



# УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
НОВИ САД  
Департман за рачунарство и аутоматику  
Одсек за рачунарску технику и рачунарске комуникације

## ЗАВРШНИ (BACHELOR) РАД

Кандидат: Бранислав Кордић  
Број индекса: е12657

Тема рада: Статистичко тестирање програма помоћу алата MaTeLo

Ментор рада: проф. др Мирослав Поповић

Нови Сад, јун, 2012



## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Кордић Бранислав		
Ментор, МН:	др Поповић Мирослав, ред. проф.		
Наслов рада, НР:	Статистичко тестирање програма помоћу алате MaTeLo		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2012		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страна/цитата/табела/слика/графика/прилога)	7/58/4/6/22/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	Статистичко тестирање, MaTeLo, поузданост, средње време до отказа, синтаксни анализатор		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	Статистичко тестирање има доминантну улогу у процесу израде програма. У раду је описан процес статистичког тестирања, одређивање поузданости програма и средњег времена до отказа. За стварање тест случајева и рачунање поменутих карактеристика коришћен је алат MaTeLo. Анализом добијених резултата утврђено је да алат MaTeLo даје објективне процене тестираног програма.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	др Иштван Пап, доц.	
	Члан:	др Ковачевић Јелена, доц.	Потпис ментора
	Члан, ментор:	др Поповић Мирослав, ред. проф.	



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO:</b>			
Identification number, <b>INO:</b>			
Document type, <b>DT:</b>	Monographic publication		
Type of record, <b>TR:</b>	Textual printed material		
Contents code, <b>CC:</b>	Bachelor Thesis		
Author, <b>AU:</b>	<b>Branislav Kordić</b>		
Mentor, <b>MN:</b>	<b>Popović Miroslav, PhD</b>		
Title, <b>TI:</b>	<b>Statistical testing using MaTeLo tool</b>		
Language of text, <b>LT:</b>	Serbian		
Language of abstract, <b>LA:</b>	Serbian		
Country of publication, <b>CP:</b>	Republic of Serbia		
Locality of publication, <b>LP:</b>	Vojvodina		
Publication year, <b>PY:</b>	<b>2012</b>		
Publisher, <b>PB:</b>	Author's reprint		
Publication place, <b>PP:</b>	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, <b>PD:</b> (chapters/pages/ref./tables/pictures/graphs/appendices)	<b>7/58/4/6/22/0/0</b>		
Scientific field, <b>SF:</b>	Electrical Engineering		
Scientific discipline, <b>SD:</b>	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, <b>S/KW:</b>	<b>Statistical testing, MaTeLo, reliability, mean time to failure, parser</b>		
<b>UC</b>			
Holding data, <b>HD:</b>	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, <b>N:</b>			
Abstract, <b>AB:</b>	<p><b>Statistical testing plays a dominant role in software development. In the thesis the process of statistical testing is described, as well as the software reliability and the mean time to failure. To create test cases and the computation of these characteristics MaTeLo tool has been used. The results obtained show that MaTeLo tool provides an objective assessment of the test program.</b></p>		
Accepted by the Scientific Board on, <b>ASB:</b>			
Defended on, <b>DE:</b>			
Defended Board, <b>DB:</b>	President:	Ištván Pap, PhD	
U	Member:	Jelena Kovečević, PhD	Menthor's sign
	Member, Mentor:	Miroslav Popović, PhD	

## **Zahvalnica**

Srdačno se zahvaljujem svom mentoru prof. dr Miroslavu Popoviću na stručnoj pomoći, razumevanju i savetovanju tokom izrade završnog (*bachelor*) rada.

Takođe se zahvaljujem asistentu Vladimиру Marinkoviću na stalnoj podršci i razumevanju tokom izrade rada.

## SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove .....	3
2.1	MaTeLo Usage Model Editor .....	4
2.1.1	Stanja sistema .....	5
2.1.2	Prelazi iz trenutnog u naredno stanje sistema .....	5
2.1.3	Ulazi u sistem .....	7
2.1.4	Broj aktivacija sistema .....	8
2.1.5	Informacije i scenariji korišćenja sistema .....	9
2.2	MaTeLo Testor.....	11
2.2.1	Stvaranje apstraktnih test slučajeva.....	11
2.2.2	Analiza sistema na osnovu dobijenih rezultata .....	15
2.3	MaTeLo Convertor.....	19
2.3.1	Prevođenje apstraktnih test slučaja.....	20
2.4	Statističko testiranje zadatog programa.....	21
2.4.1	Model korišćenja zadatog programa .....	21
2.4.2	Stvaranje test slučajeva .....	24
2.4.3	Platforma i okruženje za automatsko izvršavanje test slučajeva.....	26
3.	Koncept rešenja.....	27
3.1	Sintaksni analizator u MaTeLo skupu alata .....	27
3.2	Arhitektura sintaksnog analizatora.....	29
3.2.1	Sintaksni analizator datoteke sa ulaznim parametrima sistema .....	30
3.2.1.1	Datoteka sa rezultatima testova.....	30
3.2.1.2	Verzija sistema.....	30
3.2.1.3	Broj grupe testova .....	31

---

3.2.1.4	Ulagana xml datoteka.....	31
3.2.1.5	Direktorijum izlazne xml datoteke.....	31
3.2.1.6	Ispravno definisana datoteka sa ulaznim parametrima .....	32
3.2.2	Sintaksni analizator datoteke sa rezultatima testova .....	32
3.2.2.1	Zaglavje rezultata.....	33
3.2.2.2	Ime testnog slučaja.....	33
3.2.2.3	Rezultat testiranja.....	33
3.2.2.4	Korak na kome se desila greška.....	34
3.2.2.5	Vreme izvršenja testa .....	34
3.2.2.6	Ispravno definisana datoteka sa rezultatima testova .....	35
3.2.3	DOM Sintaksni analizator .....	36
3.2.3.1	Čvor zbirke slučaja.....	36
3.2.3.2	Čvor test slučaja .....	37
3.2.3.3	Čvor prelaza iz tekućeg u naredno stanje.....	38
4.	Programsko rešenje .....	39
4.1	Config klasa.....	40
4.1.1	configFile atribut .....	41
4.1.2	testResultFile atribut.....	41
4.1.3	systemVersion atribut.....	41
4.1.4	testSuiteNumber atribut.....	42
4.1.5	inputXmlFile atribut .....	42
4.1.6	outputXmlDir atribut.....	42
4.1.7	outputFilePathForMaTeLo metoda .....	42
4.2	TestResult klasa.....	42
4.2.1	testCaseName atribut.....	43
4.2.2	testVerdict atribut .....	43
4.2.3	testStepError atribut .....	43
4.2.4	testExecutionTime atribut .....	43
4.3	ConfigFileParser modul .....	43
4.3.1	int ParseConfigFile (Config * config).....	43
4.4	TestResultParser modul.....	44
4.4.1	int ParseTestResult (fstream * file, list< TestCaseResult > * list) .....	44
4.5	DOMSyntaxAnalyzer modul.....	44
4.5.1	int InitializeDOMSyntaxAnalyzer (void).....	45
4.5.2	bool ParseXMLFile (XercesDOMParser * parser, string inXmlFile).....	45

4.5.3	int SaveXML (DOMDocument * doc, string outXmlFile) .....	45
4.5.4	DOMELEMENT* GetDOMELEMENT (DOMELEMENT * root, string elementName)	
	45	
4.5.5	void GetAllDOMELEMENT (DOMELEMENT * root, string elementName, list<	
	DOMELEMENT * > * list) .....	46
4.5.6	int GetDOMELEMENTCount (DOMELEMENT * root, string elementName).....	46
4.5.7	void FillSuiteNod (DOMELEMENT * suite, list< TestCaseResult > * results,	
	Config *config).....	46
4.5.8	void FillTestCaseNod (list<DOMELEMENT*> *domElementList,	
	list<TestCaseResult> *results) .....	46
5.	Rezultati .....	48
5.1	Rezultati testiranja.....	48
5.2	Rezultati analize podataka dobijenih u toku testiranja.....	49
5.2.1	Pouzdanost i MTTF za svaku verziju programa.....	50
5.2.2	Kumulativna pouzdanost i MTTF .....	53
6.	Zaključak .....	56
7.	Literatura.....	58

## SPISAK SLIKA

Slika 2.1 – Arhitektura MaTeLo alata .....	3
Slika 2.2 – MaTeLo Usage Model Editor .....	4
Slika 2.3 – Verovatnoće prelaza u modelu korišćenja .....	7
Slika 2.4 – Ugrađeni tipovi podataka.....	8
Slika 2.5 – Podešavanje broja aktivacija sistema.....	9
Slika 2.6 – Primer formalne definicije scenarija korišćenja upotrebom UML jezika.....	10
Slika 2.7 – MaTeLo Testor, strategije stvaranja test slučaja.....	12
Slika 2.8 – Stvaranje zbirke apstraktnih test slučajeva .....	14
Slika 2.9 – MaTeLo Testor, unos i analiza dobijenih rezultata .....	16
Slika 2.10 – Parametri test kampanje, Milerovi koeficijenti.....	19
Slika 2.11 – MaTeLo Convertor, prevođenje apstraktnih u izvršive test slučajeve.....	20
Slika 2.12 – Blok šema procesa statističkog testiranja zadatog programa.....	21
Slika 2.13 – Zadati program, Babel Fish prevodioč.....	22
Slika 2.14 – Model korišćenja zadatog programa.....	23
Slika 3.1 – Tok podataka u MaTeLo alatu sa ručnim unosom rezultata.....	28
Slika 3.2 – Tok podataka u MaTeLo alatu sa upotrebom sintasnog analizatora .....	29
Slika 3.3 – Blok šema namenskog MaTeLo sintaksnog analizatora.....	29
Slika 5.1 – Trend otkrivanja grešaka u toku testiranja.....	49
Slika 5.2 – Pouzdanost i MTTF programa (1 aktivacija).....	50
Slika 5.3 – Pouzdanost i MTTF programa (2 aktivacije).....	51
Slika 5.4 – Kumulativna pouzdanost i kumulativni MTTF programa (1 aktivacija).....	53
Slika 5.5 – Kumulativna pouzdanost i kumulativni MTTF programa (2 aktivacije).....	54

## SPISAK TABELA

Tabela 2.1 – Korišćene reči u toku testiranja.....	24
Tabela 2.2 – Zbirke test slučajeva za testiranje Babel Fish programa.....	25
Tabela 4.1 – Datoteke programskog rešenja.....	40
Tabela 5.1 – Rezultati testiranja .....	48
Tabela 5.2 – Verovantoča korišćenja, pouzdanost i MTTF za 1 i 2 aktivacije .....	51
Tabela 5.3 – Kumulativna pouzdanost i kumulativni MTTF za 1 i 2 aktivacije .....	55

## SKRAĆENICE

**MaTeLo** - *Markov Test Logic*, Skup alata za statističko testiranje

**MTTF** - *Mean Time to Failure*, Srednje vreme do otkaza

**BBT** - *Black Box Testing*, Testiranje zasnovano na modelu crne kutije

**API** - *Application Interface*, Aplikativna sprega

**MMI** - *Man Machine Interface*, Sprega između mašine i čoveka

**DOM** - *Document Object Model*, Dokument u objektnoj predstavi

**XML** - *Extensible Markup Language*, Jezik za označavanje podataka

**XSLT** - *Extensible Markup Language Style Sheet*, Šema za izmenu XML sadržaja

**PDF** - *Portable Digital Format*, Prenosivi digitalni dokument

**MCM** - *Markov Chain Model*, Model Markovljevih lanaca

**MSC** - *Message Sequence Chart*, Grafik razmene poruka

**UML** - *Unified Modeling Language*, Jezik za unificirano modeliranje

**SD** - *Sequence Diagram*, Dijagram sekvenci

**DLL** - *Dynamic Linking Library*, Biblioteka za dinamičko povezivanje

## 1. Uvod

U ovom radu opisana je metodologija statističkog testiranja (eng. statistical usage testing) i jedna njena primena na zadatom programu na osnovu zadatog operativnog profila upotreboom programskog alata MaTeLo. Na osnovu zadatog operativnog profila programa potrebno je prvo napraviti zbirku apstraktnih test slučajeva (eng. test suite) i proveriti njen kvalitet. Nakon provere kvaliteta zbirke apstraktnih test slučajeva potrebno je prevesti dobijene apstraktne test slučajeve u izvršive test slučajeve (eng. test case) prilagođene nekoj od platformi namenjenih za automatsko izvršavanje (eng. test bed). Zadati program je potrebno testirati prevedenom zbirkom test slučaja i prikupiti dobijene rezultate. Na osnovu dobijenih rezultata testiranja procenjuje se pouzdanost (eng. reliability) sistema i srednje vreme do otkaza (eng. mean time to failure). Pored toga, neophodno je realizovati programski alat za automatsko ažuriranje rezultata apstraktnih test slučaja. Na kraju, treba definisati i način integracije ovog rešenja statističkog testiranja u sistem BBT razvijen u Institutu RT-RK, Novi Sad.

Programski skup alata MaTeLo namenjen je za statističko testiranje korišćenja sistema. Alat je razvijen kao zajednički Evropski projekat u kome je učestvovalo šest saradnika iz industrije i dva univerziteta [1]. MaTelo svoj rad zasniva na upotrebi logike Markovljevih lanaca poznatih iz teorije verovatnoće. Njegova svrha je da stvara test slučaje za sistem opisan modelom verovatnoća [2].

Skup alata MaTeLo čine tri pojedinačna višefunkcionalna alata. Ova tri alata zajedno omogućuju formiranje modela korišćenja sistema (eng. usage model) koji se želi testirati, automatsko stvaranje apstraktnih test slučaja, prevođenje apstraktnih test slučaja u izvršive test slučaje i analizu dobijenih rezultata (eng. test campaign analysis). Pod analizom se podrazumeva određivanje informacije kao što je pouzdanost sistema, pokrivenost zahteva ili srednjeg vremena do otkaza. Za analizu sistema potrebno je uneti veliki broj podataka koje su prikupljeni u toku

izvršavanja zbirke testa slučaja kao što je konačni rezultat test slučaja (eng. verdict), vreme izvršenja ili, ukoliko test nije uspešno izvršen, korak na kome se desila greška, itd. Činjenica da postoji veliki broj podataka koje je potrebno uneti kako bi dobijene porocene bile relevantne, ukazuje na to da je proces unosa rezultata dugotrajan, naporan i podložan greškama. Kako MaTeLo nema namenski alat za automatsko ažuriranje rezultata koji bi ujedno ubrzao proces ažuriranja rezultata i smanjio verovatnoću pojave greške, u narednim poglavljima je opisan sintaksni analizator (eng. parser) koji pruža tu mogućnost.

U drugom poglavlju, Teorijske osnove, detaljno je opisan skupa alata MaTeLo kao i njihov način upotrebe u procesu statističkog testiranja. Nakon opisa mogućnosti alata na osnovu zadatog modela korišćenja programa konkretno je opisan proces testiranja upotrebotom alata MaTeLo sa svim pojedinostima na koje treba obratiti pažnju.

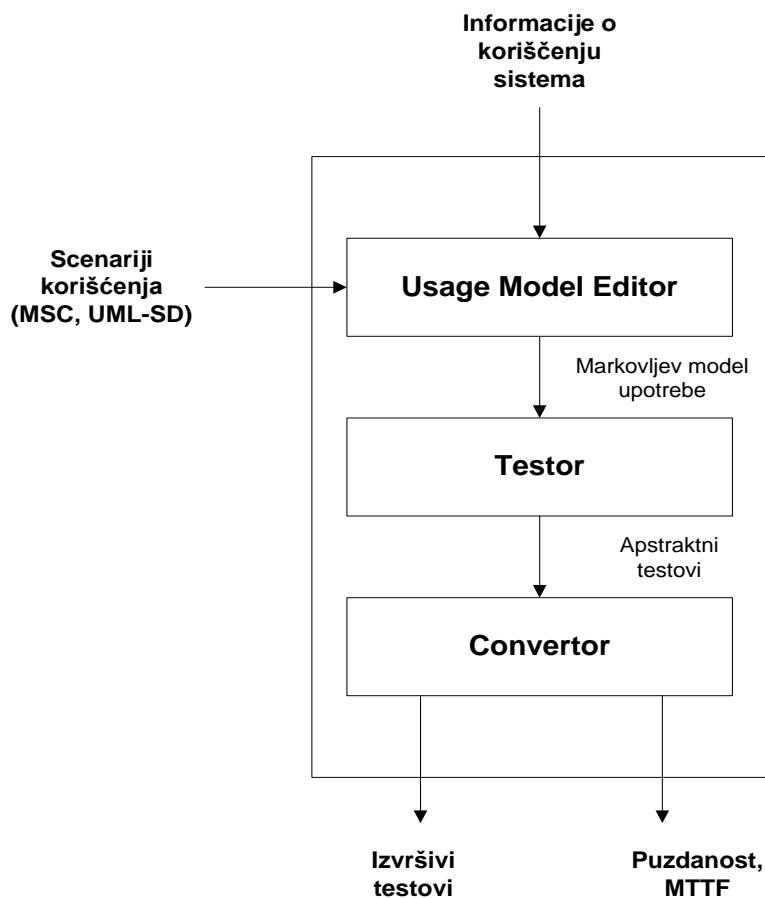
U trećem poglavlju je dat koncept rešenja i arhitektura sintaksnog analizatora. Detaljno je opisana funkcionalnost, namena svih gradivnih blokova kao i način upotrebe sintaksnog analizatora u skupu alata MaTeLo. Analizirane su sve datoteke koje su neophodne za rad sistema. Takođe, priloženi su primeri ispravno popunjeneih datoteka.

U četvrtom poglavlju je priložen spisak svih h,.hpp, i.cpp datoteka, detaljan opis API-a kao i ostale pojedinosti koje se odnose programsku realizaciju sintaksnog analizatora.

U poslednjem poglavlju *Rezultati* predstavljeni se dobijeni rezultati testiranja programa, odnosno dobijene procene pouzdanosti programa i srednjeg vremena do otkaza.

## 2. Teorijske osnove

Ovo poglavlje se bavi arhitekturom MaTeLo alata za statističko testiranje. Dat je detaljan opis alata koji čine MaTeLo kao i njihove funkcije. Zbog svoje višefunkcionalnosti, opisane su samo najznačajnije funkcije svakog od alata. Takođe je predstavljen zadati model korišćenja zadatog programa koji je potrebno statistički testirati.



Slika 2.1 – Arhitektura MaTeLo alata

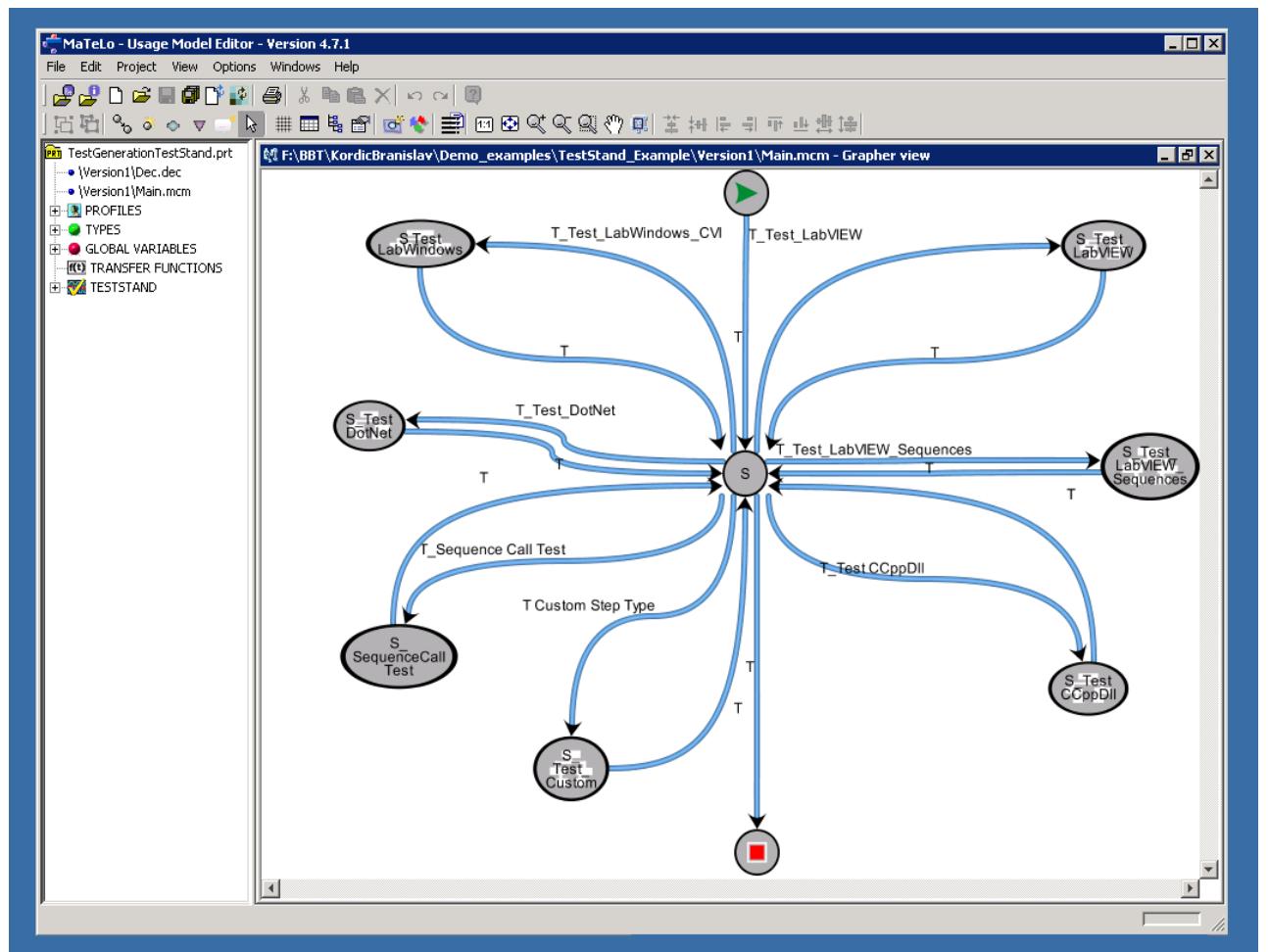
MaTeLo skup alata, kao što se može videti na predhodnoj slici (Slika 2.1), sastoji se iz tri pojedinačna alata, a to su:

- MaTelo Usage Model Editor
- MaTeLo Testor
- MaTeLo Convertor

Kako se svaki od alata može smatrati kao zasebna funkcionalna celina, u daljem nastavku pojedinično je opisan svaki alat u redosledu u kom su navedeni.

