source* se definira način dohvata paketa aplikacije, te je jednak elementu *SrcPath* objašnjenom kod definiranja elementa *Script* u poglavlju 3.2.2.
- zavisnosti o drugim programskim paketima – element *DependsOn*
  Definira se lista biblioteka koje moraju postojati tijekom prevođenja aplikacije (*CompileTime* lista), te tijekom izvršavanja aplikacije (*RunTime* lista).
- tip prevođenja – element *Compile*
  Atributom *type* ovog elementa se definira način prevođenja. Aplikacija se može prevesti prije distribucije na računala domaćine (prevođenje na pokretaču ispitnog okruženja – TFE-u) ili na samim domaćinima (prevođenje na testnom agentu – TA-u). U danom primjeru definirano je prevođenje na TFE-u.

3.2.6. Sklopovska oprema

Potrebno je definirati računala koja su domaćini testiranih aplikacija. Ovaj dio je zasada odrađen prilično minimalistički kako bi se pokrili osnovni zahtjevi. Kreiran je jednostavan način za definiranje mrežnih sučelja i operacijskog sustava na računalu domaćinu.

Grafički prikaz podelemenata je dan slikom 3.10.



Slika 3.10. Element za definiranje računala

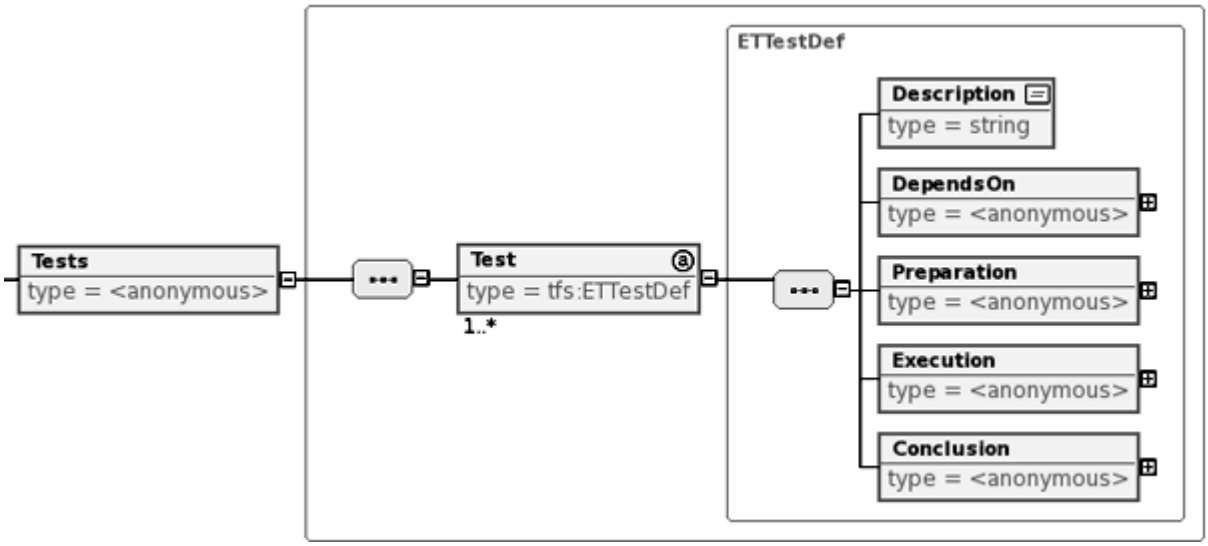
Sljedećim isječkom koda je prikazana tipična uporaba ovog elementa:

```
<Computer CompID="server" type="real" >
  <Description>Server computer</Description>
  <Network>
    <Interface
      IfaceID="eth0"
      type="ethernet"
      address="10.0.0.2"
      netmask="255.255.255.0"
      gateway="10.0.0.1"
    />
  </Network>
  <OS type="linux" />
</Computer>
```

Za računalo se definira identifikator računala atributom *CompID*, te tip računala atributom *type*. Nadalje, uloga računala se ukratko opisuje elementom *Description*, te se računalu definiraju mrežna sučelja unutar *Network* elementa listom *Interface* elemenata. Svaki *Interface* element sadrži atribute potrebne za aktiviranje mrežnog sučelja na računalu čije ime odgovara standardnim nazivima koji se koriste prilikom aktivacije mrežnog sučelja. s vrijednošću atributa *type* elementa *OS* se definira operacijski sustav računala.

3.3. Opis testa

Od ovog elementa počinje definiranje samog testnog scenarija (eng. *test case*) aplikacije. Predviđeno je da se testiranje cjelokupne aplikacije razbija na testiranje pojedinih dijelova, pa je stoga omogućeno definiranje više testnih scenarija u ovom dijelu kako je prikazano slikom 3.11.



Slika 3.11. Element za opis testa

Za pojedini test se definira identifikator testa, opis testirane funkcionalnosti, te lista testova o kojima zavisi izvođenje ovog testa. Prilikom izvođenja cjelokupnog testa gradi se hijerarhija podtestova koje treba izvršiti prije ovog podtesta. Ukoliko neki od testova o kojima zavisi ovaj test ne uspije ispravno završiti, tada se prekida izvođenje cjelokupnog testa te dojavljuje status korisniku.

Idućim isječkom koda je prikazana uporaba osnovnog dijela opisa testa:

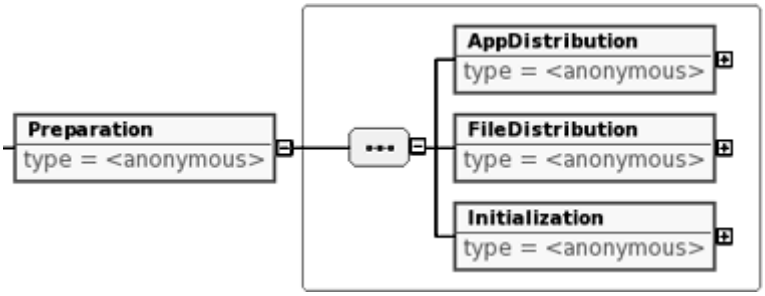
```
    <Test TestID="test-case-01" >
      <Description>This one tests basic functionality</Description>

      <DependsOn>
        <Test name="none" />
      </DependsOn>
```

Za test se definira identifikator atributom *TestID* za daljnje referenciranje na ovaj scenarij, kratak opis scenarija *Description* elementom, te lista podtestova o kojima ovaj podtest ovisi unutar *DependsOn* elmenta.

U narednim potpoglavljima će biti detaljno opisani ostali podelementi.

3.3.1. Priprema testa

Prije pokretanja samog testa potrebno je izvršiti distribuciju svih datoteka i aplikacija, te pokrenuti inicijalizacijske skripte koje služe za dodatno postavljanje ispitne okoline. To je preslikano u opis XML elementima, te prikazano slikom 3.12.



Slika 3.12. Element za definiranje pripreme testa

Distribucija obuhvaća raspodjelu datoteka i aplikacija na računala na kojima će se koristiti. Potrebno je primjetiti da se ne definiraju domaćini za skripte koje je potrebno izvršavati već će se to odrediti prilikom parsiranja predane XML datoteke na osnovu poziva definiranih skripti u dijelovima XML-a za pokretanje skripti.

