



# 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: Стефан Стојановић

Број индекса: RA16/2013

Тема рада: Једно решење алата за измену извештаја у процесу интеграције AUTOSAR софтвера

Ментор рада: доц. др Богдан Павковић

Нови Сад, октобар, 2020.



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
21000 НОВИ САД, Трг Доситеја Обрадовића 6

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, <b>РБР</b> :	
Идентификациони број, <b>ИБР</b> :	
Тип документације, <b>ТД</b> :	Монографска документација
Тип записа, <b>ТЗ</b> :	Текстуални штампани материјал
Врста рада, <b>ВР</b> :	Завршни (Bachelor) рад
Аутор, <b>АУ</b> :	Стефан Стојановић
Ментор, <b>МН</b> :	доц. др Богдан Павковић
Наслов рада, <b>НР</b> :	Једно решење алата за измену извештаја у процесу интеграције AUTOSAR софтвера
Језик публикације, <b>ЈП</b> :	Српски / латиница
Језик извода, <b>ЈИ</b> :	Српски
Земља публикавања, <b>ЗП</b> :	Република Србија
Уже географско подручје, <b>УГП</b> :	Војводина
Година, <b>ГО</b> :	2020.
Издавач, <b>ИЗ</b> :	Ауторски репринт
Место и адреса, <b>МА</b> :	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО</b> : (поглавља/страница/ цитата/табела/слика/графика/прилога)	7/30/0/1/15/0/0
Научна област, <b>НО</b> :	Електротехника и рачунарство
Научна дисциплина, <b>НД</b> :	Рачунарска техника
Предметна одредница/Кључне речи, <b>ПО</b> :	AUTOSAR, XML, Python, алат, измена, извештај, тест, интеграција
УДК	
Чува се, <b>ЧУ</b> :	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, <b>ВН</b> :	
Извод, <b>ИЗ</b> :	У овом раду описана је имплементација Python апликације за измењивање AATR докумената. Апликација представља графичко окружење које кориснику нуди све функционалности потребне за успешно учитавање, измену и чување података унутар докумената XML формата.
Датум прихватања теме, <b>ДП</b> :	
Датум одбране, <b>ДО</b> :	16.10.2020.
Чланови комисије, <b>КО</b> :	Председник: ванредни професор Иван Каштелан
	Члан: доцент Марија Антић
	Члан, ментор: доцент Богдан Павковић
	Потпис ментора



## 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> :	Stefan Stojanović
Mentor, <b>MN</b> :	Bogdan Pavković, PhD
Title, <b>TI</b> :	One solution for the report change tool in the process of integrating AUTOSAR software
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> :	2020.
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/appendixes)	7/30/0/1/15/0/0
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> :	AUTOSAR, XML, Python, editor, report, test, integration
<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> :	This paper describes the implementation of a Python application used to edit AUTOSAR AATR files. The application is developed in the form of a GUI that provides all the tools and options necessary for loading, editing and saving XML files.
Accepted by the Scientific Board on, <b>ASB</b> :	
Defended on, <b>DE</b> :	16.10.2020.
Defended Board, <b>DB</b> :	President: associated professor Ivan Kaštelan, PhD
	Member: assistant professor Marija Antić, PhD
	Member, Mentor: assistant professor Bogdan Pavković, PhD
	Menthor's sign

## **Zahvalnost**

Zahvaljujem se mentoru, doc. Dr Bogdanu Pavkoviću i Čedomiru Jovanoviću na pruženoj stručnoj pomoći i savetovanju tokom izrade ovog rada.

Posebnu zahvalnost dugujem svojoj porodici, čija je neizmerna podrška bila od presudnog značaja za celo moje školovanje.



# УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



## SADRŽAJ

1. Uvod .....	1
2. Teorijske osnove .....	2
2.1 AUTOSAR .....	2
2.2 AUTOSAR arhitektura.....	4
2.3 Testiranje softvera.....	7
2.4 Proces testiranja integrisanih aplikacija unutar AUTOSAR softvera.....	8
2.5 XML.....	10
3. Koncept rešenja .....	12
4. Programsko rešenje .....	15
4.1 Korišćeni alati i okruženje.....	15
4.2 Struktura projekta.....	18
4.2.1 Modul <i>xml_editor.py</i> .....	19
4.2.2 Modul <i>dialogs.py</i> .....	20
4.2.3 Modul <i>pdf_exporter</i> .....	21
4.2.4 Modul <i>create_issue_list.py</i> .....	23
4.2.5 Modul <i>classes.py</i> .....	23
4.3 Tok programa .....	24
5. Rezultati .....	27
6. Zaključak .....	30
7. Literatura.....	31

## SPISAK SLIKA

Slika 2.1: AUTOSAR partneri .....	2
Slika 2.2: Jedan od osnovnih ciljeva AUTOSAR standarda .....	3
Slika 2.3: AUTOSAR arhitektura .....	4
Slika 2.4: Arhitektura AUTOSAR Classic platforme .....	5
Slika 2.5: Softverska komponenta.....	6
Slika 2.6: Izgled XML elementa .....	11
Slika 4.1: Izgled alata <i>wxFormBuilder</i> .....	16
Slika 4.2: <i>jinja2</i> šablon za čuvanje podataka .....	17
Slika 4.3: Organizacija <i>Python</i> modula koda projekta .....	18
Slika 4.4: Namenski prozor za podešavanje podrazumevanih putanja .....	20
Slika 4.5: Namenski prozor za prikaz rezultata generisanja PDF dokumenata .....	20
Slika 4.6: Jedna od stranica generisanog PDF dokumenta .....	22
Slika 4.7: Primer generisane liste izveštaja .....	23
Slika 5.1: Izgled finalne aplikacije tokom korišćenja .....	27
Slika 5.2: Rezultat poređenja izlaznih datoteka prethodno korišćene aplikacije i završne aplikacije projekta .....	28

## **SPISAK TABELA**

Tabela 4.1: Metode za dodavanje novih čvorova u stablo podataka.....	25
--	----

## SKRAĆENICE

**AAT** – *Application Acceptance Test* – Test validnosti aplikacije

**AATR** – *Application Acceptance Test Report* – Rezultat testa validnosti aplikacije

**AIT** – *Application Integration Test* – Test integracije aplikacije

**API** – *Application Programming Interface* – Programsko sučelje aplikacije

**AUTOSAR** – *Automotive Open System Architecture* – Standard za razvoj softvera u automobilske industriji

**ECU** – *Electronic Control Unit* – Elektronska upravljačka jedinica

**GUI** – *Graphical User Interface* – Grafička korisnička sprega

**RTE** – *Runtime Environment* – Izvršno okruženje

**SIT** – *Software Integration Test* – Test integracije softvera

**SWC** – *Software Component* – Softverska komponenta

**W3C** – *World Wide Web Consortium* – Međunarodna organizacija za Internet standarde

**XML** – *Extensible Markup Language* – Jezik za označavanje dokumenata



## 1. Uvod

Od nastanka prvog automobila pa do danas, ljudske potrebe transporta se gotovo neprestano proširuju i postaju sve složenije. Zajedno sa sve boljom optimizacijom postojećih rešenja, te potrebe uslovljavale su konstantan napredak automobila kao prevoznog sredstva. Konstantan i sve brži razvoj automobilske industrije doveo je do toga da je današnji automobil daleko složeniji sistem od automobila od pre samo nekoliko decenija. Konstantna nova otkrića i usavršavanja iz oblasti mehanike, bezbednosti u vožnji, energetske efikasnosti, aerodinamike, elektronike i dr. omogućavaju proizvodnju sve kvalitetnijih prevoznih sredstava, kao i dodavanje raznih novih komponenti u njihov postojeći sklop. Dok je u prošlosti broj elektronskih komponenti u automobilu bio zanemarljivo mali, danas gotovo svaka komponenta vozila ima neku elektronsku karakteristiku. Bilo da je reč o senzorima za pomoć pri parkiranju, kamerama za veću vidljivost sa svih strana vozila, ili upozorenju o nevezanom sigurnosnom pojasu, elektronika je sve prisutnija u svakodnevnom korišćenju modernog automobila.