## 2.1 MaTeLo Usage Model Editor

MaTeLo Usage Model Editor je vizuelni (grafički) alat, koji je prilazan na Slici 2.2 u kome se stvara (fomira) model korišćenja sistema koji se želi testirati.



Slika 2.2 – MaTeLo Usage Model Editor

Prikazan je izgled MaTeLo Usage Model Editora sa proizvoljno definisanim modelom korišćenja koji je priložen uz MaTeLo programski paket. Zadati model korišćenja zadatog programa koji je potrebno statistički testirati upotrebom MaTeLo alata, što je jedna od tema rada, opisan je posebno u nastavku poglavlja.

MaTeLo Usage Model Editor, kao vizuelno razvojno okruženje za pravljenje modela korišćenja sistema, obezbeđuje osnovni skup funkcionalnosti, kao i druga vizuelna razvojna okruženja koja su uglavnom tehničke prirode (na primer upravljanje i rad sa datotekama). One nisu razmatrane u nastavku. Detaljnije informacije o tome su dostupne u posebnom dokumentu koji je priložen uz MaTeLo alat. Kako je model korišćenja zadat, u nastavku rada su opisane najznačajnije funkcije i mogućnosti alata koje se odnose pravljenje i podešavanje parametara modela korišćenja sistema.

Model korišćenja je sačinjen od stanja (eng. states) u kojima se sistem može nalaziti, prelaza (eng. transitions) koji predstavljaju ulaz (stimuli) sistema i upravljanja (kontrole) koja se primenjuje za zadati ulaz. Model korišćenja sistema predstavlja osnovu za MaTeLo Testor, alat koji omogućuje automatsko stvaranje zbirki apstraktnih test slučajeva.

### 2.1.1 Stanja sistema

Svako stanje sistema predstavlja stabilno stanje u kome se sistem može nalaziti pre nadolazećeg ulaza [3]. U svakom modelu upotrebe postoje dva obavezna stanja, a to su početno (eng. Invoke) i završno (eng. Terminate) stanje koja su prikazana na Slici 2.2. MaTeLo alat razlikuje dva tipa stanja sistema.

- Normalno (osnovno) stanje (eng. Normal state)
- Makro stanje (eng. Macro state)

Osnovna razlika između makro stanja i osnovnog stanja je ta što makro stanje može objediniti (grupisati) više osnovnih ili makro stanja, dok osnovno stanje predstavlja jedinstveno stanje sistema koje ne sadrži ni jedno drugo stanje. Prilikom izrade modela korišćenja makro stanja olakšavaju logičko i funkcionalno objedinjavanje stanja sistema.

### 2.1.2 Prelazi iz trenutnog u naredno stanje sistema

Prelazi predstavljaju putanje kojima se iz jednog stanja može preći u neko drugo stanje ukoliko se na ulazu testiranog sistema desio očekivani događaj za dati prelaz. Prelazi su razdvojeni jedni od drugih upotrebom stanja [3]. Svakom prelazu pored definisag ulaza

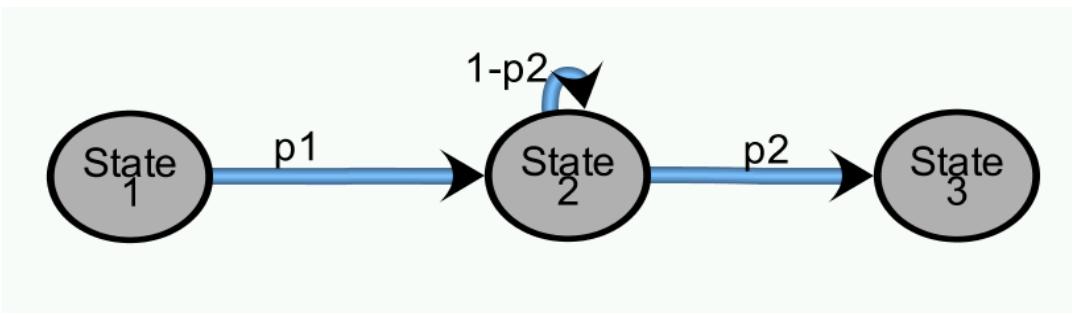
obavezno se pridružuje i verovatnoća prelaza. Verovatnoće prelaza trebaju biti što približnije stvarnoj upotrebi sistema kako bi MaTeLo alat mogao stvoriti što pouzdaniju zbirku testnih slučajeva za dati sistem. Vrednosti verovatnoća mogu biti utvđene prema unapred određenim raspodelama koje su ponuđene od strane alata:

- Skoro uvek (eng. Almost always)
- Veoma često (eng. Very often)
- Često (eng. Often)
- Normalna (eng. Normal)
- Ponekad (eng. Sometimes)
- Retko (eng. Rarely)
- Nikad (eng. Never)

*Normalna* raspodela se koristi kada se želi da MaTeLo alat dodeli jednakе verovatnoće prelaza iz trenutnog stanja ka svim ostalim stanjima, odnosno, kad se želi da svi prelasci budu ravnopravno zastupljeni prilikom stvaranja testnih slučajeva. Ukoliko se želi postići da neki prelaz bude skoro uvek izvršen, tada se koristi *Skoro uvek* raspodela. *Često* raspodela daje prelazu dva puta, dok *Veoma često* raspodela daje deset puta veću verovatnoću pojavljivanja prelaza u odnosu na verovatnoću koja daje *Normalna* raspodela. Određeni prelazi nikad neće biti izvršeni ukoliko se njima dodeli *Nikad* raspodela. *Ponekad* raspodela prelazu daje dva dok *Retko* raspodela daje deset puta manju verovatnoću pojavljivanja nego verovatnoća koja se dobije kada se koristi *Normalna* raspodela. Pri ručnom podešавају vrednosti verovatnoća prelaza mora se obratiti pažnja da zbir verovatnoća iz jednog stanja ka svim ostalim stanjima bude jednak jedinici.

$$\sum_{i=1}^n p_i = 1 \quad (1)$$

Na Slici 2.3 je prikazan primer koji treba da pokaže način određivanja verovatnoća prelaza. U primeru postoje tri stanja: *State 1*, *State 2* i *State 3*. Prepostavimo da se sistem nalazi u stanju *State 1*. Iz *State 1* stanja sistem nema ni jedan drugi mogući prelaz, osim u stanje *State 2*. Prema tome, vrednost verovatnoće  $p_1$ , da sistem pređe iz stanja *State 1* u stanje *State 2* je jednak jedinici, što odgovara obrascu (1). Iz stanja *State 2* sistem može da ostane u stanju *State 1* ili da pređe u stanje *State 3*. Ako bi se za određivanje verovatnoća oba prelaza koristila *Normalna* raspodela koja daje iste verovatnoće svakom prelazu, verovatnoća prelaza  $p_2$ , da sistem iz stanja *State 2* pređe u stanje *State 3*, bi bila 0.5. Na osnovu obrasca (1) se može zaključiti da je verovatnoća da sistem ostane u stanju *State 2* takođe jednak 0.5.



Slika 2.3 – Verovatnoće prelaza u modelu korišćenja

Ukoliko postoje više kategorija korisnika istog sistema na istom modelu korišćenja moguće je definisati više profila upotrebe. Model korišćenja ostaje isti, ali se verovatnoće prelaza iz određenih stanja podešavaju tako da budu u skladu sa stvarnom upotrebom od strane korisnika tog profila.

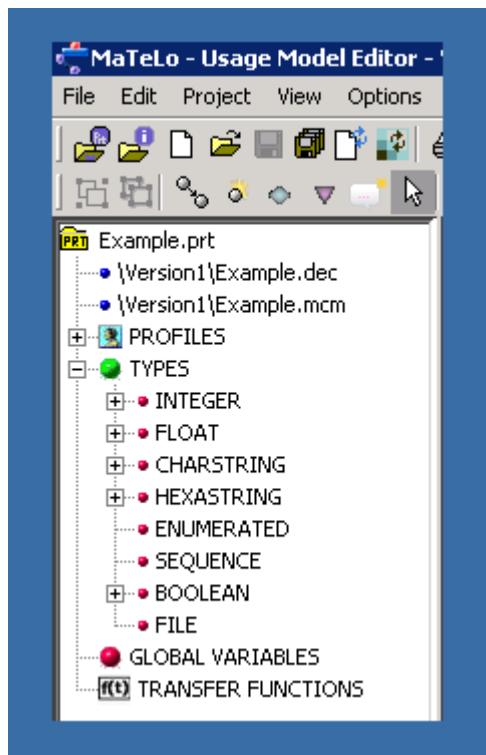
Svakom prelazu u modelu korišćenja se može pridružiti odeđeni zahtev. Zahtev predstavlja neki događaj ili akciju koju sistem prilikom konkretnog prelaza mora da ispunи kako bi se smatrao ispravnim. Ovo predstavlja ključnu vezu kojom se omogućuje provera rada realizovanog sistema i apstraktnog modela korišćenja sistema. Zahtevi se uglavnom postavljaju na osnovu opisa rada sistema (specifikacije).

### 2.1.3 Ulazi u sistem

Ulez u sistem se definiše kao vrednost nekog od osam osnovnih tipova podataka koji su podržani. Njihova vrednost se smatra kao pojava ulaza u sistem, što za posledicu ima prelazak sistema iz tekućeg stanja u neko naredno stanje. Ulazi predstavljaju spoljašnje događaje (akcije) prouzrokovane od strane korisnika sistema što može biti mašina ili čovek.

Podržani tipovi podataka od strane MaTeLo Usage Model Editor su:

- Celeobrjna vrednost (eng. Integer)
- Decimalna vrednost sa pokretnim zarezom (eng. Float)
- Nizovi znakova (eng. Charstring)
- Nizovi heksadecimalnih cifara (eng.. Hexastring)
- Nabranja (eng.. Enumerated)
- Nizovi (eng. Sequence)
- Logičke promenjive Bulovog tipa (eng. Boolean)
- Datoteke (eng. File)



Slika 2.4 – Ugrađeni tipovi podataka

Pomenuti tipovi su prikazani na Slici 2.4. Oni predstavljaju skup osnovnih tipova podataka većine programskih jezika koji su danas u širokoj upotrebi kao što su programski jecizi *C*, *C++* ili *JAVA*. Za jednostavne tipove podataka MaTeLo nudi mogućnost stvaranja novog tipa podatka kako bi podaci u modelu korišćenja sistema odgovarali stvarnim tipovima podataka koji se koristi pri realizaciji sistema. To se radi na način kao što se u *C* jeziku stvara novi tip podatka upotrebom *typedef* naredbe.

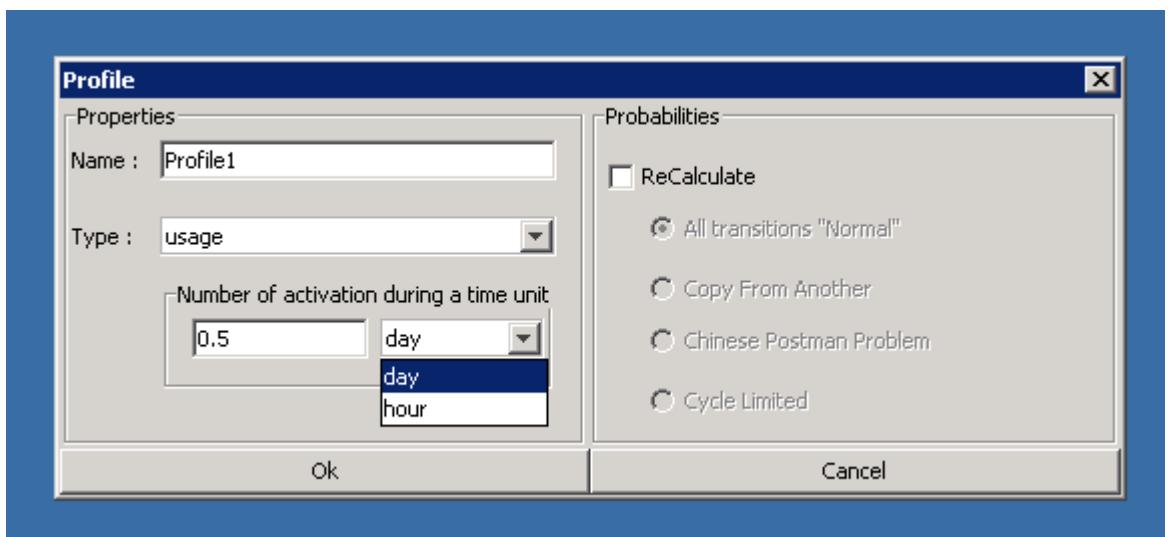
#### 2.1.4 Broj aktivacija sistema

Pored mogućnosti ručnog podešavanja verovatnoća svakog prelaza izuzetno je važna mogućnost podešavanja broja aktivacija sistema. Pored modela korišćenja i podešenih verovatnoća prelaza ova informacija predstavlja najvažniji podatak koji se koristiti u MaTeLo Testor alatu.

Broj aktivacija sistema predstavlja podatak koji definiše frekvenciju (dinamiku) korišćenja konkretnog sistema u određenoj jedinici vremena (eng. time unit). Za napravljeni model korišćenja u MaTeLo Usage Model Editoru broj aktivacija ne predstavlja broj aktivacija celokupnog sistema, nego se odnosi na broj aktivacija svakog pojedinačnog prelaza. Na ovaj način se pri analizi sistema određuje pouzdanost i MTTF svakog dela sistema pojedničano na osnovu kojih se određuje ekvivalentna pouzdanost i MTTF celog sistema.

Kao što se može videti na Slici 2.4 MaTeLo alat omogućava odabir jedne od dve raspoložive vremenske jedinice: Dan (eng. Day) i Sat (eng. Hour). Vremenske jedinice mogu se smatrati velikim što može biti ograničenje. Međutim, kao broj aktivacija je moguć unos razlomljenih (decimalnih) vrednosti, čime se može postići bolja rezolucija.

Mogućnost postojanja više tipova modela korišćenja sistema, promenom verovatnoća prelaza i broja aktivacija sistema omogućuje se efikasnije, temeljnije i lakše tesiranje sistema.



Slika 2.5 – Podešavanje broja aktivacija sistema

### 2.1.5 Informacije i scenariji korišćenja sistema

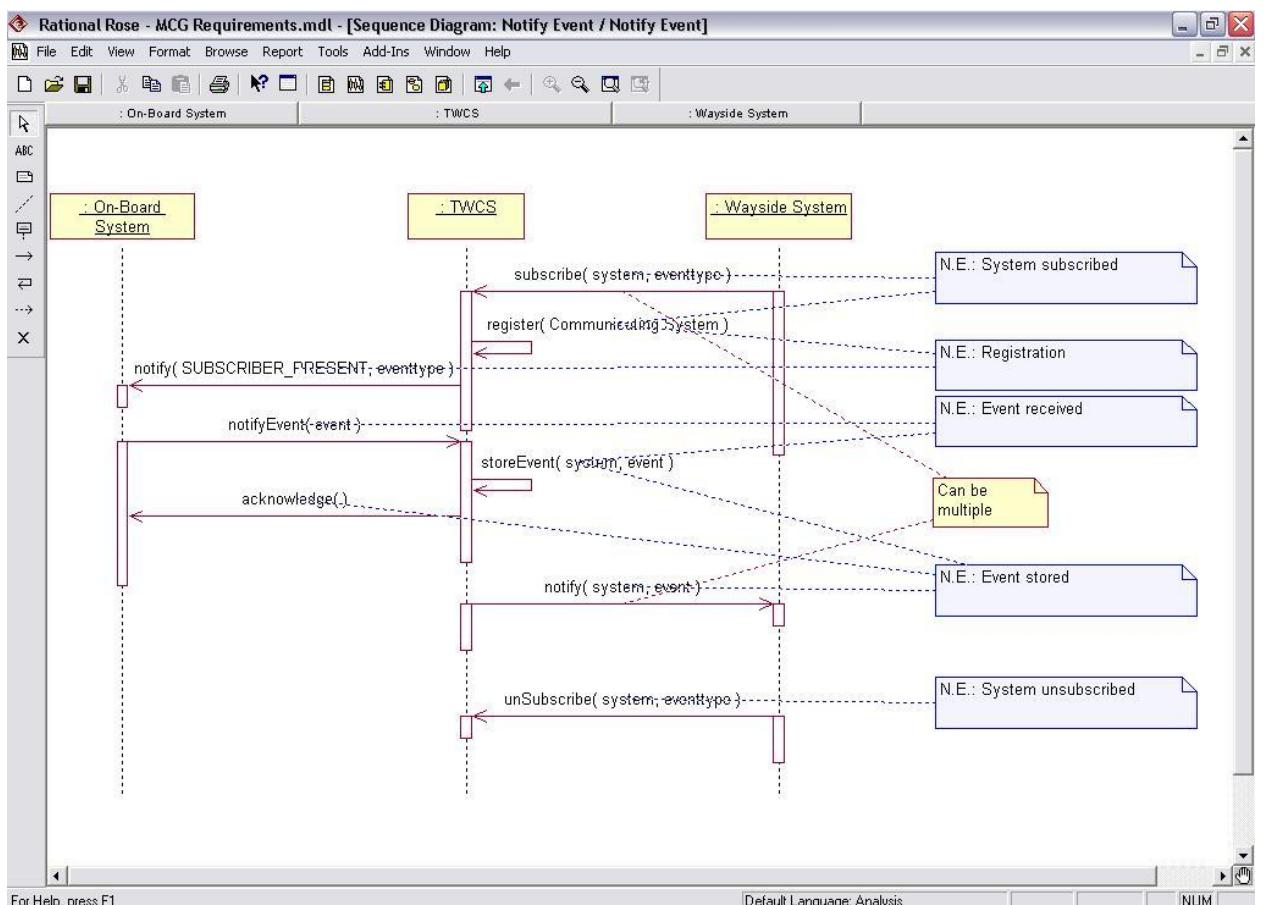
Kombinovanjem stanja i prelaza i menjanjem različitih vrednosti ulaza formira se model korišćenja sistema. Cilj je da se modelom korišćenja obuhvate najverovatniji scenariji korišćenja sistema kako bi sistem bio testiran na način na koji se koristi. Na Slici 1.1, koja se nalazi na početku ovog poglavlja, pokazano je da postoje dva osnovna izvora informacija koja se koriste u toku izrade modela korišćenja:

- Informacije o upotrebi sistema na osnovu korišćenja
- Formalni scenariji korišćenja (*MSC, UML-SD*)

Informacije o upotrebi sistema na osnovu korišćenja predstavljaju informacije dobijene eksperimentalnim metodima korišćenja sistema ili pouzdanim statističkim metodima kao što su Poasonova, Gausova ili neka druga raspodela ukoliko odgovaraju stvarnoj dinamici korišćenja sistema. Dinamika korišćenja nekih sistema koji se testiraju lako se uklopaj u pomenute, dobro poznate matematičke raspodele, dok je sa druge strane za neke sisteme izuzetno teško pronaći

odgovarajuću raspodelu. U tom slučaju se eksperimentalno moraju prikupiti informacije o korišćenju sistema u cilju što boljeg određivanja verovatnoća.