Za inicijalizaciju okoline u kojoj će se aplikacija izvršavati omogućeno je korištenje elemenata XML-a za pokretanje skripte i primjenu predloška na određenu datoteku. Definira se na kojem od kreiranih domaćina se izvršava inicijalizacijski slijed naredbi (pozivi skripti, te primjena predložaka).

Primjer korištenja *Preparation* elementa je prikazan u nastavku.

Odsječak za definiciju distribucije aplikacije:

```
      <Preparation>
        <AppDistribution>
          <Application name="wget">
            <Host name="drone1" />
            <Host name="drone2" />
          </Application>
        .
        .
```

U prethodnom odsječku XML koda prikazano je definiranje distribucije aplikacije. Atributom *name* elementa *Application* se odabire aplikacija koju treba postaviti na računala. Odabir računala se opisuje listom *Host* elemenata s *name* atributom koji specifira računalo domaćin za navedenu aplikaciju. Za vrijednost *name* atributa u *Application* elementu se odabira jedan od identifikatora definiranih aplikacija, a za *name* atribut *Host* elementa identifikator definiranog računala domaćina.

U primjeru je dana definicija distribucije aplikacije *wget* na računala *drone1* i *drone2*.

Odsječak za definiciju distribucije datoteka:

```
        .
        .
        <FileDistribution>
          <File name="F_servedFile">
            <Host name="server" />
          </File>

          <File name="F_thttp_conf">
            <Host name="server" />
          </File>
        </FileDistribution>
        .
        .
```

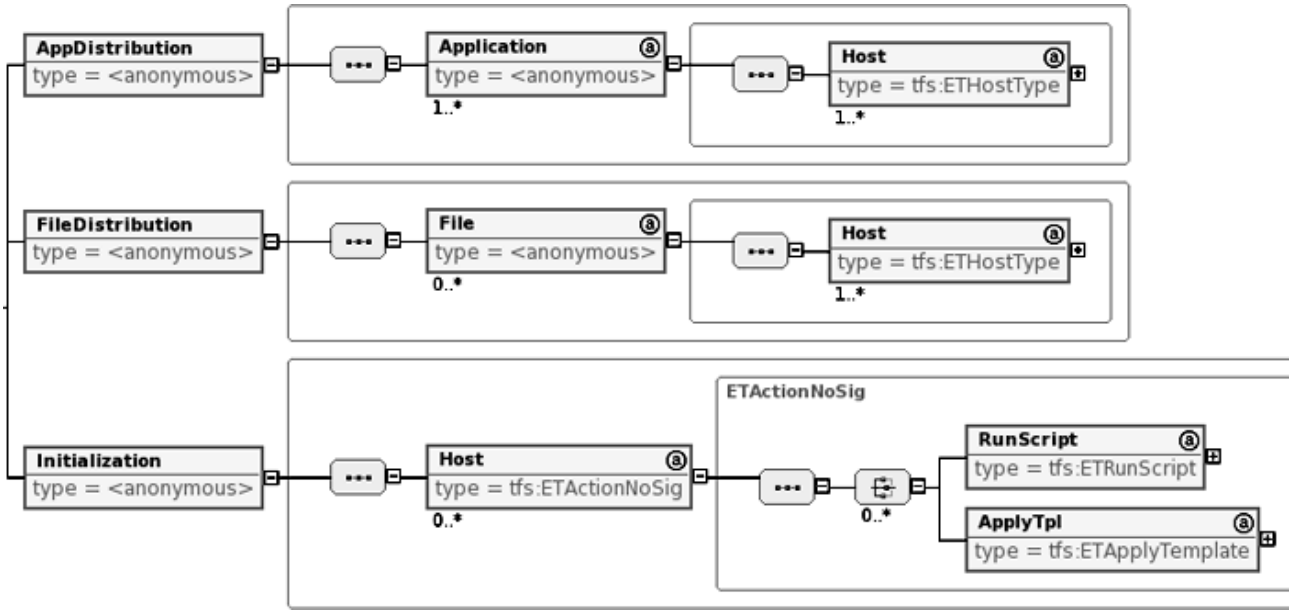
Slično kao i kod distribucije aplikacija, u *FileDistribution* elementu se definira koja se datoteka mora postaviti na koje računalo.

Odsječak za definiciju inicijalizacije testa:

```
        .
        .
        <Initialization>
          <Host name="server">
            <ApplyTpl Tpl="T_thttp_conf" File="F_thttp_conf">
              <Replace Sym="IP" Var="V_SERVER_IP" />
              <Replace Sym="PORT" Var="V_SERVER_PORT" />
              <Replace Sym="HTDOCS_FOLDER" Var="V_HTDOCS_FOLDER" />
            </ApplyTpl>

            <RunScript script="S_calc_checksum">
              <Connect>
                <In Par="file" Var="V_FILE_NAME" />
                <Out Ret="sum" Var="V_file_sum" />
              </Connect>
            </RunScript>
          </Host>
        </Initialization>
      </Preparation>
```

Pod elementom *Initialization* se nalazi lista *Host* elemenata koji definiraju atributom *name* domaćina na kojem se mora izvršiti niz akcija. Akcija može biti pokretanje neke od definiranih skripti elementom *RunScript* ili primjena predloška na datoteku elementom *ApplyTpl*. Detaljnije o elementima u poglavlju 3.3.2.

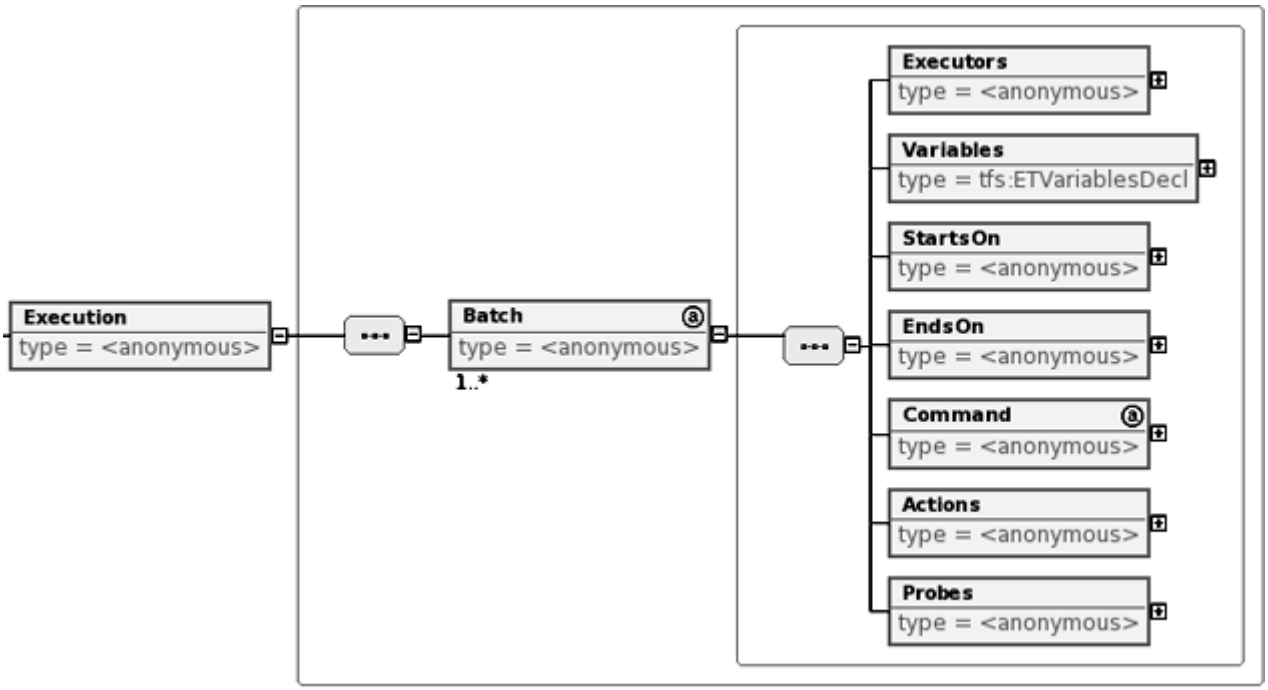
Grafički prikaz ukupnog elementa je dan slikom 3.13.



Slika 3.13. Detaljan prikaz podelemenata za pripremu testa

3.3.2. Izvršavanje testa

Ovim dijelom se definira izvršavanje samog testa. Element se sastoji od skupa zadataka (eng. *batch*) koji se izvršavaju na računalima domaćinima. Osnovni podelementi su grafički prikazani slikom 3.14.



Slika 3.14. Element definicije izvršavanja

Svaki *Batch* element se promatra kao nezavisna jedinica u izvršavanju testa sa svojim prostorom varijabli. Ovim elementom se olakšava kreiranje identičnih zadataka tako da se u *Executors* elementu popiše lista domaćina (elementi *Host*) na kojim će se izvršavati ovaj zadatak.

Odječak XML koda za definiranje Batch i Executors elemenata:

```
<Execution>
  <Batch BatchID="wget_batch" >
    <Executors>
      <Host name="drone1" />
      <Host name="drone2" />
    </Executors>
    .
    .
```

U drugom podelementu – *Variables*, se definiraju varijable koje se koriste u zadatku. Ovo su **privatne varijable** kojima se pristupa s istog agenta pomoću početne putanje *self* dok se s drugih agenata varijabli pristupa pomoću putanje koja se sastoji od identifikatora agenta na kojem je varijabla, točke, te identifikatora varijable.

Na primjer ako agent ***A*** želi pročitati varijablu ***VARB*** na agentu ***B***, potrebno je za ime varijable dati punu putanju varijable u sljedećem obliku ***B.VARB***. Za pristupanje prethodno definiranim javnim varijablama ne treba davati putanju, već je dovoljno samo navesti ime varijable (npr. ***JAVNA_VAR***).

Element *Variables* je identičan po strukturi i značenju atributa *Variables* elementu za definiranje javnih varijabli.

Kako bi se odredilo kada počinje zadatak, a kada završava definiraju se elementi *StartsOn* i *EndsOn* prikazano idućim odječkom XML koda:

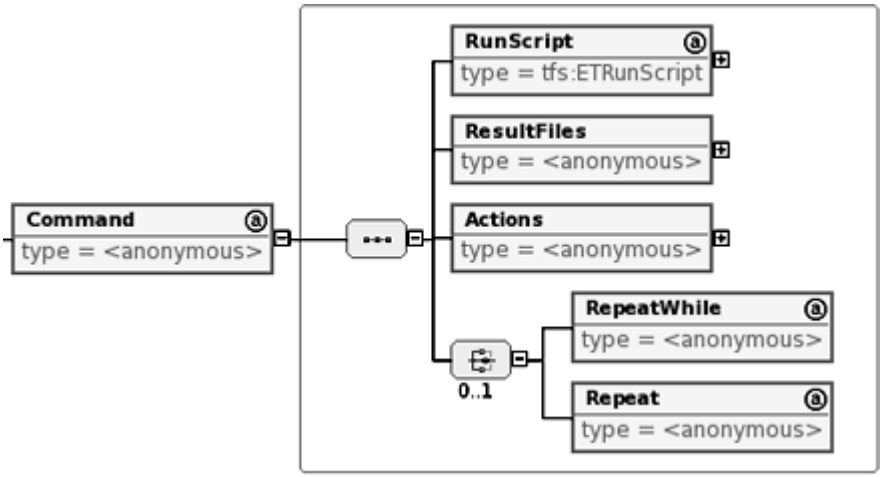
```
    .
    .
    <StartsOn>
      <Trigger from="server" signal="server_ready" />
    </StartsOn>

    <EndsOn>
      <Trigger from="server" signal="server_died" />
    </EndsOn>
    .
    .
```

Definiraju se *Trigger* elementi u podlisti koji sadrže atribute za:

- identifikator agenta koji šalje signal – *from*
  za vrijednost se odabire jedan od definiranih domaćina
- identifikator signala – *signal*
  za vrijednost se unosi signal na kojeg se želi započeti ili završiti zadatak

Ovime je omogućeno odabiranje točno određenog signala koji je poslan s točno određenog domaćina. Ukoliko nije važno s kojeg je domaćina siglan poslan, za vrijednost *from* atributa se navodi *any*.

Idući podelement je stavka *Command.* Ovaj element definira pokretanje ispitivane aplikacije, te obradu događaja koji se kreiraju tijekom izvršavanja aplikacije. Cijelo podstablo elemenata je dano slikom 3.15.



Slika 3.15. Elementi za definiranje komande

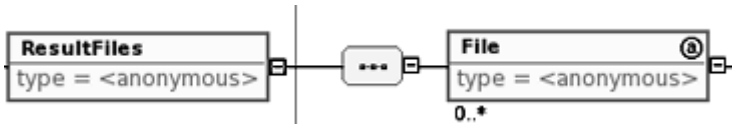
Odsječak XML koda:

```
.
.
<Command type="script" >
   <RunScript script="S_wget_starter">
      <Connect>
         <In Par="IP" Var="V_SERVER_IP" />
         <In Par="Port" Var="V_SERVER_PORT" />
         <In Par="Path" Var="V_FILE_NAME" />
      </Connect>
   </RunScript>
   .
   .
```

Da bi se definiralo pokretanje aplikacije koristi se *RunScript* element koji izabire izvršnu skriptu atributom *script* za startanje aplikacije. Podelementima *In* elementa *Connect* se definira prospajanje varijabli prema parametrima funkcije na način da se zada identifikator parametra atributu *Par*, a identifikator varijable atributom *Var*. Definiranje dodjeljivanja vrijednosti varijabli je slično samo što se umjesto *Par* atributa koristi *Ret* atribut *Out* elementa. Primjer korištenja je dan u odsječku XML-a za definiranje *Actions* elementa.

Odsječkom je prikazano pokretanje skripte *S_wget_starter* kojoj su kao ulazni parametri predane vrijednosti varijabli *V_SERVER_IP*, *V_SERVER_PORT* i *V_FILE_NAME*.

Jednom pokrenuta aplikacija generira izlazne datoteke koje se definiraju *Files* podelementima *ResultFiles* elementa kao što je prikazano na slici 3.16.