Najbolje rešenje za prikupljanje informacija i određivanja verovatnoća prelaza je eksperimentalna metoda korišćenja sistema, ali nju je nekada nemoguće primeniti zato što je vremenski veoma zahtevna i zato je nepraktična za upotrebu u tom smislu. Testiranje sistema na osnovu statističkih informacija o korišćenju nema za cilj testiranje celokupnog sistema, nego samo određenih delova sistema, kao i željnih scenaraija korišćenja koji su od interesa. U tom smeru, MaTeLo Usage Model Editor omogućuje definisanje scenarija korišćenja sistema pomoću formalnih jezika za specifikaciju kao što su *MSC* (Message Sequence Chart) i *UML-SD* (Unified Modeling Language-Sequence Diagram) jezik. Ovo daje mogućnost, ukoliko se želi, da se provere najkritičniji scenariji korišćenja sistema. Na Slici 2.5 je dat primer *UML-SD* dijagrama napravljenog u programu *IBM Rational Rose*. Konkretni primer *UML-SD* scenarija korišćenja isporučuje se kao referentni uz MaTeLo programski paket.



Slika 2.6 – Primer formalne definicije scenarija korišćenja upotrebom UML jezika

Definisani *MSC* ili *UML-SD* scenariji korišćenja se neposredno učitavaju u MaTeLo Usage Model Editor. Ako su scenariji korišćenja sistema ispravno definisani, nakon učitavanja MaTeLo Usage Model Editor na osnovu njih automatski pravi model korišćenja sistema.

Nakon formiranja modela može se preći na sledeći korak u procesu statističkog testiranja željenog sistema a to je stvaranje apstraktnih test slučajeva. U ovom radu se podrazumeva da postoji formiran unapred zadati model korišćenja zadatog programa koji se želi statistički testirati i on je posebno obrađen, tako da se u daljem nastavku izostavljaju detalji vezani za MaTeLo Usage Model Editor. Više informacija o MaTeLo Usage Model Editoru i načinu njegovog korišćenja dostupni su u dokumentu *MaTeLo User Manual*.

## 2.2 MaTeLo Testor

MaTeLo Testor je drugi deo MaTeLo alata koji ima dvostruku namenu. Koristi se za automatsko stvaranje apstraktnih test slučajeva i za analizu sistema, odnosno procenu karakteristik testiranog sistema na osnovu dobijenih rezultata testiranja.

### 2.2.1 Stvaranje apstraktnih test slučajeva

MaTeLo Testor koristi model korišćenja (napravljen u predhodno opisanom MaTeLo Usage Model Editoru) za stvaranje apstraktnih zbirk test slučaja. Prilikom stvaranja test slučaja najbitnije su informacije o verovatnoći prelaza iz tekućeg u naredno stanje sistema i verovatnoće pojavljivanja definisanih ulaza u sistem. Pomenute verovatnoće se koriste kako bi se dobio, sa jedne strane posmatrano, slučajan, ali opet, sa druge strane, najverovatniji scenario korišćenja posmatranog sistema.

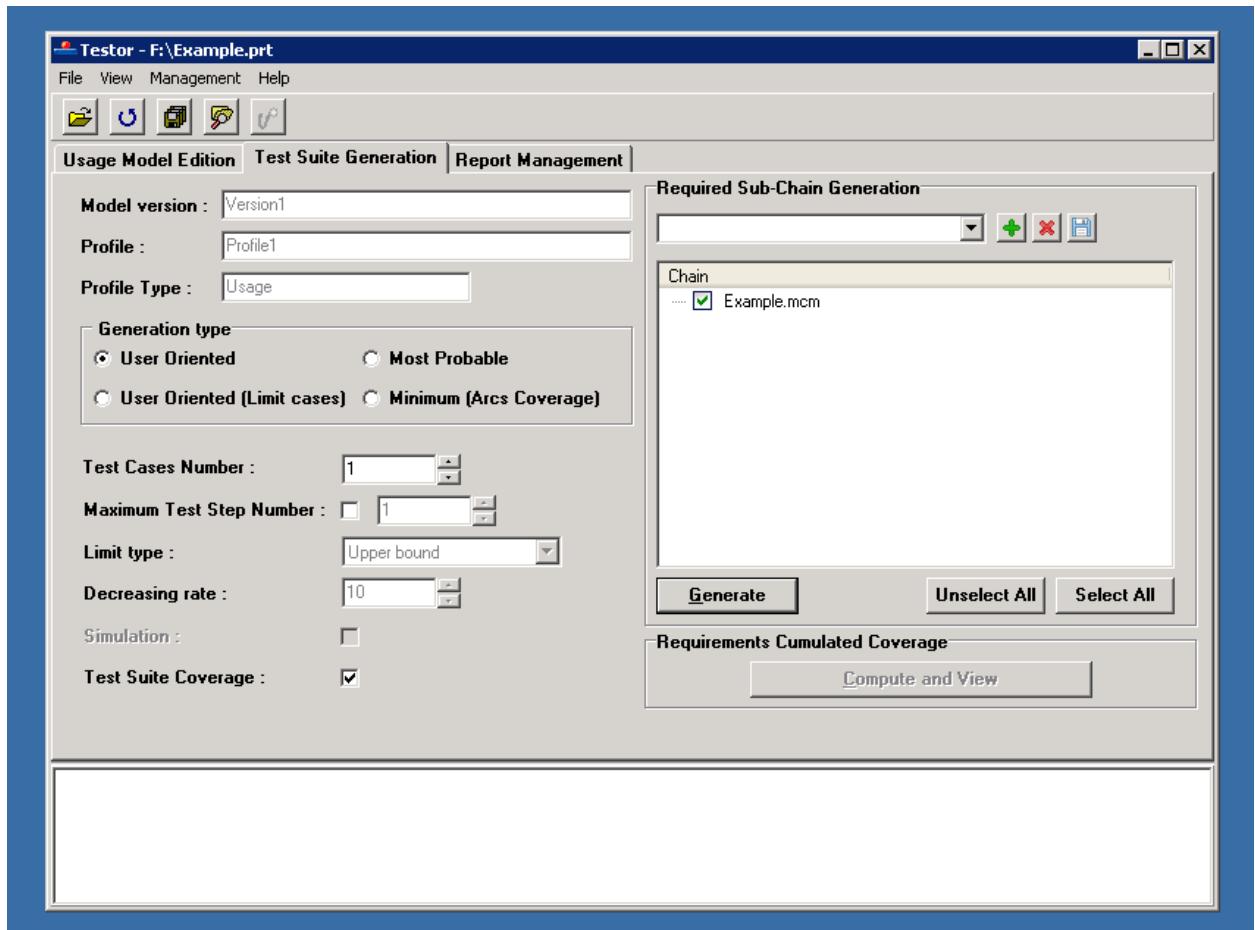
Prilikom testiranja mogu se izabrati različite strategije na osnovu kojih se stvaraju apstrakti test slučajevi. Ukupno postoje četiri strategije:

- Korisnički orijentisana (eng. User Oriented)
- Koriscnički orijentisana sa ograničenjima (eng. User Oriented (Limit cases))
- Strategija najverovatnije upotrebe (eng. Most Probable)
- Strategija minimalnog pokrivanja modela (eng. Minimum (Arcs Coverage))

Pored različitih startegija testiranja postoji mogućnost odabira posebog dela sistema koji se želi testirati. Delovi sistema koji se mogu posmatrati kao odvojene funkcionalne celine se nalaze u posebnim *mcm* (Markov Chain Model) datotekama stovrenim u MaTeLo Usage Model

Editoru. Odabir posebnog dela sistema koji se testira predstavlja korisnu mogućnost u tom smislu što se svaki deo sistema ne mora na isti način testirati, odnosno prilikom testiranja se ne mora koristiti ista strategija stvaranja test slučaja. Ukoliko se testira složeni sistem promena strategije testiranja može da donese značajnu uštedu na vremenu.

Na Slici 2.7 sa leve strane se nalaze ponuđene strategije kao i dodatni parametri koji se koriste u toku automatskog stvaranja test slučajeva dok se sa desne strane nalaze odabrani delovi sistema koji se testiraju.



Slika 2.7 – MaTeLo Testor, strategije stvaranja test slučaja

Korisnički orijentisana strategija prilikom stvaranja test slučajeva kao najbitniji parametar uzima verovatnoće prelaza iz trenutnog u naredno stanje, kao i verovatnoće pojavljivanja određenih ulaza u sistem. Najverovatniji prelazi i ulazi u sistem najviše su zastupljeni u dobijenim test slučajevima.

Korisnički orijentisana strategija sa ograničenjima predstavlja poseban slučaj predhodne strategije. Ona se razlikuje u načinu odabira ulaza u sistema dok se prilikom odabira prelaza poštuju definisane verovatnoće. Naime, za svaki skup ulaza koji su definisani za određeni prelaz

definiše se verovatnoća njihovog pojavljivanja. Za razliku od predhodnog slučaja, gde su se za ulaz u sistem birali ulazi sa najvećim verovatnoćama, ovde se biraju isključivo ulazi sa najvećom, odnosno, sa najmanjom verovatnoćom. Odabir ulaza zavisi od podešavanja *Limit type* parametra u MaTeLo Testoru. Moguće je podesiti da oba ulaza budu birana (eng. Random bound), da bude biran samo ulaz sa najvećom verovatnoćom pojavljivanja (eng. Upper bound) ili da bude biran samo ulaz sa najmanjom verovatnoćom pojavljivanja (eng. Lower bound). Ovom strategijom se omogućuje provera rada sistema u graničnim slučajevima.

Strategija najverovatnije upotrebe takođe predstavlja poseban slučaj Korisnički orijentisane strategije. Prilikom odabira prelaza iz tekućeg u naredno stanje MaTeLo Testor bira prelaz sa najvećom verovatnoćom kao u Korisniči orijinetisanoj strategiji. Pre prelaska u naredno stanje bira se najverovatniji ulaz, potom MaTeLo Testror preračunava verovatnoće svih mogućih prelaza iz tekućeg stanja umanjujući odabrani prelaz za definisani procenat. Ovaj procenat je uključen kao parametar. On je podešen od strane korisnika korz MMI spregu (eng. Man Machine Interface) i poznat je kao stopa umanjenja (eng. decreasing rate).

Strategija mimimalnog pokrivanja, za razliku od predhodne tri strategije, ne uzima u obzir verovatnoće prelaza. Cilj ove strategije jeste da se sa što manje stvorenih test slučaja obuhvati funkcionalnost celog modela korišćenja. Pored smanjenje broja test slučaja MaTeLo Testor će pokušati i da maksimalno smanji broj koraka izvršenja test slučaja (eng. test step number). Bitno je napomenuti da se ova strategija ne može koristiti ukoliko se u modelu koriste uslovni (eng. conditional) ili asihroni (eng. asynchronous) događaji.

Nakon odabira strategije testiranja, podešavanja podrebnih parametara koji zavise od odabira strategije i definisanja delova sistema čija se funkcionalnost želi proveriti potrebno je uneti broj test slučaja koji će biti stvoreni. Nakon unosa broja test slučaja MaTeLo Testor ima sve podatke koji su potrebni za automatsko stvaranje zbirke apstraktnih test slučaja.

Na Slici 2.8 prikazan je izgled stvorene apstraktne zbirke koju čine deset test slučajeva. MaTeLo Testor za svaki test slučaj nudi sledeće informacije koji se odnose na pokrivenost modela korišćenja sistema:

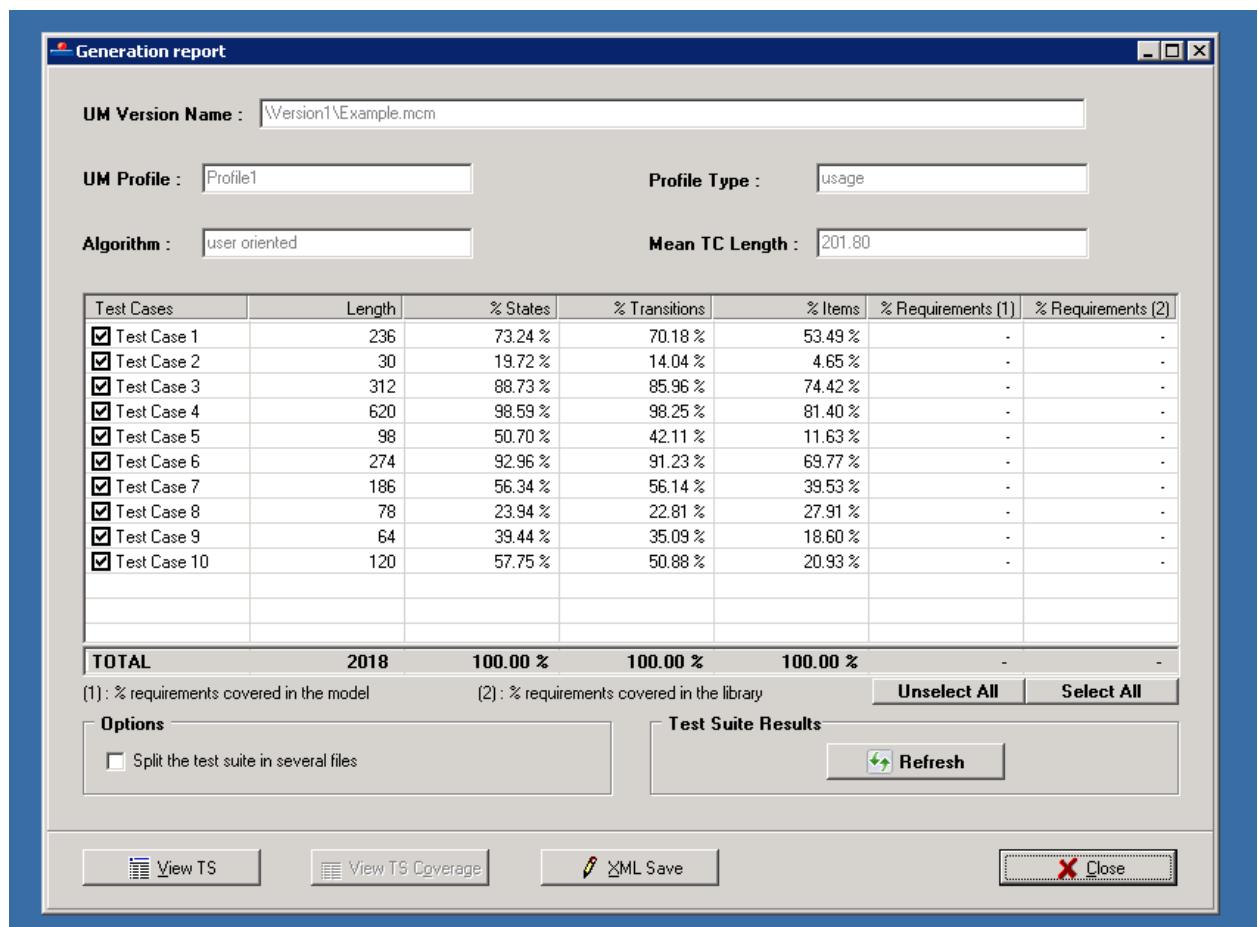
- Ime test slučaja (eng. Test Case)
- Dužinu test slučaja (eng. Length)
- Procentualnu pokrivenost stanja (eng. States [%])
- Procentualnu pokrivenost prelaza (eng. Transitions [%])
- Procentualno pokrivenost ulaza (eng. Items)
- Procentualno pokrivenost zahteva (eng. Requirements (1, 2) [%])

Dužina test slučaja predstavlja ukupan broj koraka, odnosno prelaza u modela korišćenja koji se izvršavaju prilikom pokretanja test slučaja. Svakim korakom je predstavljena neka od funkcija sistema. Broj koraka test slučaja, zavisi od veličine i složenosti modela korišćenja, kao i od odabrane strategije testiranja.

Procentualna pokrivenost stanja, prelaza i ulaza sistema daje podatak koliko se od ukupnog broja stanja prelaza i ulaza, koji čine model korišćenja, pojavilo u konačnom test slučaju. Idealno bi bilo ukoliko bi svaki test distigao stoprocentnu pokrivenost.

Procentualna pokrivenost zahteva se odnosi na ukupan broj postavljenih zahteva koji trebaju biti provereni prilikom testiranja sistema.

Takođe su priloženi podaci koje se odnose na ukupnu zbirku testova. Na Slici 2.8 pokrivenost stanja, prelaza i ulaza u sistem je stoprocentna dok je dužina zbirke, ukupan broj test koraka svih testnih slučajeva, dve hiljade osamnaest. Nisu postavljeni nikakvi zahtevi koji se odnose na model korišćenja sistema pa zato pokrivenost zahteva nije dostupna.



Slika 2.8 – Stvaranje zbirke apstraktnih test slučajeva

Ako je dobijena zbirka apstraktnih test slučajeva zadovoljavajuća, čuva se za dalje upotrebu. Ukoliko nije, proces stvaranja test slučajeva se može ponoviti. Sve dobijene zbirke ili pojedinačno dobijeni test slučajevi se čuvaju u *xml* datoteci. Pomenuta *xml* datoteka ima dvojaku ulogu. U prvom koraku, koraku stvaranja apstarktnih zbirki test slučajeva, predstavlja izlaz iz MaTeLo Testora. Tada se u njoj nalaze sve informacije potrebe za prevođenje (konverziju) apstraktnih test slučaja u izvršive test slučaje koji su prilagođeni za pokretanje na nekoj namenskoj platformi za automatsko izvršavanje kao što su: TestStand, SeleniumHQ, JUnit ili HP Quick Professional. Izlazna *xml* datoteka sadrži sledeće:

- Informacije o prelazima
- Informacije o stanjima sistema
- Ulazi u sistem
- Očekivane vrednosti nakon prelaza
- Informacije o sistemu koji se testira

Sve pomenute informacije se koriste od strane MaTeLo Convertor alata koji omogućuje prevođenje apstraktnih test slučajeva u izvršive test slučajeve. Pošto se dobiju izvršivi test slučajevi može se početi sa testiranjem sistema i priklupljanjem rezultata. Nakon završetka dobijeni rezultati se moraju proslediti MaTeLo Testoru da bi se dobila informacija kao što je pouzdanost sistema ili srednje vreme do otkaza. Dobijeni rezultati testiranja se upisuju (integrišu) u *xml* datoteku sa već postojećim informacijama. Za svaki pokrenuti test slučaj se dodaju sledeći podaci:

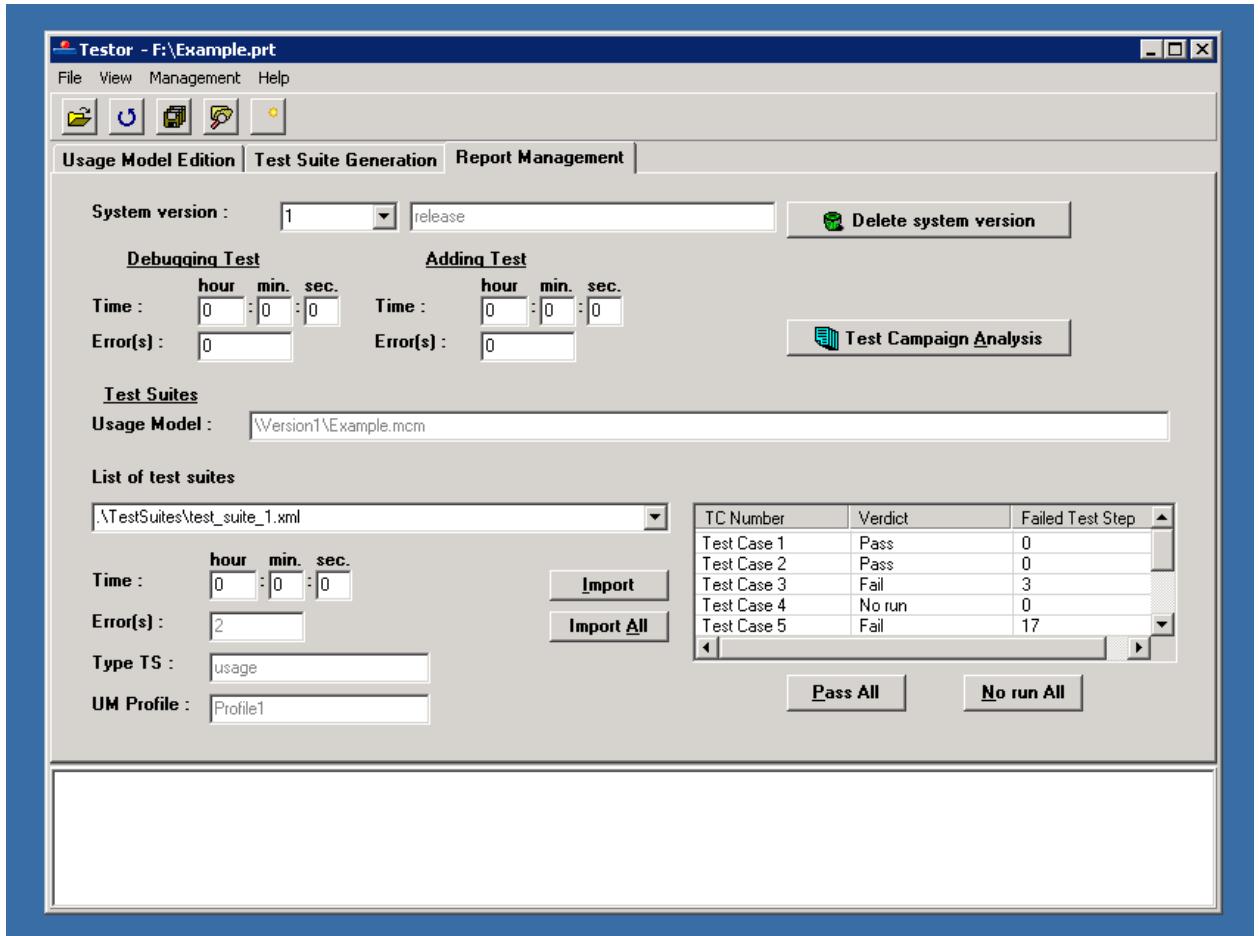
- Rezultat testa (eng. verdict)
- Vremena izvrešnja (eng. execution time)
- Korak na kome se desila greška (eng. step error)

Proces unošenja dobijenih rezultata može biti ručan ili automatski. Ručno unošenje se vrši pomoću internet pretraživača, što je u slučaju velikog broja test slučaja neefikasno i podložno greškama. Automatsko unošenje se ostvaruje korišćenjem sintaksnog analizatora koji je detaljno opisan kasnije. Više o tome u narednom poglavljju.