Slika 3.16. Element za definiranje izlazne datoteke

Definira se izlazna datoteka *File* elementom koju se parsira u trenutku dobivanja određenog signala na način prikazan idućim odsječkom koda:

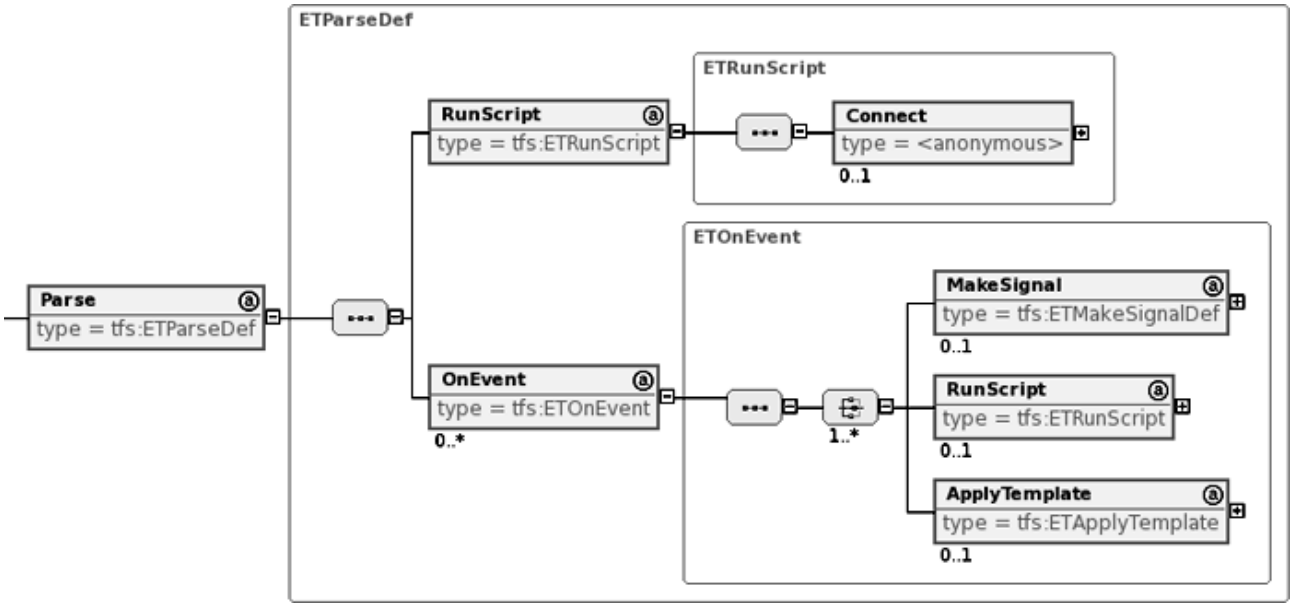
```
.
.
<ResultFiles>
   <File collect="no" path="image.iso">
      <Parse signal="wget_finished" from="self">
         <RunScript script="S_checksum">
            <Connect>
               <In Par="orig_sum" Var="V_file_sum" />
            </Connect>
         </RunScript>

         <OnEvent name="PASS">
            <MakeSignal type="info" name="good_download" />
         </OnEvent>

         <OnEvent name="FAIL">
            <MakeSignal type="error" name="bad_download" />
         </OnEvent>
      </Parse>
   </File>
</ResultFiles>
.
.
```

Unutar *File* elementa za definiranje izlazne datoteke se koriste atributi za:

- očuvanje datoteke – *collect*
  postavlja se na vrijednost *yes* ukoliko je potrebno sačuvati datoteku nakon jednog izvršavanja komande, vrijednost *no* inače
- putanju datoteke – *path*
  koristi se za definiranje putanje izlazne datoteke

Po primanju signala odabranog *signal*, te *from* atributima *Parse* elementa pokreće se skripta za parsiranje definirana *RunScript* elementom. Prilikom parsiranja datoteke skripta kreira događaje koji se mogu obuhvatiti *OnEvent* elementima. Kad skripta izda poruku o očekivanom događaju, izvode se akcije koje su unutar liste odgovarajućeg *OnEvent* elementa (definirano vrijednošću *name* atributa).

Parse element je grafički prikazan na slici 3.17.

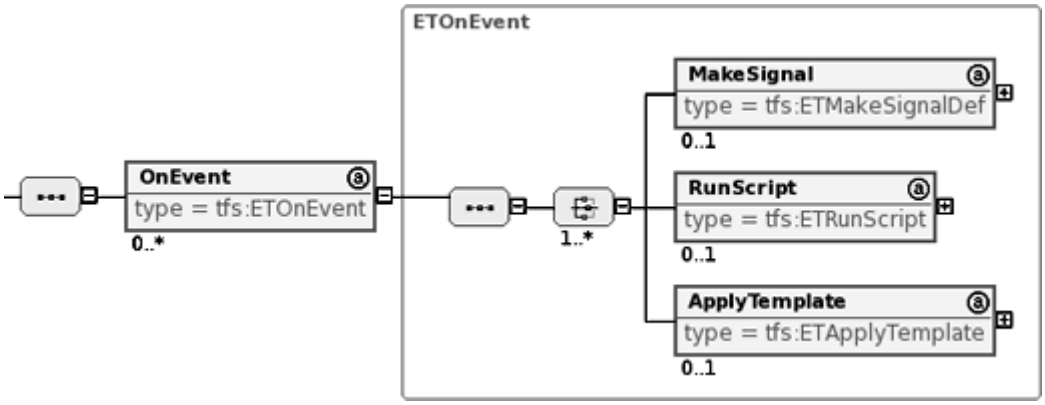


Slika 3.17. Element za parsiranje izlazne datoteke komande

Element *MakeSignal* se koristi za generiranje signala koji se prosljeđuje svim agentima koji očekuju ovaj signal. Signal se definira atributima za:

- tip signala – *type*
  atribut može poprimiti vrijednosti *info*, *error*, *fatal*. Vrijednost *info* se koristi ukoliko se želi poslati običan signal, *error* ukoliko se želi izvjestiti o grešci, te *fatal* ukoliko se želi dojaviti sustavu da je došlo do greške koja uzrokuje zaustavljanje izvođenja testa
- naziv signala – *name*
  identifikator signala koji će biti generiran

Nadalje, za komandu je također moguće definirati akcije koje se trebaju izvršiti prilikom hvatanja događaja generiranog komandom. U *Actions* podelementu elementa *Batch* se nalazi lista događaja koji se očekuju izvršavanjem komande. Akcije se izvršavaju prilikom poruke komande da se događaj dogodio. *OnEvent* podelementi su dani slikom 3.18.



Slika 3.18. Podelementi za definiranje akcije

Primjer *Actions* elementa je dan idućim odsječkom XML-a:

```
.
.
<Actions>
  <OnEvent name="START">
    <MakeSignal type="info" name="wget_started" />

    <RunScript script="S_inc_counter">
      <Connect>
        <In Par="cur_count" Var="self.V_num_down"/>
        <Out Ret="inc_count" Var="self.V_num_down"/>
      </Connect>
    </RunScript>
  </OnEvent>

  <OnEvent name="END">
    <MakeSignal type="info" name="wget_finished" />

    <RunScript script="S_continue">
      <Connect>
        <In Par="num_down" Var="self.V_num_down"/>
        <Out Ret="continue" Var="self.V_continue"/>
      </Connect>
    </RunScript>
  </OnEvent>

  <OnEvent name="DIE">
    <MakeSignal type="fatal" name="wget_died" />
  </OnEvent>
</Actions>
.
.
```

Moguće je izvršiti skriptu, kreirati signal i primjeniti predložak.

Za komandu je moguće definirati koliko se puta mora izvršiti pomoću podelementa *RepeatWhile* i *Repeat*. Kod prvog pristupa predaje se ime varijable koju se provjerava nakon jednom izvršene komande. Za znakovne varijable niz mora biti prazan da bi se završilo izvršavanje. Za brojevne varijable vrijednost varijable mora biti jednaka nuli.

Primjer korištenja je prikazan idućim odsječkom:

```
.
.
<RepeatWhile Var="V_continue" />
</Command>
```

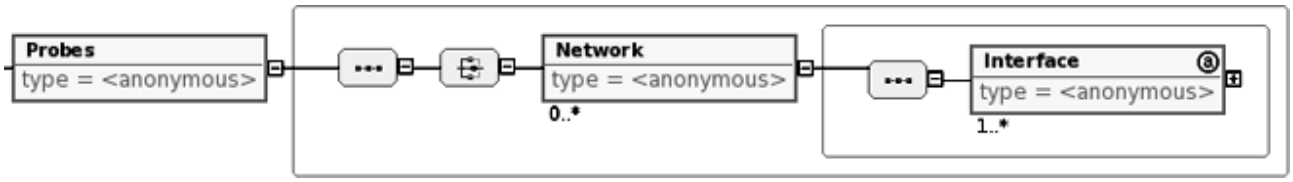
U odsječku je prikazan *RepeatWhile* element koji prekida ponavljanje izvršavanja komande kada vrijednost cjelobrojne varijable *V_continue* postane nula.

Također, moguće je definirati akcije koje se trebaju izvršiti na kraju ili početku *Batch*-a njegovim podelementom *Actions* prikazanim u idućem odsječku XML koda:

```
<Actions>
  <OnBatchEvent name="END">
    <MakeSignal type="info" name="wget_downloads_finished" />
  </OnBatchEvent>
</Actions>
```

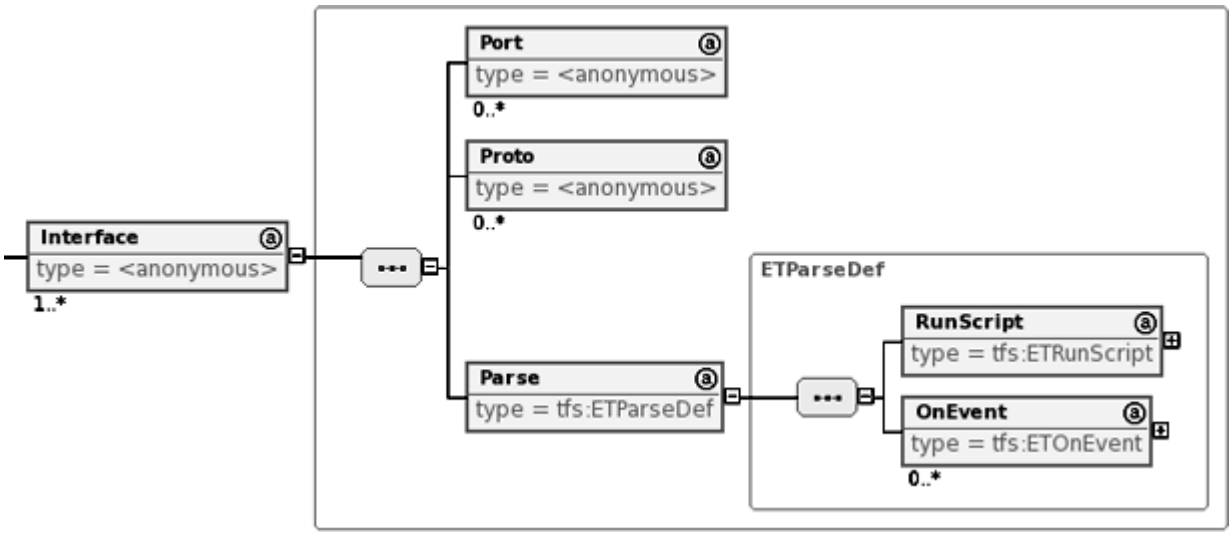
Da bi se razdvojio ovaj *Actions* element od *Actions* elementa za komandu, koriste se podelementi tipa *OnBatchEvent*. Nazivi i značenje atributa, te akcije koje je moguće pokrenuti su identični *Actions* elementu za definiranje akcija na događaje generirane komandom.

Posljednji podelement *Batch* elementa je *Probes*. Ovaj element omogućuje postavljanje sonde na računalu domaćinu koja će na predefiniran način davati tok podataka o mjerenoj vrijednosti. Za sad je predviđen samo *Network* tip sonde, te je element grafički prikazan na slici 3.19.



Slika 3.19. Element za definiranje monitora

U *Interface* podelementu moguće je definirati filter kojim će se definirati parametri željenog prometa, te su podlementi detaljno prikazani na slici 3.20.



Slika 3.20. Element za kreiranje filtera

Također se *Parse* elementom definira skripta za parsiranje toka podataka kojom se može vršiti ispitivanje mrežnog prometa. Za definiranje akcija na generirane događaje koristi se lista *OnEvent* elemenata.

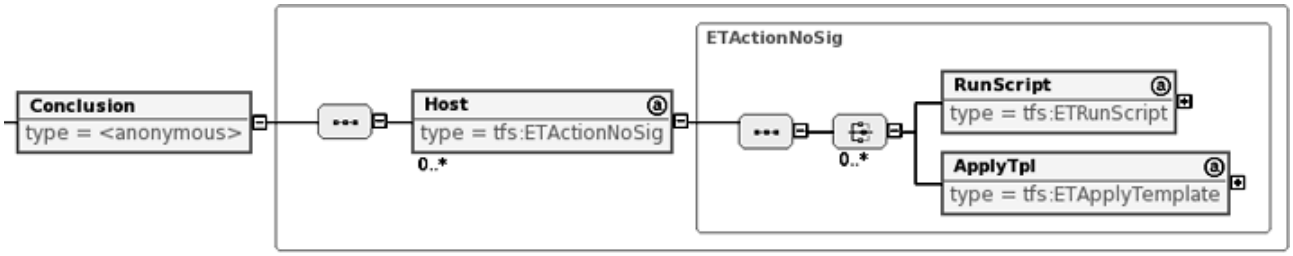
Primjer korištenja *Probes* elementa u XML-u:

```
<Probes>
  <Network>
    <Interface name="eth0">
      <Port number="8080"/>
      <Proto name="TCP"/>
      <Parse signal="wget_downloads_finished" from="any">
        <RunScript script="S_check_headers">
          <Connect>
            <In Par="IP" Var="V_SERVER_IP" />
            <In Par="Port" Var="V_SERVER_PORT" />
            <In Par="Path" Var="V_FILE_NAME" />
          </Connect>
        </RunScript>
      </Parse>
    </Interface>
  </Network>
</Probes>
```

Za kreiranje filtera može se definirati mrežno sučelje, port i protokol s odgovarajućim elementima *Interface*, *Port*, te *Proto*. Podelement *Parse* je identičan kao kod elementa *Parse* pri definiranju parsiranja izlaza komande.

3.3.3. Zaključenje testa

Conclusion dio se koristi kako bi se izvršili neki krajnji koraci nakon samog izvršavanja testa. Moguće je pokretati skripte koje obavljaju posao zaključenja testa. Primjena može biti recimo da bi se počistila okolina brisanjem nepotrebnih datoteka ili vršenje finalne analize rezultata testa. Grafički prikaz elemenata je dan slikom 3.21.