### 2.2.2 Analiza sistema na osnovu dobijenih rezultata

Pod analizom sistema se smatra određivanje karakteristika sistema kao što su pouzdanost, srednje vreme do otkaza ili nekih drugih statistički dobijenih podataka koji su zasnovani na analizi dobijenih rezultata test slučaja kao što je verovatnoća otkaza (eng. failure intensity) ili

lokalizacija grešaka (eng. localisation of errors). Izgled MaTeLo Testora u kome se unose i analiziraju dobijeni rezultati prikazan je na Slici 2.9.



Slika 2.9 – MaTeLo Testor, unos i analiza dobijenih rezultata

Da bi MaTeLo Testor mogao dati procenu karakteristika sistema neophodno je uneti rezultate prikupljene tokom testiranja sistema. Pre unosa dobijenih rezultata u MaTeLo Testoru za svaku zbirku test slučaja potrebno je registrovati novu verziju sistema. Pošto se registruje nova verzija sistema, može se otpočeti sa unosom dobijenih rezultata. MaTeLo Testor pruža dva načina unosa rezulta: ručni i automatski.

Ručni (manuelni) unos rezultata je omogućen u samom MaTeLo Testoru. Ručni unos rezultata koji je pomenut u predhodnoj sekciji odnosi se na unos rezultata u *xml* datoteku koja se koristi u MaTeLo Testoru prilikom automatskog unosa koji je opisan u nastavku. Pre unosa rezultata potrebno je odabrat odgovarajuću zbirku test slučaja kojom je sistem testiran, tj. zbirku test slučaja na osnovu koje su dobijeni rezultati. U donjem desnom uglu na Slici 2.9 nalazi se spisak svih test slučaja koji čine odabranu zbirku. Nakon toga se unose sledeći podaci:

- Rezultat test slučaja (eng. Verdict)

- 
- Korak na kome se desila greška (eng. Failed Test Step)
  - Ukupno vremena izvrešnja zbirke test slučaja (eng. Time)

Rezultatom test slučaja se označava da li je konkretni testni slučaj uspešno izvršen (eng. Pass), da li se desila greška u toku njegovog izvršenja (eng. Fail) ili test slučaj nije ni pokrenut (eng. No Run). U zavisnosti od rezultata unosi se odgovarajući broj koji predstavlja korak izvršenja na kome se desila greška. Ukupno vreme izvršenja svih test slučaja, koji čine zbirku, ručno se unosi u posebnom (Time) polju. Polje koje označava ukupan broj otkrivenih grešaka (eng. Error(s)) se automatski ažurira na osnovu unetih rezultata.

Drugi, automatski, način unosa rezultata se zasniva na učitavanju *xml* datoteke koja sadrži već unete rezultate testiranja. Pomenuta *xml* datoteka se može dobiti na dva načina. Prvi način je ručni unos svih rezultata test slučaja upotrebom internet pretraživača. Ovo je jedini način unosa koji je obezbeđen od strane MaTeLo alata. Ukoliko postoji veliki broj rezultata koji se moraju uneti ručni unos predstavlja dugotrajan proces podložan greškama. Drugi način je automatskim užuriranjem sadržaja već postojeće *xml* datoteke upotrebom sintaksnog analizatora koji je detaljno opisan u narednom poglavlju. Dobijena *xml* datoteka se učitava (importuje) u MaTeLo Testor. Ukoliko je *xml* datoteka ispravno popunjena stavke koje su morale biti ručno unete se automatski ažuriraju (konačni rezultat, korak na kome se desila greška i ukupno vreme izvršenja zbirke test slučaja).

Pored unosa prikupljenih rezultata izvršenih test slučaja koji su stvorenii MaTeLo skupom alata MaTeLo Testor omogućuje unos i dodatnih informacija koje se odnose na rezultate test slučajeva kojima je sistem testiran, a koji nisu stvorenii MaTeLo skupom alata. Takođe je moguće uneti informacije o vremenu koje je proteklo u otklanjanu otkrivenih grešaka u sistemu (eng. Debugging Test) i vremena koje je proteklo za unos rezultata (Adding Test). Sve ove informacije se koriste pri analizi i proceni karakteristika ispitivanog sistema.

Princip funkcionisanja dela MaTeLo Testora, i uopšteno dela MaTeLo alata, koji je zadužen za procenu karakteristika ispitivanog sistema je takav da mora postajiti nekoliko ispitivanih i analiziranih verzija sistema na osnovu čijih rezultata bi se moglo izvesti pouzdane procene karakteristika sistema. Što ima više ispitivanih verzija sistema, procenjene karakteristike su pouzdanije. Naime, MaTeLo alat može dati procenu sistema na osnovu samo jedne verzije sistema, ali relevantnost tako dobijenih podataka je diskutabilna. Prilikom svake procene nove verzije sistema, MaTeLo Testor određuje karakteristike za novu verziju sistema kao i kumulativne (eng. cumulative) karakteristike sistema kod kojih se prilikom procene koriste i dobijeni rezultati predhodnih verzija sistema. Nakon završene analize (test kampanje) sistema MaTeLo Testor daje detaljan uvid u stanje ispitivanog sistema. Postoji veliki broj ponuđenih,

statistički dobijenih podataka, koji su služe za praćenje razvoja sistema (pouzdanost, kumulativna pouzdanost, pokrivenost zahteva, kumulativna pokrivenost zahteva, itd.) namenjenih za rukovodeći (eng. managment) i razvojni (eng. development) tim. U ovom radu najvažniji informacije predstavljaju procena pouzdanosti, srednjeg vremena do otkaza po verziji sistema kao i kumulativne procene pomenutih karakteristika sistema.

MaTeLo određuje pouzdanost sistema na osnovu broja aktivacija sistema koji se dese u nekoj jedinici vremena. Detaljan opis i uloga broj aktivacija dat je ranije u ovom poglavlju (odeljak 2.1.3). Tačan analitički oblik za izračunavanje pouzdanosti sistema dat je obrazcem (2)

$$R(t) = (p_u)^k \quad (2)$$

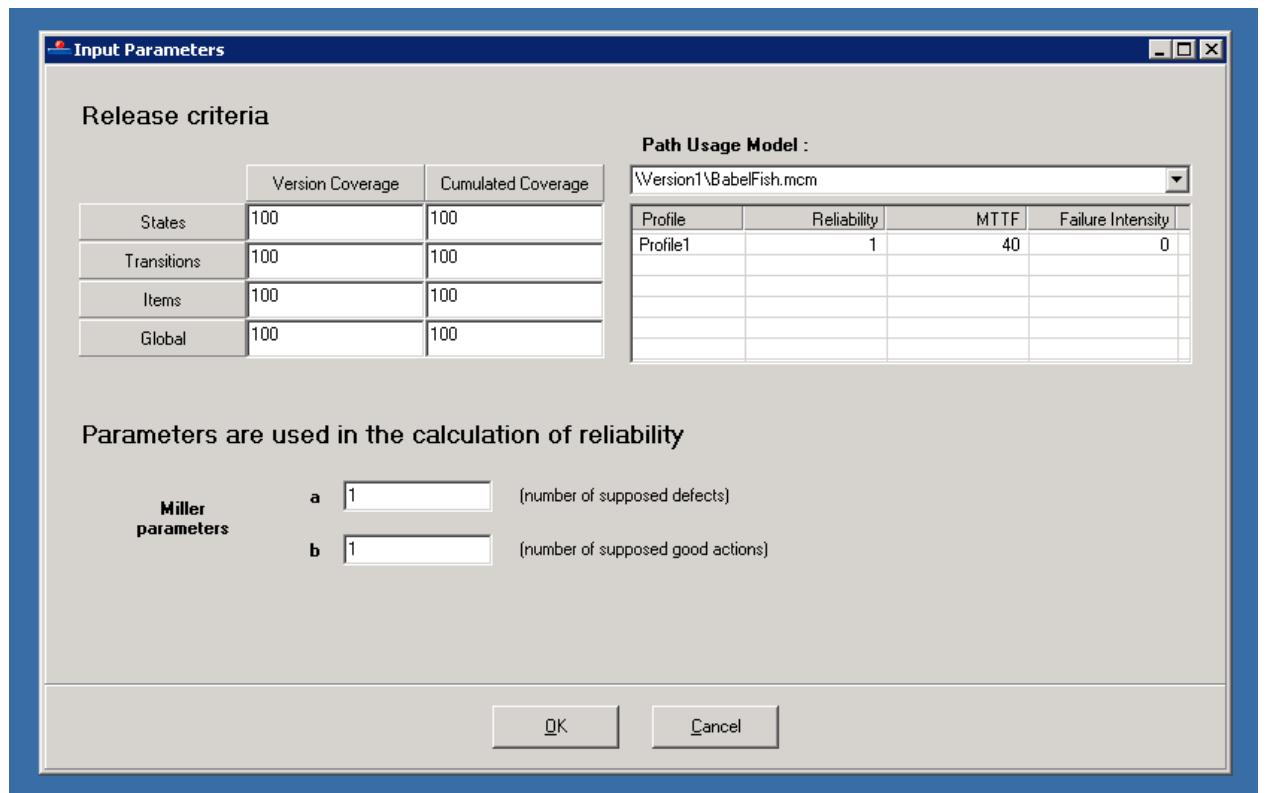
Pouzdanost sistema ( $R$ ) eksponencijalno zavisi od verovatnoće korišćenja (usage probability) sistema ( $p_u$ ) i broja aktivacija sistema ( $k$ ) u jedinici vremena ( $t$ ). Da bi pouzdanost sistema bila ispravno izračunata u modelu korišćenja se moraju podesiti broj aktivacija sistema i željena vremenska jedinica. Verovatnoća korišćenja se računa na osnovu složenih matematičkih modela i algoritama koji su ugrađeni u MaTeLo Testor.

Za razliku od pouzdanosti sistema gde postoji analitički obrazac po kome se ona izračunava, za izračunavanje srednjeg vremena do otkaza takav obrazac nije eksplicitno dat u MaTeLo dokumentaciji. MaTeLo definiše srednje vreme do otkaza kao vreme koje je potrebno da se otkrije prvi otkaz u sistemu, i izračunava se na osnovu procenjene pouzdanosti.

Mogućnost procene kumulativne pouzdanosti (eng. cumulated reliability) i kumulativne vrednosti srednjeg vremena otkaza (eng. cumulative MTTF) na osnovu modela korišćenja i rezultata dobijenih testiranjem sistema predstavlja suštinu rada MaTeLo alata. Prethodno je dat analitički izraz za izračunavanje pouzdanosti tekuće verzije sistema i pomenut je način određivanja srednje vrednosti do otkaza, takođe za tekuću verziju sistema. Za procenu kumulativne pouzdanosti sistema koristi se Milerov model (eng. The Miller Reliability Model). Milerov model je zasnovan na upotrebi Baesove statistike i omogućava korisniku modela da iskoristi predhodno znanje o sistemu koji se testira [4]. U Milerovom modelu informacije o predhodnom stanju sistema su dostupne preko Milerovih koeficijenata ( $a, b$ ) i predstavljaju podatak o verovatnoći otkaza sistema. Milerovi koeficijenti u test kampanji su dostupni preko posebnih parametara koji se unose za svaku verziju sistema posebno. Naime, nakon unetih rezultata dobijenih izvršavanjem zbirke test slučaja, unose se parametri test kampanje, odnosno Milerovi koeficijenti. Unos Milerovih koeficijenata u okviru MaTeLo Testora je prikazan na Slici 2.10. Parametar  $a$  predstavlja prvi Milerov koeficijent kojim se označava broj otkaza sistema (eng. number of supposed defect) koji su otkriveni prilikom testiranja predhodne verzije sistema. Parametar  $b$  predstavlja drugi Milerov koeficijent kojim se označava broj uspešno

izvršenih upotreba (akvicija) sistema (eng. number of supposed good actions) koje su potvrđene prilikom testiranja prethodne verzije sistema. Prilikom unosa konkretnih vrednosti parametri  $a$  i  $b$  predstavljaju broj test slučjeva kod kojih je otkrivena grešaka i broj test slučaja koji su uspešno izvršeni, respektivno. Praktično ovo znači da ukoliko zbirka test slučajeva ima deset test slučaja i ako su u toku testiranja prijavljene tri greške u sistemu, parametar  $a$  će imati vrednost tri, dok će parametar  $b$  imati vrednost sedam. Kada se vrši analiza početne verzije sistema oba parametra imaju vrednost jedan.

Kao i prilikom određivanja srednjeg vremena do otkaza po verziji sistema koja je računata na osnovu dobijene pouzdanosti, MaTeLo kumulativnu vrednost srednjeg vremena do otkaza izračunava na osnovu kumulativne pouzdanosti sistema.



Slika 2.10 – Parametri test kampanje, Milerovi koeficijenti

## 2.3 MaTeLo Convertor

MaTeLo Convertor je treći deo u nizu MaTeLo skupa alata koji omogućava sledeće funkcionalnosti:

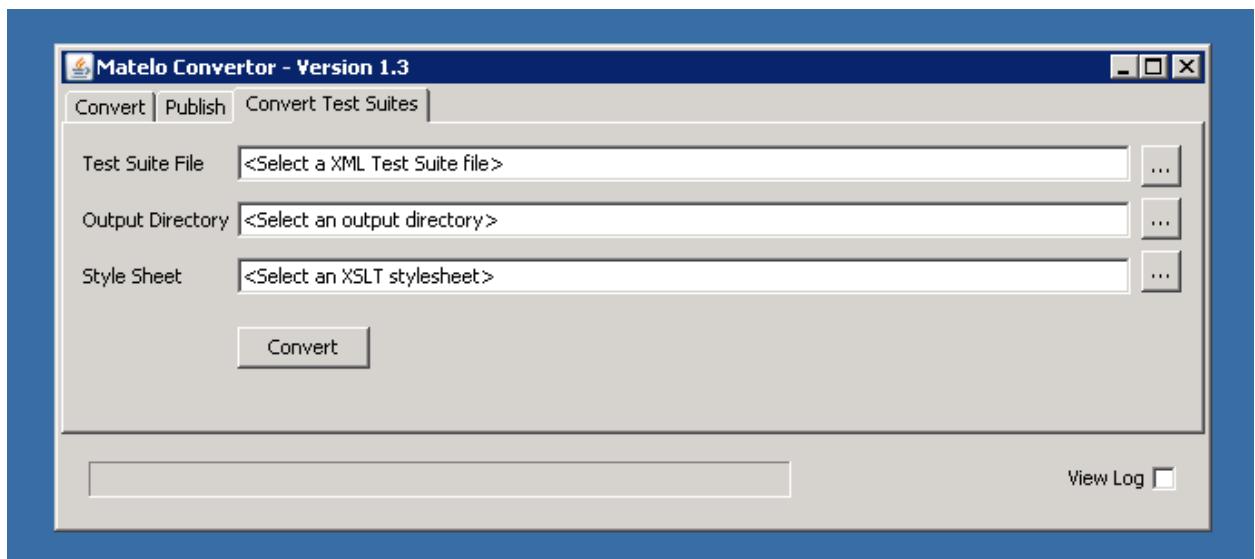
- Prevođenje starijih verzija MaTeLo projekata u novije verzije
- Izdavanje (Publish) zbirki test slučaja u PDF dokument

- Prevođenje apstraktnih test slučaja u izvršive test slučaje

Prevođenje starijih verzija MaTeLo projekata u novije verzije i izdavanje zbirki test slučaja u PDF format predstavlja više stvar tehničke mogućnosti koje je odnose na obezbeđivanje usklađenosti rada MaTeLo alata sa starijim verzijama (eng. backward compatibility) i rad sa daotekama.

### 2.3.1 Prevođenje apstraktnih test slučaja

Prevođenje apstraktnih test slučaja u izvršive test slučaje ima ključnu ulogu u omogućavanju izvršavanja apstraktnih test slučajeva koji su automatski stvoren na osnovu modela korišćenja sistema.



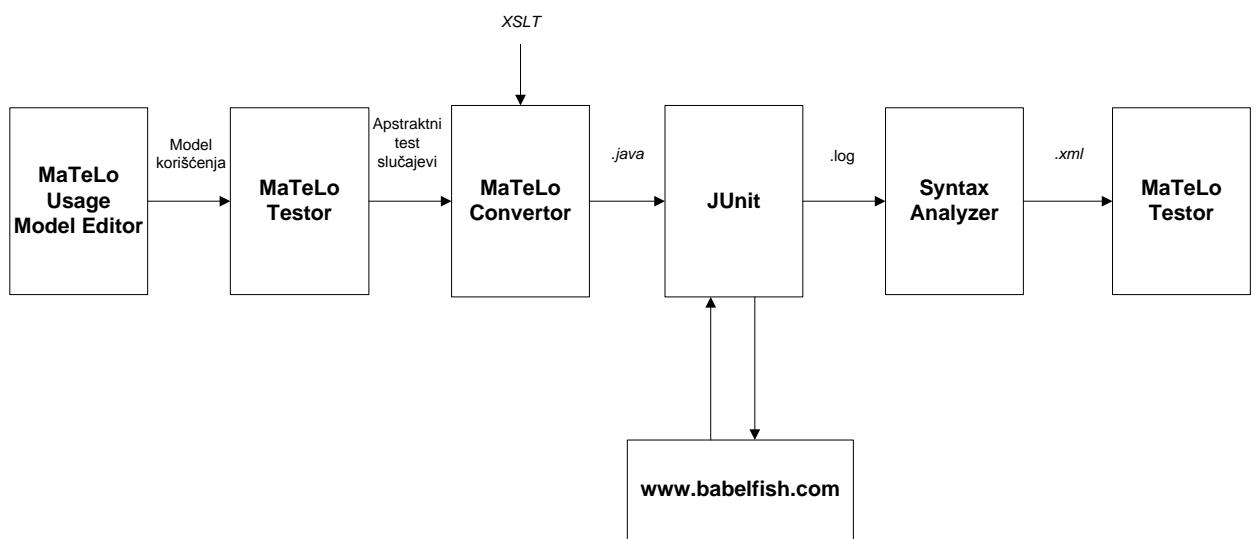
Slika 2.11 – MaTeLo Convertor, prevođenje apstraktnih u izvršive test slučajeve

Na Slici 2.11 je prikazan izgled MaTeLo Convertora. Da bi se omogućilo prevođenje, potrebno je definisati zbirku apstraktnih test slučaja koja se želi prevesti, tj. *xml* datoteku u kojoj se nalaze definisani ulazi u sistem i koraci izvršenja test slučaja, i XSLT datoteku koja predstavlja šemu po kojoj se test slučajevi prevode. Format izlaznih (prevedenih) datoteka zavisi od ciljane platforme na kojoj se žele pokretati dobijeni test slučajevi. Mogućnost prevođenja apstraktnih test slučaja predstavlja veoma korisnu mogućnost zato što MaTeLo alat obezbeđuje ugrađene funkcije automatizacije za mali broj platformi namenjenih za automatsko izvršavanje test slučaja kao što su EXAM, TestStand, TTCN-3. U koliko se želi da isti apstraktni test

slučajevi budu izvršavani na više platformi sve što je potrebo uraditi jeste definisati novu šemu prevođenja za konkretnu platformu.