Slika 3.21. Elementi zaključenja testa

Primjer korištenja elementa *Conclusion* je dan sljedećim odsječkom koda:

```
<Conclusion>
  <Host name="server">
    <RunScript script="S_server_cleanup" />
    <RunScript script="S_analyze_log" />
  </Host>

  <Host name="drone1">
    <RunScript script="S_client_cleanup" />
  </Host>

  <Host name="drone2">
    <RunScript script="S_client_cleanup" />
  </Host>
</Conclusion>
```

Definiraju se računala, pa onda skripte koje je potrebno na njima izvršiti. Odsječkom je definirano pokretanje skripte *S_client_cleanup* na domaćinima *drone1* i *drone2*, te skripti *S_server_cleanup* i *S_analyze_log* na domaćinu *server*.

4. Parser

Prethodno definiran XML je potrebno učitati (de-serijalizirati) u objekte kako bi se jednostavnije koristili pri daljnem radu ispitnog okruženja. Ispitno okruženje će biti izvedeno programskim jezikom Python, tako da je i parser izveden u Python-u.

4.1. Python implementacija

U ovom poglavlju bit će ukratko pojašnjen način rada parsera, te kako ga koristiti kao python modul.

Za potebe učitavanja podataka iz XML datoteke koristi se izrađena klasa *TFTest.TFDOM.tfdom* koja se oslanja na *xml.dom.minidom* klasu za samo čitanje iz XML-a. Razlog kreiranja *tfdom* klase koja enkapsulira *minidom* je u tome što *minidom* tretira nizove znakova između dvaju početaka elemenata kao čvor s tekstualnim sadržajem. To uvelike otežava rad s XML-om prilikom učitavanja podataka u ovom slučaju jer ne postoji koristan sadržaj u prethodno definiranom XML-u na ovim pozicijama.

Jedino mjesto koje ima tekstualni čvor općenito u XML-u je *Description* element. Pri učitavanju podataka *minidom*-om se zanemaruju svi tekstualni čvorovi osim čvora unutar *Description* elementa.

Ostali elementi se spremaju u polje objekata istim redosljedom kako se pojavljuju u XML-u. Atributi se spremaju u asocijativno polje koje koristi naziv atributa za ključ polja, a vrijednost atributa za vrijednost polja.

Nakon učitavanja kompletnog XML-a, *minidom* objekt se uništava.

Kreirane metode ove klase su identične podskupu metoda *minidom* klase po imenu i funkcionalnosti.

Učitan objekt *tfdom* klase se koristi unutar *TFTest.TFLoader* modula za popunjavanje objekata s podacima iz XML-a.

Kod *tfdom* klase:

```
from xml.dom import minidom, Node
class tfdom:
    """Simple DOM implementation

    Uses xml.dom.minidom to build basic DOM
    then fills this simple DOM.
    """
    def __init__( self, xml = None ):
        if xml is not None:
            self.load_from_xml( xml )

    def load_from_xml( self, xml ):
        self.name = ''
        self.tagName = self.name
        self.elements = []
        self.childNodes = self.elements
        self.attributes = {}
        self.text = ''
        if xml is None:
            return
        self.name = xml.tagName

        #attributes
        for att in xml.attributes.items():
            self.attributes[ att[0] ] = att[1]

        # text
        if self.name == 'Description':
            self.text = xml.firstChild.data
            return
        for node in xml.childNodes:
            if node.nodeType == Node.ELEMENT_NODE:
                n = tfdom( node )
                self.elements.append( n )

    def parseFile( self, filename ):
        xmldoc = minidom.parse( filename )
        root = xmldoc.documentElement
        self.__init__( root )
        xmldoc.unlink()

    def getChildrenByTagName( self, name ):
        ret = []
        for el in self.elements:
            if el.name == name:
                ret.append( el )
        return ret

    def getAttribute( self, name ):
        if self.attributes.has_key( name ):
            return self.attributes[name]
        else:
            return ''

    def hasAttribute( self, name ):
        return self.attributes.has_key( name )
```

4.1.1. Modul TFTest.TFLoader

Svrha ovog modula je učitavanje strukturiranih podataka iz XML datoteke u objekte koji će se koristiti za postavljanje i izvođenje testa u ispitnom okruženju.

Hijerarhija objekata je preuzeta iz definiranog XML-a.

4.1.2. Primjer korištenja

U sljedećem primjeru kreiran je Python program koji učitava XML datoteku u *TFTest* objekt i ipisuje osnovne podatke o testu.

```
from TFTest import TFLoader

def basic_info( xml_path ):
    tf_test = TFLoader.load_test( xml_path )
    print "Test ID:", tf_test.id
    print "Test version:", tf_test.version
    print "Test description: ", tf_test.info.desc
    print "SUBTESTS:"
    for test_id in tf_test.tests.keys():
        one_test = tf_test.tests[test_id]
        print 'Test', test_id, "(", one_test.desc, ")"

if __name__ == '__main__':
    basic_info( 'xml/wget-v0.1.xml' )
```

Rezultat izvođenja programa je sljedeći:

```
$ python ./info.py
Test ID: HttpTest
Test version: 0.8
Test description:  Simple test that tests thttpd and wget
SUBTESTS:
Test Scenarij1 ( This one tests basic functionality )
$
```

5. Zaključak

Definiranje XML-a za opis testa je poduhvat kojim se pokušalo odgovoriti na postavljene zahtjeve, te definirati format koji bi dozvolio slobodu izraza, te široku primjenu na podrčju testiranja mrežnih aplikacija.

Kreiran XML je jednostavan za generiranje i održavanje radi naziva elemenata i atributa koji su vrlo bliski funkciji koju opisuju. Novi korisnik može vrlo jednostavno se naviknuti na format datoteke radi prirodnih riječi korištenih za elemente, te brzo napisati jednostavan test za razvijanu aplikaciju.

Za naprednije korištenje potrebno je dobro razmisliti o potrebama, te izgraditi podršku za odvijanje testa putem skripti koje će obavljati posao. XML u kombinaciji s ispitnim okruženjem ne pomaže u segmentu dizajniranja testa, već samo olakšava postavljanje i automatsko izvođenje testa, a ispitno okruženje bi trebalo ponuditi vođenje dnevnika o svim događajima koji su se zbili, signalima koji su poslani, stanju varijabli i sl. što bi trebalo stvoriti dobar temelj za analizu ispitivane aplikacije.

Naravno, nisu pokriveni svi mogući scenariji testiranja ovim XML-om, no postavljen je dobar temelj za daljnje unapređenje i proširivanje funkcionalnosti ukoliko se pokaže potreba. Mogući nedostatci u dizajnu će biti uočeni i ispravljeni tijekom implementacije ispitnog okruženja pri ćemu će XML služiti kao nit vodilja za ostvarenje fleksibilnog ispitnog okruženja za ispitivanje mrežnih aplikacija.