## 2.4 Statističko testiranje zadatog programa

Predhodno je opisan opšti postupak upotrebe i principi rada alata koji čine MaTeLo skup alata. U ovom odeljku je na zadatom programu objašnjen je proces korišćenja svakog od MaTeLo alata. Na Slici 2.12 je prikazan tok podataka i redosled korišćenja alata u procesu statističkog testiranja zadatog programa. Ažuriranje dobijenih rezultata testiranja u *xml* datoteku kao i analiziranje dobijenih rezultata (procene karakteristika sistema) je izostavljen. Ažuriranje rezultata je detaljno opisan u poglavlju *Teorijske osnove* u kojem je posebno opisana funkcionalnost i upotreba sintaksnog analizatora u MaTeLo skupu alata. Analiza procenjenih karakteristika sistema zadatog programa je urađena u poglavlju *Rezultati*.



Slika 2.12 – Blok šema procesa statističkog testiranja zadatog programa

### 2.4.1 Model korišćenja zadatog programa

Zadati sistem koji je potrebno statistički testirati upotrebom MaTeLo alata je program realizaovan u obliku internet stranice koji služi za prevođenje teksta, Babel Fish ([babelfish.yahoo.com](http://babelfish.yahoo.com)). Pomenuti program je omogućen od strane Microsoft kompanije i prikazan je na Slici 2.13.

Da bi se program mogao koristiti potrebno je prvo pristupiti internet stranici (adresi) na kojoj se nalazi program, [babelfish.yahoo.com](http://babelfish.yahoo.com). Korišćenje programa je intuitivno. Unutar prostora koji je namenjen za unos teksta (eng. text area) unosi se reči koji se žele prevesti na

drugi jezik. Nakon unetog teksta za prevodenje bira se par jezika sa kojim se označava na kom jeziku je unet tekst i na koji jezik treba da se prevede (eng. Select from and to languages). Na kraju je potrebno dati dozvolu da se uneti tekst prevede (eng. Translate).

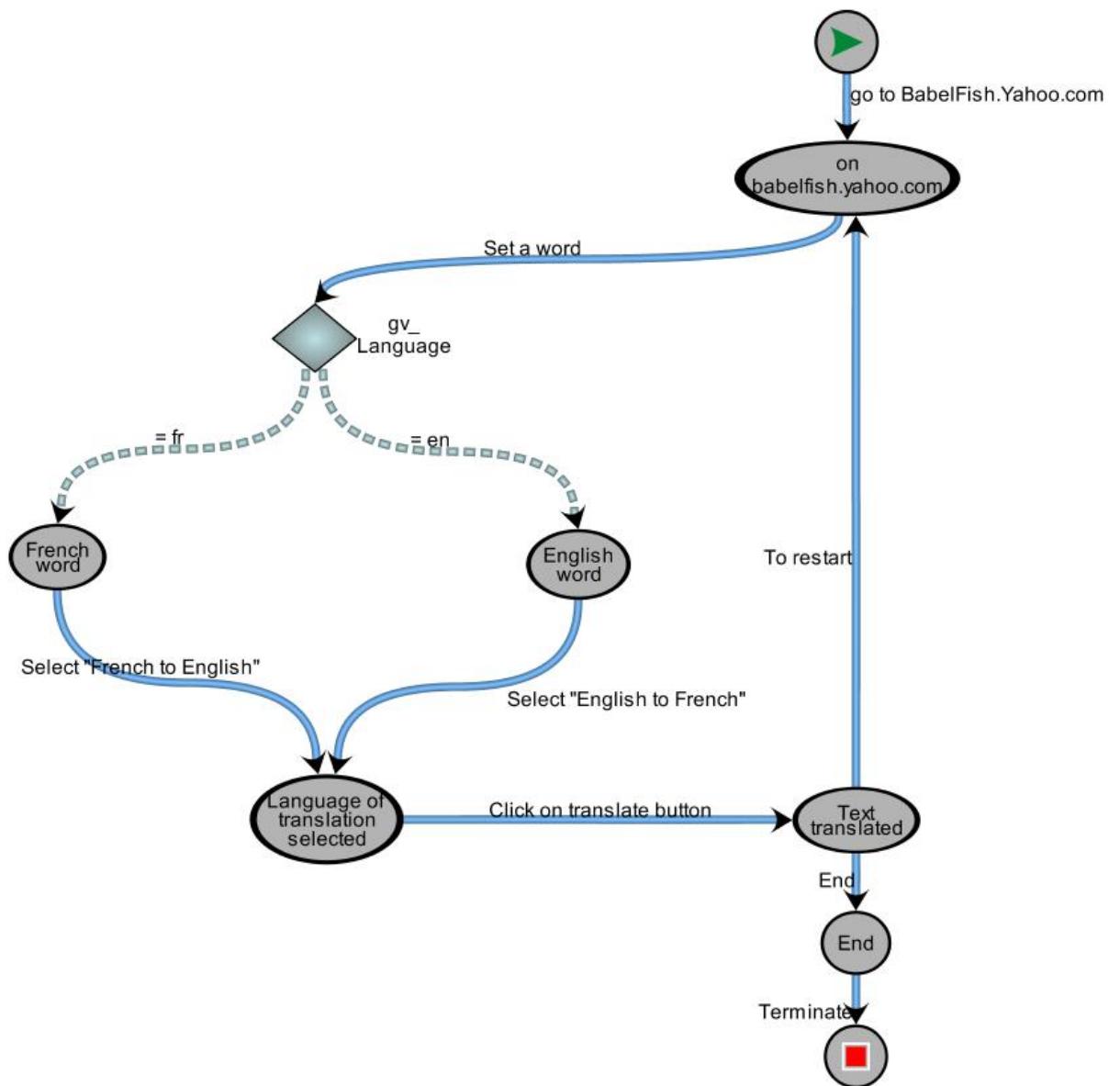


Slika 2.13 – Zadati program, Babel Fish prevodioč

Model korišćenja opisanog programa za prevodenje je dat na Slici 2.14. Ukoliko se izostave dva obavezna stanja, model korišćenja se sastoji od šest stanja. Svako stanje označava neko od stanja programa u kojem se očekuje ulaz u sistem ili u kom se obrađuju uneti podaci. Prelazi predstavljaju akcije korisnika sistema kojim se sistem prevodi iz trenutnog u naredno stanje.

Početak korišćenja sistema predstavlja odlazak na adresu na kojoj se program nalazi što je označeno prvim prelazom, *go to BabelFish.Yahoo.com*. Ukoliko je adresa ispravna i ako je program dostupan prelazi se u stanje *on BabelFish.Yahoo.com* u kom se unosi tekst koji se želi prevesti. Unos željenog teksta u modelu korišćenja je prikazan kao prelaz *Set a word*. Program se ne može testirati sa svim mogućim rečima. Zato je u modelu korišćenja napravljena gupa (baza) reči koje su korišćene u toku testiranja. Spisak korišćenih reči je dat u Tabeli 2.1. Spisak čine Engleske i Francuske reči, a ujedno su to i jezici koji su korišteni u toku prevodenja. Nakon odabira reči i jezika sa kog se prevodi tekst (prelaza *Set a word*) u modelu korišćenja se pomoću uslovnog grananja bira sledeća akcija korisnika. Uslov od koga zavisi odabir grane izvršavanje je jezik sa kog se tekst prevodi. U modelu se koriste globalne promenjive za smeštanje odabranog teksta za prevodenje i jezika teksta kako bi one bile na raspolaganju ostalim prelazima sistema. Pomoću njih se, u zavisnosti od jezika sa kog se prevodi, bira odgovarajući tekst koji treba da se dobije prevodenjem. Ovo je ključni mehanizam pomoću kog se u apstraktnom modelu korišćenja sistema omogućuje potvrda rada sistema. Model korišćenja treba da bude tako osmišljen da je u

modelu korišćenja moguće na osnovu ulaza sistema, obrade i dobijenih rezultata potvrditi rad implementiranog sistema. U zavisnosti od platforme na kojoj se test slučajevi izvršavaju način provere dobijenih vrednosti se razlikuje.



Slika 2.14 – Model korišćenja zadatog programa

Kada se odredi jezik sa kog se reč prevodi prelazi se u stanje *French word* ili *English word* koje označava konkretni jezik. U tom stanju je potrebno definisati jezik na koji se uneti tekst prevodi. Babel Fish prevodioč očekuje par reči kojima se označava jezik unetog teksta i jezik na koji se on prevodi. Odabir para jezika je označen prelazima *Select “French to English”* i *Select “English to French”*. Nakon odabira jezika prevođenja sistem prelazi u stanje *Language of translation selected* u kome se očekuje dozvola za prevođenje. Dozvola za prevođenje je prikazana *Click on tranlate button* prelazom. Nakon unosa dozvole sistem prelazi u stanje *Text translated* u kom treba da se dobije prevedeni tekst. Ovim prelazom je u modelu korišćenja

definisana očekivana vrednost koja treba da se dobije nakon prevodenja koja služi za proveru rada programa. Iz stanja *Text translated* potoje *Restart* i *End* prelazi. Prelaz *End* predstavlja završetak upotrebe programa, dok se prelazom *Restart* sistem se vraća u stanje u kome je omogućen ponovni unose teksta za prevodenje nakon čega se proces ponavlja.

Engleske reči	Francuske reči
hello	bonjour
goodbye	au revoir
thank you	merci
how are you ?	comment allez-vous ?
welcome	bienvenue
it rains	il pleut
the sky is cloudy	le ciel est nuageux
computer	ordinateur
help me	aidez-moi
fireman	pompier

Tabela 2.1 – Korišćene reči u toku testiranja

#### 2.4.2 Stvaranje **test slučajeva**

Predhodno opisani model korišćenja Babel Fish programa predstavlja ulaz u MaTeLo Testor. MaTeLo Testor na osnovu modela korišćenja, odnosno podataka koji opisuju korišćenje sistema, stvara apstraktne test slučajeve kojima se testira Babel Fish program.

Trenutna verzija MaTeLo alata korišćenjem Korisnički orijentisane strategije testiranja može da stvori zbirku koja sadrži najviše četrsto test slučajeva. Za tesiranje zadog programa stvoreno je deset zbirki sa po četrsto test slučajeva. U Tabeli 2.2 je prikazana statistika stvorenih zbirki test slučaja. Dobijena statistika se odnosi na pokrivenost modela korišćenja na osnovu koje se može birati najpogodnija strategija testiranja. Nekad se želi postići veća pokrivenost stanja sistema, dok se nekad želi postići veća pokrivenost prelaza sistema. Sve zbirke test slučaja su dobijene upotrebom Korisnički orijentisane strategije testiranja.

<b>Ime zbirke</b>	<b>Broj test slučaja</b>	<b>Pokrivenost stanja [%]</b>	<b>Pokrivenst prelaza [%]</b>	<b>Pokrivenost ulaza [%]</b>
ts_1	400	100	100	92.85
ts_2	400	100	100	92.85
ts_3	400	100	100	92.85
ts_4	400	100	100	92.85
ts_5	400	100	100	92.85
ts_6	400	100	100	92.85
ts_7	400	100	100	92.85
ts_8	400	100	100	92.85
ts_9	400	100	100	92.85
ts_10	400	100	100	92.85

Tabela 2.2 – Zbirke test slučajeva za testiranje Babel Fish programa

Zadati program se testira sa velikim brojem test slučajeva kako bi se prikupilo što više podataka na osnovu kojih se dobija procena pouzdanosti sistema i srednjeg vremena do otkaza. Kao što je u opisu MaTeLo Testora rečeno, što se više podataka obezbedi za analizu sistema dobijene procene su relevantnije.

U Tabeli 2.2 su dati i podaci o pokrivenosti stanja, prelaza i ulaza sistema. Pokrivenost stanja i prelaza je stoprocentna za svaku zbirku test slučaja. Ukoliko se uzme u obzir i složenost sistema, koja je trivijalna, i veliki broj test slučajeva po zbirci, dobijena vrednost je očekivana. Pokrivenost ulaza u sistem je uvek ista i iznosi 92.85%. Za unos u sistem postoji ukupno dvadeset mogućih reči koji se prevode na dva jezika. Verovatnoća pojavljivanja svakog od njih je ista, što znači da se najverovatnije svaka od tih reči pojavila bar u jednom test slučaju. Razlog što se ne dobije stoprocentna pokrivenost ulaza sistema je to što se ulazi, koji se koriste kao vrednosti za grananje, ne računaju prilikom određivanja statistike zbirke test slučaja. U konkretnom slučaju to su ulazi kojima se označavaju jezici unetih reči (Engleski i Francuski). Kada se dva pomenuta ulaza izostave iz pokrivenih ulaze dobije se isti procenat pokrivenosti ulaza koji je i MaTeLo proračunao.

Dobijene zbirke apstraktnih test slučajeva se smeštaju u *xml* datoteke koje predstavljaju jedan od ulaza u MaTeLo Convertor. Upotrebom MaTeLo Convertora apstraktni test slučajevi se prevode u izvršive test slučajeve koji su prilagođeni nekoj platformi za automatsko izvršavanje test slučajeva. U konkretnom slučaju to je JUnit platforma. JUnit je platforma za automatsko izvršavanje test slučajeva koji su napisani u JAVA programskom jeziku. Da bi se izvršilo testiranje Babel Fish programa potrebno je definisati posebnu šemu prevođenja apstraktnih test slučajeva iz *xml* formata u *java* format. Šema prevođenja je data XSLT šemom koja je priložena uz model korišćenja programa. Šema prevođenja predstavlja vezu između apstraktnih i izvršivih test slučajeva. Na osnovu nje se definiše mehanizam provere rezultata dobijenih u toku izvršavanja test slučajeva i očekivanih rezultata definisanih u modelu korišćenja sistema. Kao što se može videti na Slici 2.12 XSLT predstavlja drugi ulaz u MaTeLo Convetror.

Nakon dobijenih izvršivih test slučajeva u *java* formatu zadati Babel Fish program se može testirati.

#### 2.4.3 Platforma i okruženje za automatsko izvršavanje test slučajeva

Platforma za testiranje je JUnit a okruženje u kome se testira zadati program je Eclipse ili NetBeans. Eclipse i NetBeans su razvojna okruženja za JAVA programski jezik i podržavaju JUnit, tako da se prevedeni test slučajevi mogu koristiti u oba razvojna okruženja.

Prvi korak u procesu testiranja upotrebom JUnit platforme je registrovanje test slučajeva kojima se testira željeni program. Za testiranje zadatog programa postoji namenski program implementiran u JAVA programskom jeziku koji omogućuje registrovanje test slučajeva i rukovanje sa dobijenim rezultatima testiranja ali on izlazi izvan teme ovog rada pa prema tome nije opisivan. Suština testiranja zadatog programa upotrebom JUnit platofrme iz perspektive MaTeLo alata jeste stvaranje izlazne (*log*) datoteke u kojoj se nalaze svi prikupljeni podaci u toku testiranja. Za svaki pokrenuti test slučaj postoje isti skup podataka koji daju informaciju da li je test uspešno ili neuspešno izvršen. Pomenuta *log* datoteka je neophodna za rad MaTeLo Testora i namenskog sintaksni analizator. Rezultati koji se nalaze u *log* datoteci su osnova za analizu i procenu pouzdanosti i srednjeg vremena do otkaza zadatog programa.

## 3. Koncept rešenja

Potreba za sintaksnim analizatorom (parserom) se pojavila kao posledica nepostojanja odgovarajućeg alata u priloženom MaTeLo skupa alata koji bi omogućio automatsko unošenje rezultata dobijenih u toku testiranja. Omogućeno je ručno (interaktivno) unošenje rezultata pomoću internet pretraživača (Internet Explorer, Firefox, Opera) gde se celokupna zbirka test slučajeva prikazuje kao internet stranica sve zajedno sa koracima izvršenja, vrednostima ulaza u sistem, vremenima izvršenja kao i ostalim dodatnim informacijama koje su dostupne. Ovo predstavlja prihvatljivo rešenje ukoliko postoji relativno mali broj rezultata koji je potrebno uneti, međutim, ukoliko je potrebno uneti veći broj rezultata, neophodno je omogućiti automatizaciju unošenja rezultata pošto je proces sam po sebi zamoran i podložan greškama.

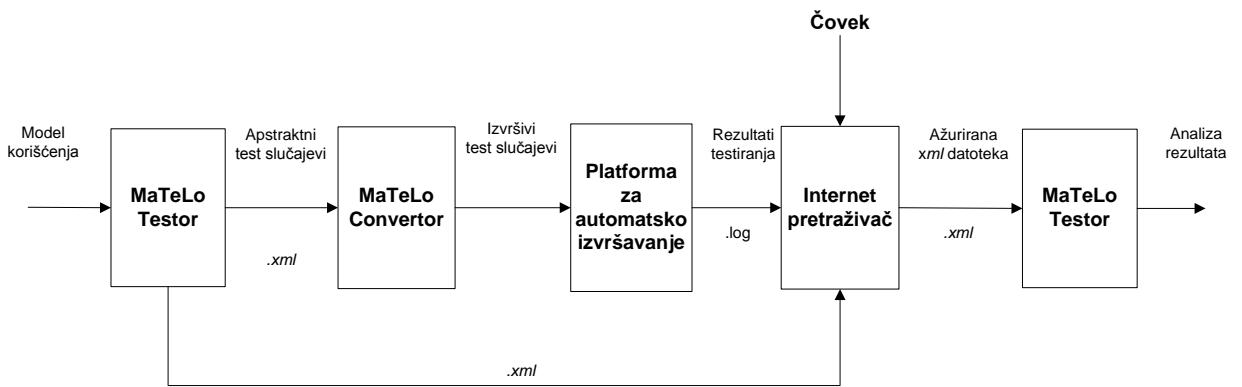
Osnovni zadatak sintaksnog analizatora je automatsko popunjavanje i formiranje izlazne *xml* datoteke na osnovu rezultata dobijenih izvršavanjem testnih slučajeva stvorenih od strane MaTeLo Testora.

U nastavku je opisana upotreba sintaksnog analizatora u MaTeLo skupu alata kao i arhitektura i osnovni principi rada sintaksnog analizatora.

### 3.1 Sintaksnii analizator u MaTeLo skupu alata

Sintaksnii analizator u MaTeLo skupu alata služi za automatizaciju procesa unosa rezultata koji su dobijeni u toku testiranja sistema. Na Slici 3.1 prikazan je osnovni (standardni) tok podataka MaTeLo alata bez upotebe sintaksnog analizatora gde se dobijeni rezultati runčno unose pomoću internet pretraživača.

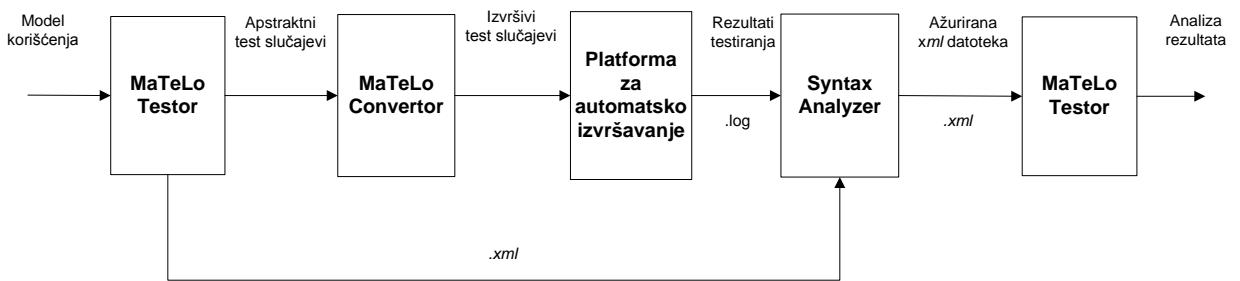
U toku podataka prikazanom na slici se podrazumeva da postoji napravljen model korišćenja sistema. Podaci koji se koriste u toku pravljenja modela su izostavljeni.



Slika 3.1 – Tok podataka u MaTeLo alatu sa ručnim unosom rezultata

Kao što se može videti na slici, model korišćenja sistema predstavlja ulaz MaTeLo Testor. MaTeLo Testor na osnovu modela korišćenja i podataka koji su u njemu sadržani stvara apstraktne test slučajeve na osnovu kojih se testira modelovani sistem. Svi apstraktni test slučajevi se smeštaju u izlaznu *xml* datoteku koja se takođe koristi za unos, odnosno ažuriranje dobijenih rezultata. Izlazna *xml* datoteka iz MaTeLo Testora predstavlja ulaz MaTeLo Convertora. MaTeLo Convertor na osnovu podataka o koracima izvršenja apstraktnih test slučaja i definisane XSLT datoteke prevodi apstraktne test slučajeve u izvršive test slučajeve željenog formata. Format (ekstenzija) datoteke u kojoj se nalaze izvršivi test slučajevi zavisi od ciljane arhitekture za automatsko izvršavanje test slučajeva. Platforma na kojoj se sistem testira omogućuje automatsko izvršavanje koraka test slučajeva kao i prikupljanje dobijenih rezultata. Dobijeni rezultati se smeštaju u posebnu (*log*) datoteku i ona predstavlja izlaz platforme za automatsko testiranje. Dobijene rezultate testiranja sistema je potrebno uneti u MaTeLo i ovo predstavlja deo čiju funkcionalnost sintaksni analizator treba da automatizuje. Jedini način unosa rezultata u *xml* datoteku koji je omogućen od strane MaTeLo alata je upotrebo nekog od internet pretraživača. Apstraktni test slučaji koji se nalaze u *xml* datoteci se učitavaju kao interaktivna internet stranica u kojoj se unose dobijeni rezultati. Na osnovu unetih rezultata pravi se *xml* datoteka koja pored podataka koji služe za prevođenje rezultata sadrži i unete rezultate test slučajeva prikupljene u procesu testiranja. MaTeLo Testor dobijenu *xml* datoteku sa rezultatima koristi kao ulaz i na osnovu nje automatski ažurira podatke o rezultatima test slučajeva. Opisani proces unosa rezultata mora biti urađen od strane čoveka. Ovakav način unosa je vremenski zahtevan i podložan greškama. Ovo predstavlja glavni problem koji treba da bude rešen upotrebom sintaksnog analizatora.

Na Slici 3.2 prikan je tok podataka MaTeLo alata sa upotrebom sintaksnog analizatora.

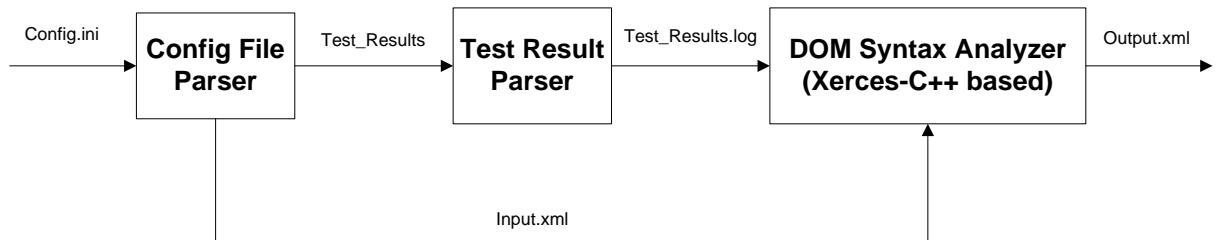


Slika 3.2 – Tok podataka u MaTeLo alatu sa upotrebom sintasnog analizatora

Sa upotrebom sintasnog analizatora tok podataka do unosa rezultata se bitnije ne menja. Jedina razlika jeste način rukovanja sa dobijenom (*log*) datotekom u kojoj se nalaze rezultati testiranja. Sintasni analizator nema potrebu za asistencijom od strane čoveka već automatizuje proces analize dobijenih rezultata i njihov unos u izlaznu *xml* datoteku. Ulaz sintasnog analizatora je *log* i *xml* datoteka. Prvo se analizira sadržaj *log* datoteke i ukoliko je ispravno definisana, grupišu se svi dobijeni podaci koji se odnose na pojedinačne test slučajeve. Potom se proverava sadržaj *xml* datoteke. Ukoliko je *xml* datoteka ispravno definisana i ako njen sadržaj odgovara sadržaju koji je očekivan od strane MaTeLo Testora, sintasni analizator unosi rezultate svakog pokrenutog test slučaja na unapred definisan način. Sadržaj dobijene *xml* datoteke je isti kao sadržaj datoteke koja je dobijena ručnim unosom pomoću internet pretraživača i predstavlja ulaz u MaTeLo Testor.

### 3.2 Arhitektura sintasnog analizatora

Sintasni analizator se sastoji iz nekoliko modula od kojih svaki modul čini posebnu logičku i funkcionalnu celinu. Blok šema namenskog sintasnog analizatora je prikazana na Slici 3.3. Moduli su zamišljeni tako da se u slučaju promene strukture ulaznih datoteka (recimo ulazne datoteka sa rezultatima) ili nekih drugih dodatnih zahteva omogući što lakše prilagođenje. U nastavku je opisani svi gradivni moduli sintasnog analizatora.



Slika 3.3 – Blok šema namenskog MaTeLo sintasnog analizatora

### 3.2.1 Sintaksni analizator datoteke sa ulaznim parametrima sistema

Sintaksni analizator ulazne datoteke sa parametrima sistema (eng. Config File Parser) omogućuje funkciju analiziranja datoteke u cilju učitavanja ulaznih parametara u sistema. Ovaj modul ima bitnu ulogu zato što on obezbeđuje komunikaciju korsnika sa sistemom preko ulazne datoteke. Ulazna datoteka sa parametrima sistema predstavlja jedini i obavezan parametar koji se mora definisati prilikom poziva sintaksnog analizatora. Ukoliko ona nije ispravno definisana parametri se pogrešno učitavaju što dovodi do greške ili još gore, može da dovede do neispravnog rada programa.

Ulazna datoteka sadrži sledeće podatke:

- Datoteka sa rezultatima testova (eng. test result file)
- Verzija sistema (eng. system version)
- Broj grupe testova (eng. test suite number)
- Ulazna *xml* datoteka
- Izlazna *xml* datoteka

Prilikom formiranja konfiguracione datoteke pomenuta polja moraju biti u istom redosledu u kom su ovde navedena. Svaka druga kombinacija se tretira kao greška. U daljem nastavku je opisano kako treba definisati svako polje konfiguracione daoteke.

#### 3.2.1.1 Datoteka sa rezultatima testova

Prvo polje u ulaznoj datoteci se koristi za definisanje ulazne datoteke u kojoj su smešteni dobijeni rezultati testiranja (eng. test result file). Sadržaj i struktura datoteke sa rezultatima su posebno opisani. Putanja i ime datoteke se definišu iza markera „test\_results=”. Sve što se pojavi iza markera se smatra putanjom uključujući i znakove razmaka (eng. space).

Primer ispravno definisanog polja:

```
test_results=..\..\example_test_results.log
```

#### 3.2.1.2 Verzija sistema

Verzija sistema nije parametar koji je neophodan za rad samog sintaksnog analizatora ali jeste obavezan za MaTeLo Testor. Verziju sistema MaTeLo Testor koristi kao jedan od parametara priikom analize testiranog sistema. Kako sintaksni analizator služi za formiranje *xml* datoteke koja je predviđena za korišćenje od strane MaTeLo Testora, ovaj parametar se mora

obezbediti. U suprotnom rezultati u *xml* datoteci se ne može automatski ažurirati (importovati) u MaTeLo Testor.

Primer ispravno definisanog polja:

```
system_version=debug
```

### 3.2.1.3 Broj grupe testova

Broj grupe testova (eng. test suite number) je još jedan od parametara koji nije neophodan za rad sintaksnog analizatora ali se ipak mora obezbediti zato što je korišćen u MaTeLo Testoru. Njegov unos je omogućen kako bi u potpunosti ostali dosledni i pružili informacije koje su predviđene. Sve unete informacije doprinose u procesu analize i značajno mogu olakšati dalji rad.

Primer ispravno definisanog polja:

```
test_suite_number=78
```

### 3.2.1.4 Ulazna xml datoteka

Ulazna *xml* datoteka je obavezan (eng. mandatory) parametar za rad sintaknog analizatora. Ona predstavlja početni *xml* dokument u koji se unosi dobijeni rezultati testiranja. Ulazna *xml* datoteka je izlaz iz MaTeLo Testora i sadrži informacije o automatski stvorenim testovima.

Primer ispravno definisanog polja:

```
input_xml_file=..\..\example_test_suite.xml
```

### 3.2.1.5 Direktorijum izlazne xml datoteke

Ovo polje je malo specifično. Njime se ne definiše putanja izlazne *xml* datoteke već se definiše samo izlazni direktorijum u kome je smeštena rezultujuća *xml* datoteka. Razlog ovome leži u činjenici da ukoliko se želi da u MaTeLo Testor automatski unesu rezultati svih testova potrebno je obezbediti da ime izlazne *xml* datoteke bude u sledećem obliku:

**ime\_datoteke\_sa\_testovima.verzija\_sistema**

Datoteka u kojoj se nalaze testovi predstavlja ime ulazne *xml* datoteke. Treba napomenuti da se iz polja gde je definisana ulazna *xml* datoteka izdvaja samo ime ulazne datoteke. Verzija sistema je obezbeđena kao poseban parametar u ulaznoj datoteci. Kako bi se formirala potpuna putanja nedostaje samo ime izlaznog direktorijuma. On se definiše ovim poljem. Na osnovu ova

tri parametra sintaksni analizator dinamički u toku izvršavanja formira potpunu putanju izlazne datoteke.

Primer ispravno definisanog polja:

```
output_xml_path=..\..\
```

### 3.2.1.6 Ispravno definisana datoteka sa ulaznim parametrima

```
test_results=..\..\example_test_results.  
log  
system_version=debug  
test_suite_number=78  
input_xml_file=..\..\example_test_suite.xml  
output_xml_path=..\..\
```

### 3.2.2 Sintaksni analizator datoteke sa rezultatima testova

Sintaksni analizator datoteke koja sadrži rezultate testova (eng. Test Result Parser) treba da omogući analizu i unos podataka u sistem koji predstavljaju rezultate pokrenutih testova. Datoteka (putanja) je definisana u ulaznoj datoteci kao poseban parametar. Ukoliko je putanja ulazne datoteke ispravna, ona se učitava u sistem i otpočinje se sa analizom sadržaja. U suprotnom se javlja poruka o grešci. Svi rezultati i svi podaci koji čine rezultat testa (ime test, vreme izvršenja, itd) se moraju nalaziti u tačno definisanom obliku i redosledu kako bi se omogućio uspešan unos u sistem.

Bitno je napomenuti da broj testova u ulaznoj *xml* datoteci, odnosno u datoteci u koju se smeštaju učitani rezultati, mora biti isti. U trenutnoj verziji sintaksnog analizatora unošenje pojedinačnih rezultata nije omogućeno. Na primer nije moguće unošenje rezultata testova 3, 4, 7 i 9 ukoliko testna grupa ima ukupno 10 testova. Moraju se obezrediti rezultati svih 10 testova i to u redosledu u kom se testovi nalaze u početnoj *xml* datoteci. U suprotnom ponašanje sintaksnog analizatora nije poznato.

Rezultat testa sadrži sledeće podatke:

- Zaglavlj rezultata
- Ime testnog slučaja (eng. test case name)

- Rezultat testiranja (eng. test verdict)
- Korak na kome se desila greška (eng. test step error)
- Vreme izvršenja testa (eng. test execution time)

Datoteka sa rezultatima testova može imati proizvoljan broj rezultata. Sledi detaljan opis svakog polja datoteke koja sadrži rezultate testova.

### 3.2.2.1 Zaglavljje rezultata

Zaglavljje rezultata se koristi kako bi se omogućilo lakše vizuelno, sintaksno i semantičko tumačenje sadržaja datoteke. Naime, kada se datoteka analizira vizuelno od strane čoveka, intuitivno je jasno da se iza ovog zaglavlja nalaze informacije koje se odnose na rezultate izvršenih testova. Isto tako, prilikom analize, zaglavljje omogućava sintaksnom analizatoru da grupiše određeni deo sintakse kojoj može da pridruži unapred određenu semantiku. Nakon toga se pristupa proveri vrednosti očekivanih markera i izdvajajuju unetih parametara.

Zaglavljje korišćeno za testiranje zadatog programa:

- TEST\_CASE\_RESULT

Iza ovog zaglavlja slede ostali podaci o koji čine rezultat.

### 3.2.2.2 Ime testnog slučaja

Ovo polje predstavlja ime testnog slučaja na koji se odnose dobijeni rezultati i obavezno je za usnos. Iako je obavezno ono se ne unosi u resultantnu *xml* datoteku već se obezbeđuje usaglašenost sa narednim verzijama sintaksnog analizatora. Svaki red u datoteci se mora završiti sa „;” (tačka-zarez) znakom.

Primer ispravno definisanog polja:

```
test_case_name=test_sute_example1;
```

### 3.2.2.3 Rezultat testiranja

Rezultat testiranja predstavlja vrednost kojom se označava da li je test uspešno izvršen ili ne. Postoje tri definisane vrednosti koje se mogu dobiti kao rezultat pokretanja testa:

- Uspešan (eng. Pass)
- Neuspešan (eng. Fail)
- Nije izvršen (eng. No run)

„Pass” vrednost označava da je test uspešno izvršen, odnosno, da nije došlo do pojave greške prilikom izvršenja testa.

„Fail” vrednost označava da je tokom izvršavanja došlo do pojave greške, odnosno, da test nije uspešno izvršen.

„No run” označava da nije test nije pokrenut.

Primer ispravno definisanog polja:

```
test_verdict=Fail;
```

### 3.2.2.4 Korak na kome se desila greška

Iako možda na prvi pogled nije logično ali ovo polje se uvek popunjava bez obzira na krajnji rezultat pokrenutog testa (eng. Pass, Fail, No run). Podatak o korak na kome se desila greška se koristi od strane MaTeLo Testora za analizu i formiranje statističkih podataka o testiranom sistemu. Kada se test ne pokrene (No run) ili kad se uspešno izvrši (Pass) vrednost unetog parametra je nula. U slučaju da test nije prošao (Fail) vrednost koja se unosi je broj koraka na kome se zaista desila greška. U opštem slučaju parametar koji se unosi mora imati vrednost u granicama [0, maksimalan broj koraka pokrenutog testa].

Primer ispravno definisanog polja:

```
test_step_error=7;
```

### 3.2.2.5 Vreme izvršenja testa

Značenje ovog polja nema potrebe posebno objašnjavati jer je intuitivno jasno šta predstavlja. Jedino što se može napomenuti jeste da je vreme izvršenja test predstavljeno u decimalnom obliku. Prevođenje u decimalni format se vrši po satima. Priloženi primer ilustruje prevođenje vremena iz *hh:mm:ss* predstave u decimalnu predstavu.

Primer:

- Vreme trajanja testa u formatu *hh:mm:ss* = 2:45:45
  1. 2 sata =  $2 \text{ sata} * (1 \text{ sat} / 1 \text{ sat}) = 2 \text{ sata}$
  2. 45 minuta =  $45 \text{ minuta} * (1 \text{ sat} / 60 \text{ minuta}) = 45 / 60 \text{ sata} = 0.75 \text{ sata}$
  3. 45 sekundi =  $45 \text{ sekundi} * (1 \text{ sat} / 3600 \text{ sekundi}) = 45 / 3600 \text{ sata} = 0.0125 \text{ sata}$
- Sumiranjem svih vrednosti dobija se konačna vrednost u decimalnom formatu:  

$$2 \text{ sata} + 0.75 \text{ sata} + 0.0125 \text{ sata} = 2.7625 \text{ sata}$$

Konačan izgled polja u datoteci je:

```
test_execution_time=2.7625;
```

### 3.2.2.6 Ispravno definisana datoteka sa rezultatima testova

Unutar datoteke se može nalaziti proizvoljan broj rezultata testova. U primeru su to tri rezultata. Jedino na šta treba obratiti pažnju jest da u opštem slučaju broj rezultata testova u datoteci u kojoj se čuvaju rezultati i broj testnih slučajeva u ulaznoj *xml* datoteci mora biti isti.

```
TEST_CASE_RESULT
  test_case_name=TS_1_
  sv2_1;
  test_verdict=Pass;
  test_step_error=0;
  test_execution_time=0.
  0042;

TEST_CASE_RESULT
  test_case_name=TS_1_
  sv2_2;
  test_verdict=Pass;
  test_step_error=0;
  test_execution_time=0.
  0053;

TEST_CASE_RESULT
  test_case_name=TS_1_
  sv2_3;
  test_verdict=Pass;
  test_step_error=0;
  test_execution_time=0.
```

0039;

### 3.2.3 DOM Sintaksni analizator

Ovo je modul koji je nešto durgačiji od predhodna dva modula. DOM (eng. Document Object Model) sintaksni analizator omogućava rad sa *xml* datotekama. Činjenica koju treba uzeti u obzir jeste da ovaj modul nije pisan od početka (od “nule”) samo za ovu svrhu već se on zasniva na korišćenju već postojećih DOM sintaksnih dokumenata. Xerces projekat predstavlja grupu sintaksnih analizatora koji su razvijeni od strane Apache organizacije (eng. Apache Software Organization), koji treba da omoguće lako i jednostavno održavanje postojećih i stvaranje novih *xml* datoteka. U suštini svi sintaksni analizatori obezbeđuju isti standardizovani DOM API. Jedina razlika jeste u programskom jeziku u kome je sintaksni analizator implementiran (C++, Java, Perl). Prilikom implementacije namenskog sintaksnog analizatora za MaTeLo skupa alata koršićen je Xerces-C++ *xml* sintaksni analizator. Sintaksni analizator implementiran u C++ programskom jeziku ima znatno više prednosti u odnosu na iste sintaksne analizatore implementirane u Java ili Perl programkom jeziku. Ako se izuzme brzina rada koja može biti bitan faktor ukoliko se radi sa vekim *xml* datotekama, prednost je to što je Xerces-C++ sintaksni analizator dostupan i na Linux operativnom sistemu. Na ovaj način eventualna migracija na Linux platformu ne bi trebala pretstavljati veći problem.

Pored elementarnih funkcija otvaranja i zatvaranja *xml* datoteka najviše se koriste funkcije koje rukuju sa atributima čvorova (eng. node attributes) *xml* dokumenta (neophodno je poznavanje strukture *xml* datoteke). Rukuje se isključivo sa atributima zato što MaTeLo Testor očekuje da sve informacije koje daju informacije o izvršenju testova (verdict, step error, time execution, itd) budu date u vidu atributa. Atributi se dodaju na tri različita tipa čvora:

- čvor zbirke test slučaja (eng. testsuite node)
- čvor testnog slučaja (eng. testcase node)
- čvor prelaska iz tekućeg u naredno stanje (eng. transition node)

Vrednosti svih atributa se popunjavaju na osnovu podataka koje su dobijeni iz ulazne datoteke sa rezultatima testova i ulazne datoteke sa parametrima.

#### 3.2.3.1 Čvor zbirke slučaja

Atributi koji se prilikom sintaksne analize dodaju *testsuite* čvor su:

- spisak prijavljenih otkaza (eng. list\_of\_fault\_reports)
- broj uspešno izvršenih testova (eng. number\_of\_test\_cases\_pass)
- ime projekta (eng. project\_name)
- ime programa (eng. software\_name)
- verzija programa (eng. softwre\_version)
- broj zbirke test slučaja (eng. test\_suite\_number)
- ukpno vreme izvršenja (eng. total\_time)
- ukupan broj uspešno izvršenih testnih koraka (eng. number\_of\_test\_step\_pass)
- ukupan broj koraka na kojima je otkrivena greška (eng. totalfaildstep)

Spisak prijavljenih otkaza - Daje informaciju o broju testova kod kojih se u toku izvršavanja pojavila greška (verdict = Fail).

Broj uspešno izvršenih testova – Predstavlja ukupan broj testova čije izvršenje je proteklo bez pojave greške.

Ime projekta – Označava ime projekta. Vrednost atributa nije obavezna.

Ime programa – Označava ime programa koja se testira. Vrednost atributa nije obavezna.

Verzija programa – Predstavlja verziju testiranog sistema.

Broj zbirke test slučaja – Predstavlja broj grupe testova. Vrednost atributa nije obavezna.

Ukpno vreme izvršenja – Atribut predstavlja ukupno vreme izvršenja zbirke testova.

Ukupan broj uspešno izvršenih testnih koraka – Daje podatak o okupnom broju uspešno izvršenih koraka svih pokrenutih testova.

Ukupan broj koraka na kojima je otkrivena greška – Daje podatak o ukupnom broju grešaka koje su se pojavile prilikom izvršenja svih testova.

### **3.2.3.2 Čvor test slučaja**

Atributi koji se prilikom sintaksne analize dodaju u *testcase* čvor su:

- provera rezultata testa (eng. check)
- prelaz na kome se desila greška (eng. steperror)
- vreme izvršenja testa (eng. time)

Provera rezultata testa – Vrednost ovog atributa predstavlja informaciju o konačnom rezultatu pokrenutog testa. Rezultat mora biti jedan od tri moguća stanja testa.

Prelaz na kome se desila greška – Definiše koraku (prelaz) na kom se desila greška. Ukoliko test nije pokrenut (No Run) ili je uspešno izvršen vrednost ovog atributa je nula (steperror=”0”).

Vreme izvršenja testa – Predstavlja vreme izvršenja testnog slučaja. Ukoliko test nije pokrenut (No Run) atribut nema vrednost (time=””).

### 3.2.3.3 Čvor prelaza iz tekućeg u naredno stanje

Prilikom sintakne analize u *transition* čvor se dodaje samo jedan atribut:

- Prelaz na kome se desila greška (eng. steperror)

Prelaz na kome se desila greška – Atribut se dodaje *transition* čvoru samo ukoliko je došlo do greške prilikom izvršavanja testa i to tačno na *transition* čvor koji odgovara koraku izvršenja testa. Treba naglasiti razliku između dodavanja atribura i dodavanja vrednosti atributa. Na primer, u slučaju *testcase* čvora atribut *time* se uvek mora dodati ali njegova vrednost zavisi od konačnog rezultata testa. Ukoliko test nije pokrenut atributu se ne dodaje nikakva vrednost.