6. Literatura

1. Extensible Markup Language (XML), URL: http://www.w3.org/XML/ (30/09/06)
2. W3C XML Schema, URL: http://www.w3.org/XML/Schema (30/09/06)
3. XML Schema Tutorial, URL: http://www.w3schools.com/schema/default.asp (30/09/06)
4. Event-driven programming - Wikipedia, the free encyclopedia, URL: http://en.wikipedia.org/wiki/Event-driven_programming(02/10/06)
5. Python Documentation, URL: http://docs.python.org/ (30/09/06)
6. xml.dom.minidom -- Lightweight DOM implementation, URL: http://docs.python.org/lib/module-xml.dom.minidom.html(30/09/06)
7. Python/XML HOWTO, URL: http://pyxml.sourceforge.net/topics/howto/xml-howto.html (30/09/06)
8. Python and XML: An Introduction, URL: http://www.boddie.org.uk/python/XML_intro.html (30/09/06)
9. Pydev, URL: http://pydev.sourceforge.net/ (30/09/06)
10. Eclipse Tools - XSD Home, URL: http://www.eclipse.org/xsd/ (30/09/06)
11. Pawel Leszek, XML development with Eclipse, URL: http://www-128.ibm.com/developerworks/library/os-ecxml/ (30/09/06)
12. David Gallardo, Getting started with the Eclipse Platform, URL: http://www-128.ibm.com/developerworks/library/os-ecov/ (30/09/06)

Dodatak – Ispis korištenog XML primjera

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<tfs:TFTest
  TFTestID="HttpTest"
  version="0.2" date="2006-08-29"
  xmlns:tfs="TFSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="TFSchema tfsch.xsd" >
  <TFTestInfo>
    <Description>Simple test that tests thttpd and wget</Description>
    <Author
      name="Jure"
      surname="Rastic"
      url="http://osl.zemris.fer.hr/"
      email="jure.rastic@gmail.com"/>

    <TFEngine version="0.1"/>
        <DependsOn />
  </TFTestInfo>

  <Definitions>
    <Variables>
      <Variable VarID="V_SERVER_IP" type="string" value="10.0.0.2" />
      <Variable VarID="V_SERVER_PORT" type="string" value="8080" />
      <Variable VarID="V_FILE_NAME" type="string" value="image.iso" />
      <Variable VarID="V_HTDOCS_FOLDER" type="string" value="./htdocs/" />
      <Variable VarID="V_file_sum" type="string" value="unknown" />
    </Variables>

    <Scripts>
      <Script ScriptID="S_calc_checksum" type="calculate" lang="python">
        <SrcPath type="local" path="scripts/calc_checksum.py" />
        <DestPath path="scripts/calc_checksum.py" />
        <IO>
          <Input>
            <Par ParID="file" index="1"/>
          </Input>

          <Output>
            <Ret RetID="sum" index="1"/>
          </Output>
        </IO>
      </Script>
      <Script ScriptID="S_wget_starter" type="execute" lang="shell">
        <SrcPath type="local" path="scripts/wget_starter.sh" />
        <DestPath path="scripts/wget_starter.sh" />
        <IO>
          <Input>
            <Par ParID="IP" index="1"/>
            <Par ParID="Port" index="2"/>
            <Par ParID="Path" index="3"/>
          </Input>
          <Output />
        </IO>
      </Script>
```

```
<Script ScriptID="thttpd_starter" type="execute" lang="shell">
  <SrcPath type="local" path="scripts/thttpd_starter.sh" />
  <DestPath path="scripts/thttpd_starter.sh" />
</Script>

<Script ScriptID="S_checksum" type="parse" lang="python">
  <SrcPath type="local" path="scripts/checksum.py" />
  <DestPath path="scripts/checksum.py" />
  <IO>
    <Input>
      <Par ParID="orig_sum" index="1"/>
    </Input>
    <Output />
  </IO>
</Script>

<Script ScriptID="S_inc_counter" type="calculate" lang="python">
  <SrcPath type="local" path="scripts/inc_counter.py" />
  <DestPath path="scripts/inc_counter.py" />
  <IO>
    <Input>
      <Par ParID="cur_count" index="1"/>
    </Input>
    <Output>
      <Ret RetID="inc_count" index="1"/>
    </Output>
  </IO>
</Script>

<Script ScriptID="S_continue" type="calculate" lang="python">
  <SrcPath type="local" path="scripts/calc_continue.py" />
  <DestPath path="scripts/calc_continue.py" />
  <IO>
    <Input>
      <Par ParID="num_down1" index="1"/>
      <Par ParID="num_down2" index="2"/>
    </Input>
    <Output>
      <Ret RetID="continue" index="1"/>
    </Output>
  </IO>
</Script>

<Script ScriptID="S_check_log" type="parse" lang="python">
  <SrcPath type="local" path="scripts/check_thttpd_log.py" />
  <DestPath path="scripts/check_thttpd_log.py" />
  <IO>
    <Input>
      <Par ParID="num_down1" index="1"/>
      <Par ParID="num_down2" index="2"/>
    </Input>
    <Output />
  </IO>
</Script>
```

```
    <Script ScriptID="S_check_headers" type="parse" lang="python">
      <SrcPath type="local" path="scripts/check_headers.py" />
      <DestPath path="scripts/check_headers.py" />
      <IO>
        <Input>
          <Par ParID="IP" index="1"/>
          <Par ParID="Port" index="2"/>
          <Par ParID="Path" index="3"/>
        </Input>
        <Output />
      </IO>
    </Script>

    <Script ScriptID="S_server_cleanup" type="execute" lang="shell">
      <SrcPath type="local" path="scripts/server_cleanup.sh" />
      <DestPath path="scripts/server_cleanup.sh" />
    </Script>

    <Script ScriptID="S_client_cleanup" type="execute" lang="shell">
      <SrcPath type="local" path="scripts/client_cleanup.sh" />
      <DestPath path="scripts/client_cleanup.sh" />
    </Script>

    <Script ScriptID="S_analyze_log" type="parse" lang="python">
      <SrcPath type="local" path="scripts/analyze_thttps_log.py" />
      <DestPath path="scripts/analyze_thttps_log.py" />
    </Script>
  </Scripts>

  <Files>
    <File FileID="F_served_file">
      <SrcPath type="local" path="files/image.iso" />
      <DestPath path="htdocs/image.iso" />
    </File>
    <File FileID="F_thttp_conf">
      <SrcPath type="local" path="confs/thttpd.conf" />
      <DestPath path="thttpd.conf" />
    </File>
  </Files>

  <Templates>
    <Template TplID="T_thttp_conf">
      <Symbol string="$ip" value="none" SymID="IP" />
      <Symbol string="$port" value="8080" SymID="PORT" />
      <Symbol string="$htdocs" value="./htdocs/" SymID="HTDOCS_FOLDER" />
    </Template>
  </Templates>

  <Applications>
    <Application AppID="wget">
      <Description>internet download client</Description>
      <Info name="wget" version="1.10.2"
        url="http://www.gnu.org/software/wget/" />
      <Author name="Hrvoje" surname="Niksic"/>
      <Repository type="source" >
        <Source type="ftp"
          path="http://ftp.gnu.org/pub/gnu/wget/wget-1.10.2.tar.gz" />

      </Repository>
```

```
        <DependsOn>
          <CompileTime />
          <RunTime />
        </DependsOn>
        <Compile type="beforeDistribution" />
      </Application>

      <Application AppID="thttp">
        <Description>
          thttpd is a simple, small, portable, fast,
          and secure HTTP server.
        </Description>
        <Info name="Tiny httpd" version="2.25"
          url="http://www.acme.com/software/thttpd/" />
        <Author name="Jef" surname="Poskanzer" email="jef@acme.com" />
        <Repository type="source" >
          <Source
            type="http"
            path="http://www.acme.com/software/thttpd/thttpd-2.25b.tar.gz"/>
        </Repository>