Konkretni *steperror* atribut se dodaje samo onda kad se desila greška i vrednost atributa je jednak koraku na kome se desila greška.

## 4. Programsко рење

MaTeLo sintaksni analizator je implementiran u *C/C++* programskom jeziku korišćenjem *Microsoft Visual C++* razvojnog okruženja.

Kao što je u predhodnom poglavlju pomenuto, za rad i manipulaciju sa *xml* datoteka i njenim sadržajem koristi se Xerces-C++ sintaksni analizator. Celokupna funkcionalnost Xerces-C++ sintaksnog analizatora je unutar projekta dostupna kao dinamička biblioteka (*dll*).

Dinamička biblioteka predstavlja veoma praktičan način za realizaciju različitih programskih modula iz više razloga. Ovo bi mogao biti jedan od načina integracije sintaksnog analizatora u BBT sistem razvijen u Institutu RT-RK.

Sledi tabela sa spiskom svih datoteka korišćenih pri implementaciji programskog rešenja. Uz svaku datoteku dat je kratak opis sadržaja u njoj. U nastavku je detaljno opisan API modula koji su korišćeni prilikom realizacije MaTeLo sintaksnog analizatora. Detaljnije informacije o API-u Xerces-C++ biblioteke su dostupne u zvaničnoj Apache dokumentaciji (*Xerces-C++ API Reference*).

Ime datoteke	Sadržaj
Config.h	Definicija klase koja predstavlja sadržaj ulazne datoteke sa parametrima (konfiguracione datoteke).
Config.cpp	Implementacija metoda istoimene klase.
ConfigFileParser.h	API sintaksnog analizatora ulazne datoteke sa parametrima.
ConfigFileParser.cpp	Implementacija API-a sintaksnog analizatora ulazne datoteke.
DOMSyntaxAnalyzer.h	API sintaksnog analizatora <i>xml</i> datoteka.

DOMSyntaxAnalyzer.cpp	Implementacija API-a sintaknog analizatora za rad sa <i>xml</i> datotekama.
DOMTreeErrorReporter.hpp	API modula za otkrivanje i signalizaciju grešaka.
DOMTreeErrorReporter.cpp	Implementacija API-a modula za otkrivanje grešaka.
SyntaxAnalyzerErrorHandler.hpp	API modula rukovaoca greškama.
SyntaxAnalyzerErrorHandler.cpp	Implementacija API-a rukovaoca greškama.
TestCaseResult.h	Klasa koja sadrži sve informacije o učitanim rezultatima testova.
TestCaseResult.cpp	Implementacija metoda istomene klase.
TestResultParser.h	API sintaknog analizatora ulazne datoteke koja sadrži rezultate testova.
TestResultParser.cpp	Implementacija API-a sintaksnog analizatora datoteke sa rezultatima testova.
XercesString.hpp	Klasa koja omogućava prebacivanje (konverziju) ASCII znakova u format koji koristi Xerces-C++ sintaksnog analizatora.
XercesString.cpp	Implementira metode istoimene klase.
Constants.h	Definicija svih konstanti korišćenih u programu.
ReturnCode.h	Spisak svih povratnih vrednosti programa.
Main.h	API glavnog programa.
main.cpp	Implementacija API-a glavnog programa.

Tabela 4.1 – Datoteke programskog rešenja.

## 4.1 Config klasa

Config klasa je zamišljena tako da objedini sve parametre koji su učitani iz ulazne datoteke u kojoj se definišu sve potrebne informacije za rad sintaknog analizatora. Ova klasa omogućava da se iz bilo kog dela programa jednostavno pristupa unetim parametrima. Uneti parametri se na početku programa učitavaju preko posebnog nameskog sintaksnog analizatora koji je prilagođen strukturi ulazne datoteke.

Definicija klase:

```
class Config{  
    string configfile;  
    string testResultFile;  
    string systemVersion;  
    string testSuiteNumber;  
    string inputXmlFile;  
    string outputXmlDir;  
public:  
    Config(void);  
    ~Config(void);  
...  
Standardne set i get metode.  
...  
    string outputFilePathForMaTeLo();  
};
```

#### 4.1.1 configFile atribut

Definiše putanju ulazne datoteke iz koje će biti učitani parametri koje koristi sintaksni analizator u toku svog rada. Ovaj atribut je zamišljen da se u narednim verzijama sistema smešta u vidu komentara u izlaznu *xml* datoteku kako bi se u slučaju pojave grešaka olakšalo rešavanje problema.

#### 4.1.2 testResultFile atribut

Predstavlja putanju ulazne datoteke u kojoj se nalaze rezultati testova koji se koriste za popunjavanje izlazne *xml* datoteke.

#### 4.1.3 systemVersion atribut

Atribut označava verziju testiranog sistema koja se unosi u uzlazu *xml* datoteku.

#### 4.1.4 testSuiteNumber atribut

Atributu predstavlja redni broj grupe testova koji je korišćen za testiranje sistema.

#### 4.1.5 inputXmlFile atribut

InputXmlFile atribut predstavlja atribut kojim se definiše putanja ulazne datoteke u kojoj se nalaze testovi stvoreni od strane MaTeLo Testora.

#### 4.1.6 outputXmlDir atribut

Atribut predstavlja izlazni derktorijum u koji se smešta rezultujuća izlazna *xml* datoteka.

#### 4.1.7 outpuFilePathForMaTeLo metoda

Na osnovu atributa verzije sistema i putanje izlaznog direktorijuma formira ime izlazne *xml* datoteke koje je očekivano od strane MaTeLo Testora (odeljak 3.2.1.5).

## 4.2 TestResult klasa

TestResult klasa predstavlja grupisanje (agregaciju) svih podataka koji čine potpun rezultat pokrenutog testa. Svi podaci sadržani u rezultatima testova koji su dobijeni u priloženom JUnit primeru su korišćeni od strane MaTeLo Testora. Prema tome svi podaci su definisani u klasi. U opštem slučaju klasa može proširiti broj sadržanih podataka koji se direktno ne moraju koristit od strane MaTeLo alata.

Definicija klase:

```
class TestCaseResult{  
public:  
    string testCaseName;  
    string testVerdict;  
    string testStepError;  
    string testExecutionTime;  
  
    TestCaseResult(void);  
    ~TestCaseResult(void);  
};
```

#### 4.2.1 testCaseName atribut

Atribut predstavlja ime testa na koji se odnose dobijeni rezultat.

#### 4.2.2 testVerdict atribut

Atribut predstavlja konačnu ocenu (presudu) pokrenutog testa.

#### 4.2.3 testStepError atribut

Atribut označava redni broj koraka na kome se desila greška prilikom izvršavanja testa

#### 4.2.4 testExecutionTime atribut

Atributom predstavlja vreme izvršenja testa.

### 4.3 ConfigFileParser modul

ConfigFileParser modul omogućava parsiranje ulazne (*ini*) datoteke sa parametrima koji se neophodni za rad sintaksnog analizatora.

API:

- *int ParseConfigFile (Config \* config)*

#### 4.3.1 int ParseConfigFile (Config \* config)

Opis: Funkcija na osnovu putanje ulazne datoteke koja je definisana u *config* parametru koji je prosleđen prilikom poziva funkcije analizira konkretnu datoteku i učitava parametre iz nje. Učitane parametre smešta u promenjivu koja je prosleđena po referenci kako bi se parametri mogli koristiti u ostalim delovima programa.

Ulagni parametri:

1. *config* - Referenca na objekat Config klase u se smeštaju parametri učitani iz ulazne datoteke.

Povratna vrednost: Kod greške.

## 4.4 TestResultParser modul

TestResultParser modul omogućava sintaksnu analizu datoteke koja sadrži rezultate testnih slučajeva. Testni slučajevi su stvoreni od strane MaTeLo Testora i na osnovu rezultata tih testova biće formirana nova izlazna *xml* datoteka.

API:

- *int ParseTestResult (fstream \* file, list< TestCaseResult > \* list)*

### 4.4.1 int ParseTestResult (fstream \* file, list< TestCaseResult > \* list)

Opis: Omogućava sintaksnu analizu ulazne datoteke u kojoj se nalaze rezultati testnih slučajeva (*test case result*). Sve informacije vezane za jedan testni slučaj se smeštaju u objekat klase TestCaseResult. Ovako grupisani rezultati se ulančavaju u listu kako bi se mogli koristiti u ostalim delovima programa.

Ulagni parametri:

1. *file* – Referenca na datoteku iz koje se čitaju rezultati.
2. *list* – Referenca na listu objekata klase TestCaseResult u koju se uvezuju učitani rezultati.

Povratna vrednost: Kod greške.

## 4.5 DOMSyntaxAnalyzer modul

DOMSyntaxAnalyzer se koristi za sintaksnu analizu i rad sa *xml* datotekama. Na osnovu učitanih rezultata testova i rezultata sintaksne analize formira izlaznu *xml* datoteku. Ovaj modul koristi funkcije iz Xerces-C++ biblioteke.

API:

- *int InitializeDOMSyntaxAnalyzer (void )*
- *bool ParseXMLFile (XercesDOMParser \* parser, string inXmlFile)*
- *int SaveXML (DOMDocument \* doc, string outXmlFile)*
- *void FillSuiteNod (DOMElement \* suite, list< TestCaseResult > \* results, Config \*config)*
- *void FillTestCaseNod (list<DOMElement\*> \*domElementList, list<TestCaseResult> \*results)*
- *DOMElement\* GetDOMElement (DOMElement \* root, string elementName)*
- *void GetAllDOMElement (DOMElement \* root, string elementName, list< DOMElement \* > \* list)*

- 
- *int GetDOMElementCount (DOMElement \* root, string elementName)*

#### **4.5.1 int InitializeDOMSyntaxAnalyzer (void)**

Opis: Inicijalizuje Xerces-C++ biblioteke.

Ulazni parametri: Nema.

Povratna vrednost: Kod greške.

#### **4.5.2 bool ParseXMLFile (XercesDOMParser \* parser, string inXmlFile)**

Opis: Vrši učitavanje i sintaksnu analizu ulazne *xml* datoteke.

Ulazni parametri:

1. *parser* – Referenca na XercesDOMParser sa svim podešenim parametrima.
2. *inXmlFile* – Referenca na ulaznu *xml* datoteku.

Povratna vrednost: Indikacija o uspešnosti rada (true/false).

#### **4.5.3 int SaveXML (DOMDocument \* doc, string outXmlFile)**

Opis: Omogućuje pohranjivanje (snimanje) formirane *xml* datoteke u DOM predstavi u željenu izlaznu *xml* datoteku.

Ulazni parametri:

1. *doc* – Referenca na izlaznu *xml* datoteka u DOM predstavi.
2. *outXmlFile* – Referenca na izlaznu *xml* datoteku u ASCII predstavi.

Povratna vrednost: Kod greške.

#### **4.5.4 DOMElement\* GetDOMElement (DOMElement \* root, string elementName)**

Opis: Pronalazi prvi DOM element sa zadanim imenom.

Ulazni parametri:

1. *root* – Referenca na koren čvor (root node) traženog elementa.
2. *elementName* – Ime traženog elementa.

Povratna vrednost: Referenca na traženi element.

#### **4.5.5 void GetAllDOMElement (DOMElement \* root, string elementName, list< DOMElement \* > \* list)**

Opis: Pronalazi sve DOM elemente sa zadanim imenom i ulančava ih u listu.

Ulazni parametri:

1. *root* – Referenca na koreni čvor traženog elementa.
2. *elementName* – Ime traženog elementa.
3. *list* – Referenca na lista u koju se dodaju pronađeni elementi.

Povratna vrednost: Nema.

#### **4.5.6 int GetDOMElementCount (DOMElement \* root, string elementName)**

Opis: Određuje broj elemenata u listi sa zadanim imenom.

Ulazni parametri:

1. *root* – Referenca na koreni čvor traženog elementa..
2. *elementName* – Ime traženog elementa.

Povratna vrednost: Kod greške.

#### **4.5.7 void FillSuiteNod (DOMElement \* suite, list< TestCaseResult > \* results, Config \*config)**

Opis: Na osnovu popunjениh *testcase* čvorova u izlaznoj *xml* datoteci računa i dodaje potrebne informacije koje se odnose na *testsuite* čvor. Sve informacije se dodaju kao atributi *testsuite* čvora.

Ulazni parametri:

1. *suite* – Referenca na *suite* čvor koji se popunjava.
2. *result* – Referenca na listu sa rezultatima testova.
3. *config* - Referenca na objekat Config klase.

Povratna vrednost: Nema.

#### **4.5.8 void FillTestCaseNod (list<DOMElement\*> \*domElementList, list<TestCaseResult> \*results)**

Opis: Na osnovu učitanih rezultata testova i analizirane ulazne *xml* datoteke vrši se popunjavanje rezultata. Rezultati se dodaju u vidu *xml* atributa. Atribut se dodaje *testcase* čvorovima.

Ulazni parametri:

1. *domElementList* – Referenca na listu *testcase* čvorova kojima se dodaju rezultati.
2. *result* – Referenca na listu sa učitanim rezultatima testova.

Povratna vrednost: Nema.

## 5. Rezultati

U ovom poglavlju predstavljeni su dobijeni rezultati testiranja zadatog programa, rezultati koji se odnose na procenjenu pouzdanost i srednjeg vremena otkaza.

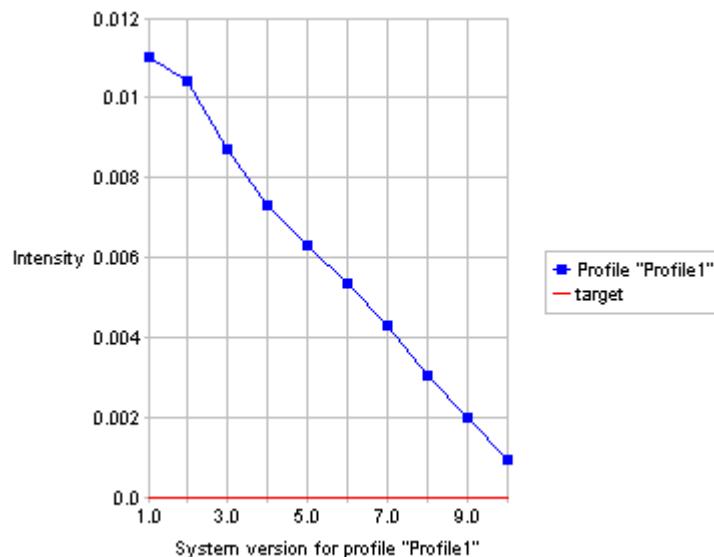
### 5.1 Rezultati testiranja

Rezultati dobijeni u toku testiranja programa se nalaze u Tabeli 5.1.

Zbirka	Broj uspešno izvršenih test slučaja	Broj neuspešno izvršenih test slučaja
ts_1	360	40
ts_2	364	36
ts_3	368	32
ts_4	372	28
ts_5	376	24
ts_6	380	20
ts_7	384	16
ts_8	388	12
ts_9	392	8
ts_10	396	4

Tabela 5.1 – Rezultati testiranja

U tabeli su prikazani dobijeni rezultati svih pokrenuti zbirki test slučajeva kojima je program testiran. Za svaku zbirku test slučajeva dat je broj test slučajeva koji su uspešno izvršeni, odnosno pri čijem izvršavanju nije prijavljena greška u toku rada kao i broj test slučajeva koji su neuspešno izvršeni, odnosno pri čijem korišćenju je došlo do prijavljivanja greške. Može se videti da broj nepoloženih test slučajeva opada u svakoj verziji sistema. Trend opadanja broja nepoloženih test slučajeva, odnosno smanjenje broja grešaka u radu sistema je realan i poželjan u toku razvoja svakog programa. U završnoj verziji programa poželjno je da se svi test slučajevi uspešno izvrše, ali je to u praksi nemoguće postići. Na početku testiranja programa, od 400 pokrenutih test slučajeva ukupno 40 test slučajeva (što je 10 % od ukupnog broja) uspelo je da otkrije grešku u radu programa. Pre narednog testiranja programa podrazumeva se da su prijavljene greške otklonjene kako bi dalje testiranje imalo smisla. U poslednjoj verziji programa od ukupno 400 pokrenutih test slučajeva, svega 4 test slučaja (1 %) je uspelo da otkrije grešku u radu programa. Ovako mali broj otkrivenih grešaka toku korišćenja programa se može smatrati kao zadovoljavajući i program se može smatrati spremnim za upotrebu. Na Slici 5.1 grafički je prikazan trend prijavljivanja broja grešaka sa porastom broja testiranih verzija programa.



Slika 5.1 – Trend otkrivanja grešaka u toku testiranja

## 5.2 Rezultati analize podataka dobijenih u toku testiranja

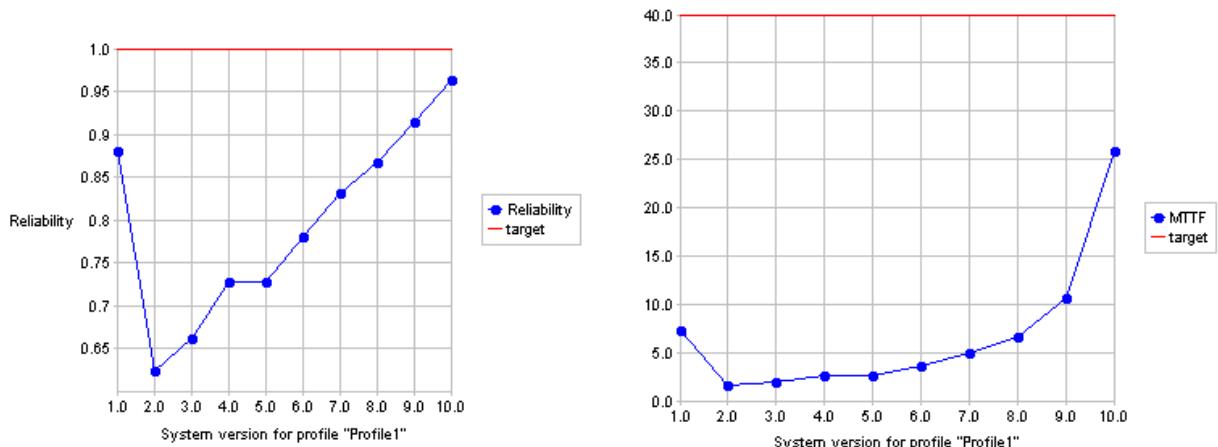
MaTeLo alat upotrebom Milerovog modela (odeljak 2.1.4) daje procenu pouzdanosti programa na osnovu ukupnog broj uspešno i neuspešno izvršenih test slučajeva i broja aktivacija programa u nekoj jedinici vremena. Ovo su tri parametra koja su obavezna i pomoću kojih se

određuje pouzdanost i srednje vreme do otkaza testiranog programa. Kao što je ranije pomenuto u opisu MaTeLo Testora (odeljak 2.2.2) dobijene karakteristika sistema se odnose na tekuću verziju sistema, dok se kumulativne procene karakteristika odnose na opštu procenu sistema.

U cilju što boljeg razumevanja načina na koji MaTeLo daje procenu karakteristika sistema urađene su dve procene zadatog programa sa istim rezultatima testiranja, ali sa različitim brojem aktivacija. U nastavku su prikazani rezultati pouzdanosti i srednjeg vremena do otkaza po verziji sistema, odnosno kumulativna pouzdanost i kumulativno srednje vreme do otkaza za program u slučaju jedne i dve aktivacije.

Za testirani program željena pouzdanost je 1 (100%). Pouzdanost 1 predstavlja teorijsku vrednost koju je u praksi nemoguće postići. U parksi se teži da pouzdanost i srednje vreme do otkaza imaju što je moguće veću vrednost. Željena vrednost srednjeg vremena do otkaza predstavlja vremenski period upotrebe programa u kom se neće pojaviti ni jedna preostala greška. Željeno srednje vreme do otkaza za zadati program je 40 dana.