        <DependsOn>
          <CompileTime/>
          <RunTime/>
        </DependsOn>

        <Compile type="beforeDistribution" />

      </Application>
    </Applications>

    <Equipment>
      <Computer CompID="server" type="real" >
        <Description>Server computer</Description>
        <Network>
          <Interface
            IfaceID="eth0"
            type="ethernet"
            address="10.0.0.2"
            netmask="255.255.255.0"
            gateway="10.0.0.1"
          />
        </Network>
        <OS type="linux" />
      </Computer>

      <Computer CompID="drone1" type="real" >
        <Description>wget computer-01</Description>
        <Network>
          <Interface
            IfaceID="eth0"
            type="ethernet"
            address="10.0.0.3"
            netmask="255.255.255.0"
            gateway="10.0.0.1"
          />
        </Network>
        <OS type="linux" />
      </Computer>
```

```
      <Computer CompID="drone2" type="real" >
        <Description>wget computer-02</Description>
        <Network>
          <Interface
            IfaceID="eth0"
            type="ethernet"
            address="10.0.0.4"
            netmask="255.255.255.0"
            gateway="10.0.0.1"
          />
        </Network>
        <OS type="linux" />
      </Computer>
    </Equipment>
</Definitions>

<Tests>
  <Test TestID="test-case-01" >
    <Description>This one tests basic functionality</Description>

    <DependsOn>
      <Test name="none" />
    </DependsOn>

    <Preparation>
      <AppDistribution>
        <Application name="wget">
          <Host name="drone1" />
          <Host name="drone2" />
        </Application>

        <Application name="thttpd">
          <Host name="server" />
        </Application>
      </AppDistribution>

      <FileDistribution>
        <File name="F_servedFile">
          <Host name="server" />
        </File>

        <File name="F_thttp_conf">
          <Host name="server" />
        </File>
      </FileDistribution>

      <Initialization>
        <Host name="server">
          <ApplyTpl Tpl="T_thttp_conf" File="F_thttp_conf">
            <Replace Sym="IP" Var="V_SERVER_IP" />
            <Replace Sym="PORT" Var="V_SERVER_PORT" />
            <Replace Sym="HTDOCS_FOLDER" Var="V_HTDOCS_FOLDER" />
          </ApplyTpl>

          <RunScript script="S_calc_checksum">
            <Connect>
              <In Par="file" Var="V_FILE_NAME" />
              <Out Ret="sum" Var="V_file_sum" />
```

```
            </Connect>
          </RunScript>
        </Host>
      </Initialization>
    </Preparation>

    <Execution>
      <Batch BatchID="wget_batch" >
        <Executors>
          <Host name="drone1" />
          <Host name="drone2" />
        </Executors>

        <Variables>
          <Variable VarID="V_num_down" type="integer" value="0" />
          <Variable VarID="V_continue" type="integer" value="1" />
        </Variables>

        <StartsOn>
          <Trigger from="server" signal="server_ready" />
        </StartsOn>

        <EndsOn>
          <Trigger from="server" signal="server_died" />
        </EndsOn>

        <Command type="script" >
          <RunScript script="S_wget_starter">
            <Connect>
              <In Par="IP" Var="V_SERVER_IP" />
              <In Par="Port" Var="V_SERVER_PORT" />
              <In Par="Path" Var="V_FILE_NAME" />
            </Connect>
          </RunScript>

          <ResultFiles>
            <File collect="no" path="image.iso">
              <Parse signal="wget_finished" from="self">
                <RunScript script="S_checksum">
                  <Connect>
                    <In Par="orig_sum" Var="V_file_sum" />
                  </Connect>
                </RunScript>

                <OnEvent name="PASS">
                  <MakeSignal type="info" name="good_download" />
                </OnEvent>

                <OnEvent name="FAIL">
                  <MakeSignal type="error" name="bad_download" />
                </OnEvent>
              </Parse>
            </File>
          </ResultFiles>

          <Actions>
            <OnEvent name="START">
              <MakeSignal type="info" name="wget_started" />
```

```
                <RunScript script="S_inc_counter">
                  <Connect>
                    <In Par="cur_count" Var="self.V_num_down"/>
                    <Out Ret="inc_count" Var="self.V_num_down"/>
                  </Connect>
                </RunScript>
              </OnEvent>

              <OnEvent name="END">
                <MakeSignal type="info" name="wget_finished" />

                <RunScript script="S_continue">
                  <Connect>
                    <In Par="num_down" Var="self.V_num_down"/>
                    <Out Ret="continue" Var="self.V_continue"/>
                  </Connect>
                </RunScript>
              </OnEvent>

              <OnEvent name="DIE">
                <MakeSignal type="fatal" name="wget_died" />
              </OnEvent>

            </Actions>

            <RepeatWhile Var="V_continue" />
          </Command>



          <Actions>
            <OnBatchEvent name="END">
              <MakeSignal type="info" name="wget_downloads_finished" />
            </OnBatchEvent>
          </Actions>

          <Probes />
        </Batch>


        <Batch BatchID="thttpd_batch" >
          <Executors>
            <Host name="server" />
          </Executors>

          <Variables />

          <StartsOn>
            <Trigger from="TFE" signal="TFTestStart" />
          </StartsOn>

          <EndsOn>
            <Trigger from="self" signal="stop_test" />
          </EndsOn>

          <Command type="script" >
            <RunScript script="thttpd_starter" />

            <ResultFiles>
```

```
            <File collect="yes" path="logs/default.log">
              <Parse signal="good_download" from="any">
                <RunScript script="S_check_log">
                  <Connect>
                    <In Par="num_down1" Var="drone1.V_num_down" />
                    <In Par="num_down2" Var="drone2.V_num_down" />
                  </Connect>
                </RunScript>

                <OnEvent name="OK">
                  <MakeSignal type="info" name="good_log" />
                </OnEvent>

                <OnEvent name="WRONG">
                  <MakeSignal type="error" name="bad_log" />
                </OnEvent>
              </Parse>
            </File>
          </ResultFiles>

          <Actions>
            <OnEvent name="DIE">
              <MakeSignal type="fatal" name="server_died" />
            </OnEvent>
          </Actions>
        </Command>

        <Actions>
          <OnBatchEvent name="START">
            <MakeSignal type="info" name="server_started" />
          </OnBatchEvent>

          <OnBatchEvent name="END">
            <MakeSignal type="info" name="stop_test" />
          </OnBatchEvent>
        </Actions>

        <Probes>
          <Network>
            <Interface name="eth0">
              <Port number="8080"/>
              <Proto name="TCP"/>
              <Parse signal="wget_downloads_finished" from="any">
                <RunScript script="S_check_headers">
                  <Connect>
                    <In Par="IP" Var="V_SERVER_IP" />
                    <In Par="Port" Var="V_SERVER_PORT" />
                    <In Par="Path" Var="V_FILE_NAME" />
                  </Connect>
                </RunScript>
              </Parse>
            </Interface>
          </Network>
        </Probes>

      </Batch>
    </Execution>

    <Conclusion>
```

```
          <Host name="server">
            <RunScript script="S_server_cleanup" />
            <RunScript script="S_analyze_log" />
          </Host>

          <Host name="drone1">
            <RunScript script="S_client_cleanup" />
          </Host>

          <Host name="drone2">
            <RunScript script="S_client_cleanup" />
          </Host>
        </Conclusion>
      </Test>
    </Tests>
</tfs:TFTest>
```