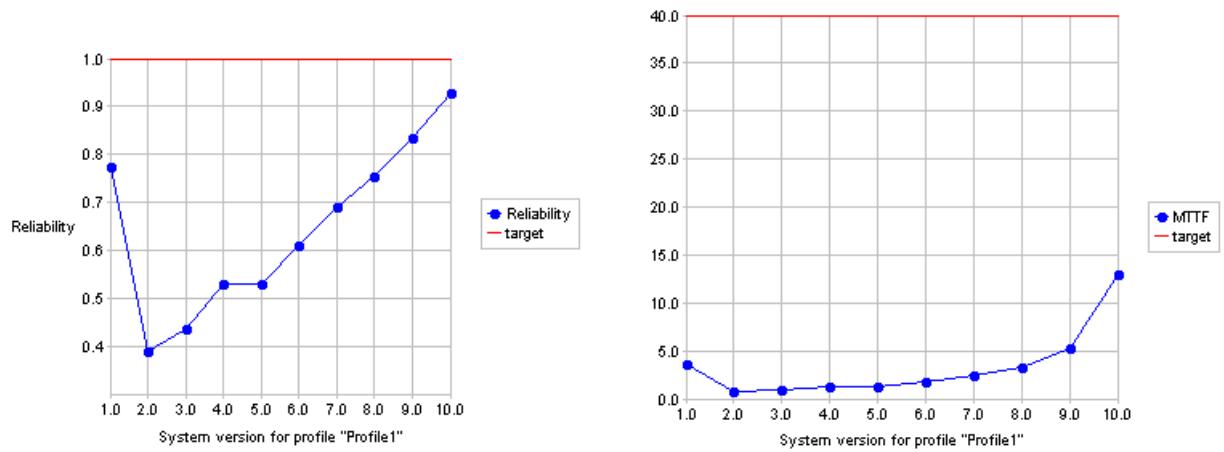
### 5.2.1 Pouzdanost i MTTF za svaku verziju programa



Slika 5.2 – Pouzdanost i MTTF programa (1 aktivacija)

Na Slici 5.2 i Slici 5.3 prikazani su grafici izračunate pouzdanosti programa i procenjenog srednjeg vremena do otkaza za jednu i dve aktivacije programa u toku jednog dana respektivno. Pouzdanost sistema (levi grafik na slikama 5.2 i 5.3) se računa na osnovu matematičkog obrazca (1) koji je dat u opisu rada MaTeLo Testora (odeljak 2.2.2). U svrhu što bolje analize dobijenih rezultata i lakšeg praćenja obrazac je ponovo dat:

$$R(t) = (p_u)^{kt}$$



Slika 5.3 – Pouzdanost i MTTF programa (2 aktivacije)

Na graficima se može uočiti su dobijene krive pouzdanosti i MTTF na oba grafika skoro iste. Naime, verovatnoća korišćenja programa ( $p_u$ ) koja se izračunava na osnovu ukupnog broja test slučajeva koji su uspešno i neuspešno izvršeni je identična za iste verzije programa bez obzira na broj aktivacija. Vrednosti verovatnoća korišćenja, pouzdanosti i MTTF za svaku verziju programa su date u Tabeli 5.2.

Verzija	Verovatnoća korišćenja	Pouzdanost (1 akt.)	MTTF (1 akt.)	Pouzdanost (2 akt.)	MTTF (2 akt.)
1	0.880088	0.880088	7.339419	0.774554	3.669709
2	0.704136	0.623744	1.657762	0.389056	0.828881
3	0.767946	0.661294	1.952410	0.437309	0.976205
4	0.812362	0.727048	2.663644	0.528598	1.331822
5	0.844107	0.726814	2.660515	0.528259	1.330258
6	0.868441	0.780363	3.552967	0.608966	1.776484
7	0.888130	0.830990	4.916807	0.690544	2.458404
8	0.905690	0.867699	6.558500	0.752901	3.279250
9	0.920982	0.913529	10.564582	0.834535	5.282291
10	0.934785	0.962809	25.888323	0.927001	12.944161

Tabela 5.2 – Verovatnoća korišćenja, pouzdanost i MTTF za 1 i 2 aktivacije

Jedini različit parametar prilikom računanja je broj aktivacija ( $k$ ). Ovo predstavlja parametar kojim se daje informacija o frekvenciji korišćenja sistema. Ako se broj aktivacija definiše kao frekvencija korišćenja sistema, logično je da sistem sa istim karakteristikama za različite frekvencijama korišćenja ima različite pouzdanosti. Iz Tabele 5.2 vidi se da je verovatnoća upotrebe programa uvek manja od 1 za sve verzije programa bez obzira na broj aktivacija, pa se na osnovu datog obrazca može zaključiti da se pouzdanost za istu verziju programa sa povećanjem broja aktivacija smanjuje, što se može i videti u Tabeli 5.2. Prema tome, dobijene pouzdanosti za sve verzije programa u slučaju jedne aktivacije su veće nego dobijene pouzdanosti u slučaju dve aktivacije programa.

Na graficima pouzdanosti može se videti da se u opštem slučaju pouzdanost svake naredne verzije raste odnosno teži idealnoj pouzdanosti. Izuzetak je početna verzija sistema. Razlog zašto početna verzija programa ima tako veliku pouzdanost u odnosu na narednu verziju leži u Milerovim koeficijentima. MaTeLo koristi Milerov model za određivanja pouzdanosti sistema tako što se prilikom izračunavanja pouzdanosti sistema daje informacija o predhodnom korišćenju sistema. Informacije o predhodnom korišćenju se odnose na broj uspešnih i neuspečnih korišćenja sistema, odnosno na broj prijavljenih grešaka. Prilikom računanja početne verzije sistema po Milerovom modelu prepostavlja se da je broj uspešnih i neuspešnih korišćenja 1 pa se zato dobije velika pouzdanost programa (0.880088 za 1 akt. i 0.774554 za 2 akt.). Na osnovu rezultata testiranja (Tabela 5.1) vidi se da je u toku testiranja prve verzije sistema otkriveno 40 grešaka, tj. program je 40 puta neuspešno korišćen dok je 360 puta uspešno korišćen. Ovi podaci predstavljaju nove podatke o korišćenju programa i unose se kao Milerovi parametri pre analize druge verzije programa. Novi podaci o korišćenju sistema daju informaciju o većem broju grešaka u programu od prepostavljene jedne greške što se i odražava na pouzdanost druge verzije sistema (0.623744 za 1 akt i 0.389056 za 2 akt.). U toku testiranja programa pronalaženo je sve manje grešaka u radu programa što se rezultuje i povećanjem pouzdanosti svake naredne verzije programa.

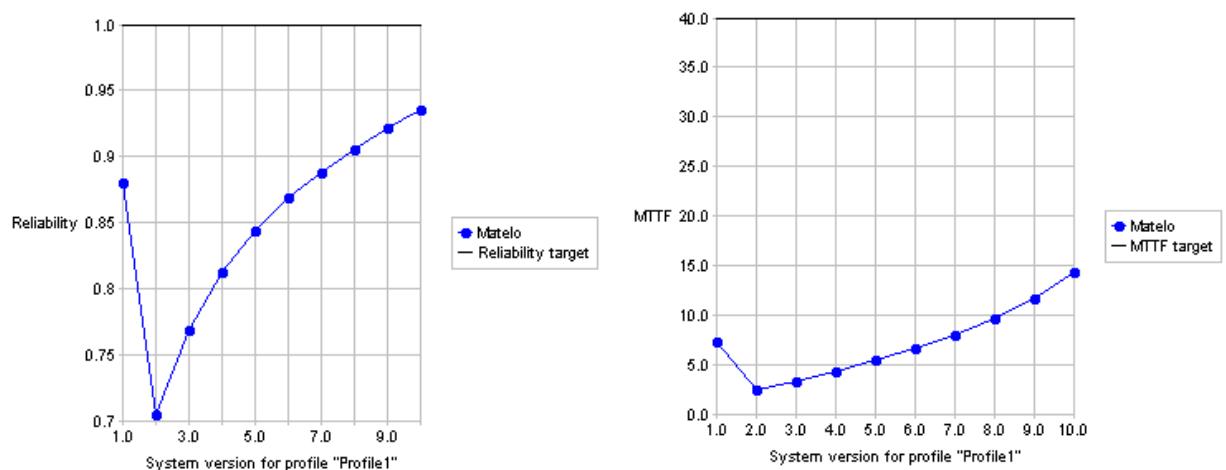
MTTF programa (desni grafik na slikama 5.2 i 5.3) se računa na osnovu dobijene pouzdanosti. Kao u slučaju grafika pouzdanosti, krive MTTF za jednu i dve aktivacije programa, dosta su slične po obliku, ali dobijene vrednosti za svaku verziju sistema su znatno različite. Naime, ako se u Tabeli 5.2 analiziraju dobijene pouzdanosti za poslednje verzije programa može se uočiti da je pouzdanost programa u slučaju jedne aktivacije veće, ali neznatno. Međutim, ukoliko se u istoj tabeli pogledaju vrednosti MTTF za jednu i dve aktivacije programa vidi se da se dobijene vrednosti MTTF za odgovarajuće verzije za dva puta razlikuju. MTTF programa u slučaju jedne aktivacije je dva puta veće nego MTTF programa u slučaju dve aktivacije. Ova

činjenica je direktno posledica uticaja broja aktivacija programa. Ukoliko u toku rada programa postoje greške, nihovo otkrivanje će se desiti pre ako se program češće koristi, odnosno ukoliko ima veću frekvenciju korišćenja.

Kao u slučaju pouzdanosti programa, na prikazanim graficima se vidi da MTTF sa porastom testiranih verzija sistema raste i teži ka željenoj vrednosti. Takođe vrednost MTTF prve verzije je izuzetak, jer je njen MTTF znatno veći od MTTF naredne verzije. Uzrok ovoga leži u različitoj vrednosti pouzdanosti konkretnе verzije programa na osnovu koje se izračunava MTTF.

### 5.2.2 Kumulativna pouzdanost i MTTF

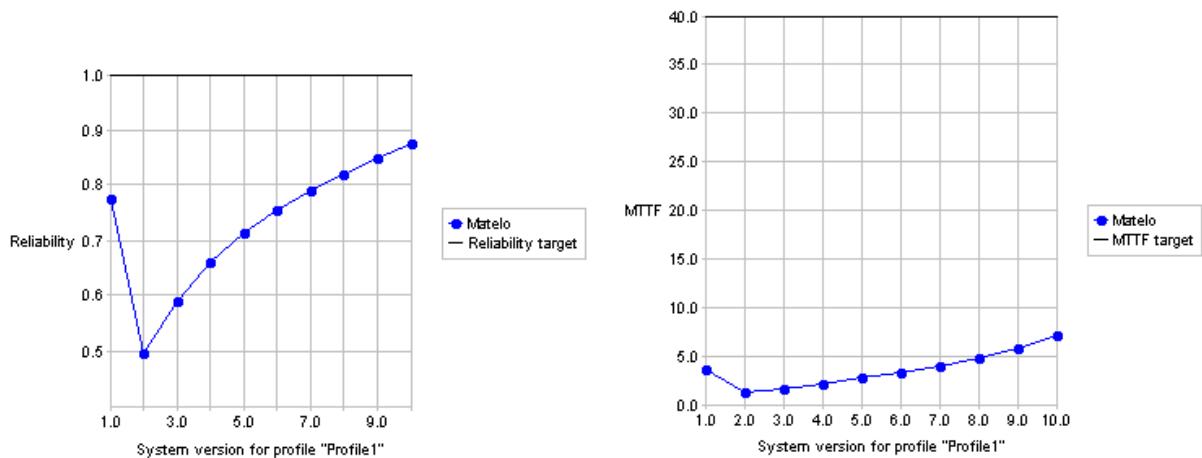
Za razliku od predhodnog odeljaka u kome su analizirani dobijeni rezultati pouzdanosti i MTTF za svaku testiranu verziju programa pojedinačno, ovde su priloženi kumulativni rezultati prethodno pomenutih karakteristika. Kumulativna pouzdanost i kumulativni MTTF predstavljaju informacije o ukupnom stanju programa i dobijene su na osnovu podataka svih ranije testiranih verzija.



Slika 5.4 – Kumulativna pouzdanost i kumulativni MTTF programa (1 aktivacija)

Na Slici 5.4 i Slici 5.5 su prikazani grafici izračunate kumulative pouzdanosti programa i kumulativnog procjenjenog srednjeg vremena do otkaza za jednu i dve aktivacije programa u toku jednog dana respektivno. Kumulativna pouzdanost sistema (levi grafik na slikama 5.4 i 5.5) predstavlja pouzdanost programa koja je izračunavana prilikom svake nove testirane verzije programa. Za razliku od pouzdanosti programa koja se računa posebno za svaku verziju kod koje se vrednost može smanjivati i povećavati u zavisnosti od testirane verzije, kod kumulativne pouzdanosti vrednost treba da raste sa svakom novom testiranom verzijom programa. Povećanje

kumulativne pouzdanosti programa je očekivana, jer je prepostavljeno da se u toku razvoja programa broj grešaka smanjuje.



Slika 5.5 – Kumulativna pouzdanost i kumulativni MTTF programa (2 aktivacije)

Na grafiku se može videti kako kumulativna pouzdanost sa povećanjem broja testiranih verzija programa raste, kao što je i očekivano. Tačne vrednosti kumulativne pouzdanosti za svaku tesiranu verziju su date u Tabeli 5.3. Trend stalnog rasta kumulativne pouzdanosti počinje od druge verzije programa bez obzira na broj aktivacija programa. Pojava naglog pada pouzdanosti između prve i druge verzije programa (za obe vrednosti aktivacija) uslovljena je Milerovim koeficijentima. Kao što je u predhodnom odeljku pomenuto, velika razlika između prepostavljenog broja grešaka koja je definisana Milerovim modelom, za prvu početnu verziju programa (jedna grešaka), i stvarnog otkrivenog broja grešaka u toku testiranja uzrok je što postoji značajna razlika između dobijene kumulativne pouzdanosti prve i druge verzije programa. Ako se pogledaju vrednosti rezultata testiranja (Tabela 5.1) i ako se njihov trend opadanja grešaka sa porastom broja verzija programa uporedi sa porastom kumulativne pouzdanosti, onda je jasno da MaTeLo alat daje objektivne procene pouzdanosti testiranog programa.

Na osnovu datih tačnih vrednosti kumulativne pouzdanosti programa za svaku verziju (Tabela 5.3) može se primetiti da pouzdanosti programa za jednu i dve aktivacije nisu toliko slične kao vrednosti pouzdanosti date u Tabeli 5.2. Ukoliko se izuzme prva vrednost, kumulativna pouzdanost se sporo menja (raste) zato što se prilikom računanja uzimaju podaci o svim testiranim verzijama programa. Ovo praktično znači da što postoji veći broj testiranih verzija programa kumulativna vrednost predstavlja relevantniju informaciju.

Verzija	Pouzdanost (1 akt.)	MTTF (1 akt.)	Pouzdanost (2 akt.)	MTTF (2 akt.)
1	0.880088	7.339419	0.774554	3.669709
2	0.704136	2.379926	0.495807	1.189963
3	0.767946	3.309337	0.589741	1.654668
4	0.812362	4.329406	0.659932	2.164703
5	0.844107	5.414659	0.712517	2.707329
6	0.868441	6.601139	0.754189	3.300570
7	0.888130	7.938961	0.788775	3.969481
8	0.905690	9.603277	0.820274	4.801638
9	0.920982	11.655383	0.848208	5.827691
10	0.934785	14.333913	0.873823	7.166956

Tabela 5.3 – Kumulativna pouzdanost i kumulativni MTTF za 1 i 2 aktivacije

Kumulativni MTTF programa (desni grafik na slikama 5.2 i 5.3) se određuje na osnovu procenjene kumulativne pouzdanosti programa. Slično kao i kod kumulativne pouzdanosti, grafici kumulativnog MTTF imaju sličan oblik, ali dobijene vrednosti se značajno razlikuju. Kumulativni MTTF u slučaju jedne aktivacije programa je duplo veći nego kumulativni MTTF u slučaju dve aktivacije programa. Ovo je posledica duplo većeg broja aktivacija programa, odnosno duplo veće frekvencije korišćenja programa. Ako se kreće od prepostavke da ne postoji idealan program bez grešaka, logično je da će program koji se duplo više koristi duplo brže otkriti grešku u radu, odnosno da će imati manji MTTF.

Bitno je i naglasiti da su vrednosti kumulativnog MTTF manje od vrednosti MTTF koji su dobijene za svaku verziju posebno (predhodni odeljak). Uzrok toga je sličan kao i kod kumulativne pouzdanosti. Naime, u predhodnom poglavljju sa svaku verziju programa prilikom određivanja MTTF su korišćenje samo podaci dobijeni testiranjem te konkretnе verzije programa što je moglo znatno da uveća ili da smanji dobijeni MTTF. Kod kumulativnog MTTF prilikom računanja se uzimaju rezultati svih testiranih verzija programa, odnosno istorija razvoja programa. Zbog toga se vrednost kumulativnog MTTF sporo menja (raste). Bez obrzira na relativno male promene, grafik kumulativnog MTTF za obe aktivacije ima trend bržeg ili sporijeg rasta ka željenoj vrednosti.

## 6. Zaključak

U ovom radu opisana je metodologija statističkog testiranja i jedna njena primena na zadatom programu na osnovu zadatog operativnog profila upotrebom programske alate MaTeLo. Na konkretnom primeru je pokazan način funkcionisanja i mogućnosti alata MaTeLo kao jednog od trenutno najperspektivnijih alata za statističko testiranje.

Metodologija je primenjena na zadatom programu uz priloženi model korišćenja programa. Upotrebom priloženog modela korišćenja opisan je proces automatiskog stvaranja apstraktnih test slučajeva i način njihovog prevođenja u izvršive test slučajeve prilagođene za izvršavanje na JUnit platformi. Na osnovu dobijenih test slučajeva pomoću JUnit platforme izvršeno je testiranje sistema i prikupljeni su rezultati testiranja programa. Dobijeni rezultati su automatski uneti u alat MaTeLo kako bi se mogla izvršiti analiza dobijenih rezultata. Takođe, razvijen je poseban sintaksni analizator koji automatizuje proces unosa podataka u cilju izbegavanja ručnog unosa. Na osnovu statističke analize rezultata dobijene su procene karakteristika testiranog programa. Kao najbitnije karakteristike analizirane su procene pouzdanost i srednje vreme do otkaza programa.

Posebno razvijeni sintaksni analizator se koristi kao dodatak skupu alata MaTeLo i on predstavlja alat koji nedostaje kako bi se proces od stvaranja test slučajeva za testiranje sistema do dobijanja procena njegovih karakteristika u potpunosti automatizovao. Naime, on automatizuje proces ažuriranja dobijenih rezultata test slučajeva čime se smanjuje vreme koje je potrebno za taj posao ukoliko bi se on radio ručno, kao i verovatnoća pojave greške u toku unosa, što je ujedno i njegova najveća prednost.

Manu sintaksnog analizatora predstavlja velika zavisnost od strukture datoteke u kojoj se smeštaju rezultati dobijeni u toku testiranja koju on koristi kao izvor podataka. U zavisnosti od platforme na kojoj se sistem testira dobijeni podaci o izvršavanju test slučajeva mogu biti

različito organizovani. Rezultati se mogu organizovati u različitim oblicima tekstualnih datoteka ili kao baze podataka, što znatno komplikuje pristup podacima.

U korišćenoj verziji alata MaTeLo uočeno je ograničenje koje se odnosi na broj test slučajeva koji se mogu stvoriti u jednoj testnoj zbirci. Naime, ako se koristi Korisnički orijentisana strategija testiranja zbirka može imati najviše četrsto testnih slučajeva što predstavlja ozbiljno ograničenje alata.

U narednim verzijama sintaksnog analizatora trebalo bi da bude omogućen pristup podacima koji se nalaze u bazi podataka kao i pristup podacima koji su uređeni po standardizovanim šemama koje su u širokoj upotrebi.

## 7. Literatura

- [1] T. Thelin, "Automated Statistical Testing Suite for Software Validation", Dep. Of Communication Systems, Lund University.
- [2] All4tech, "Generating Transition Probabilities for Automatic Model-Based Test Generation", Paris, France
- [3] A. Guiotto, B. Acquaroli, A. Martelli, "MaTeLo: Automated Testing Suite for Software Validation", Proceedings of DASIA 2003 Jun. 2003, Prague, Czech Republic.
- [4] K. Sayre, "Improved Techniques for Software Testing Based on Markov Chain Usage Models", PhD thesis, University of Tennessee, Knoxville, December 1999.
- [5] W. Dulz, Z. Fenhua, "MaTeLo – Statistical Testing Using Annotated Sequence Diagrams, Markov Chains and TTCN-3", IEEE International Conference on Quality Software, Nov. 2003.
- [6] W. Dulz, "MaTeLo - Statistical Testing Using Annotated Sequence Diagrams, Markov Chains and TTCN-3", Technical report 05/02, Department of Computer Science 7 (Computer Networks & Communication Systems), University of Erlangen.
- [7] A. Bettinotti, M. Garavaglia, "Test automation for Markov Chain Usage Models", Grupo de Desarrollo en Sistemas, Facultad Regional Delta, Universidad Tecnológica Nacional, Campana, Argentina
- [8] D. Vujović, "Generalizacija i automatizacija sistema za funkcionalno testiranje STB", MSc, Novi Sad, decembar 2011
- [9] D. Marijan, "Razvoj metodologije testiranja softvera u multimedijaalnim sistemima", Novi Sad, 2011