Kako su broj i kompleksnost računarskih sistema u automobilima vremenom vrtoglavo rasli, zajedno sa njima su se usložnjavale potrebe za većom standardizacijom i organizacijom, kako bi se unapredili bezbednost, kvalitet i efikasnost proizvodnje. Jedno od najboljih rešenja do sada ponudila je organizacija AUTOSAR (eng. *Automotive Open System Architecture*), koja je razvila istoimeni standard za razvoj, integraciju i testiranje softvera u automobilske industriji. Kao jedan od vodećih standarda, AUTOSAR je umnogome uprostio, ubrzao i poboljšao proizvodnju računarskih komponenti za vozila.

U ovom radu opisan je proces razvoja i testiranja grafičkog okruženja Python aplikacije. Glavni cilj aplikacije jeste da korisniku pruži sve opcije neophodne za učitavanje, izmenu, i čuvanje AATR XML dokumenata.

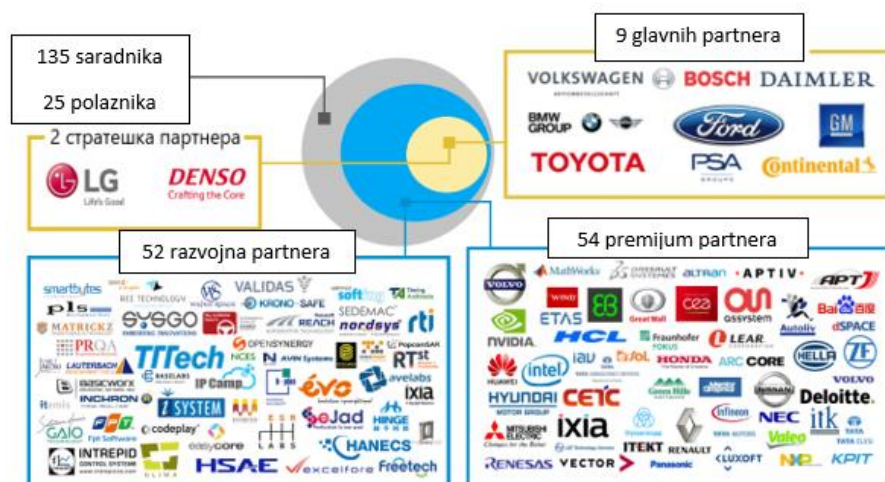
Rad se sastoji iz sedam poglavlja. Osnovni principi samog AUTOSAR standarda, kao i teorijske osnove bitne za ovaj rad, opisani su u narednom poglavlju. U trećem poglavlju dat je opis ideje i koncept funkcionisanja rešenja zadanog problema. Četvrto poglavlje se bavi strukturom i tokom samog programskog koda rešenja, kao i opisom alata korišćenih tokom razvoja. Naredna dva poglavlja posvećena su rezultatima testiranja, rezimiranju ostvarenih mogućnosti koje gotova aplikacija nudi, kao i mogućim proširenjima i poboljšanjima ostvarenog rešenja. U poslednjem poglavlju je navedena literatura korišćena tokom izrade rada.

## 2. Teorijske osnove

U ovom poglavlju dat je kratak opis teorijskih osnova na kojima se ovaj rad bazira. Predstavljen je AUTOSAR sistem, zajedno sa svojom arhitekturom, delovima od kojih se sastoji, kao i principima na osnovu kojih ceo sistem funkcioniše. Pored toga, opisana je procedura testiranja softvera, kao i pojam XML datoteka.

### 2.1 AUTOSAR

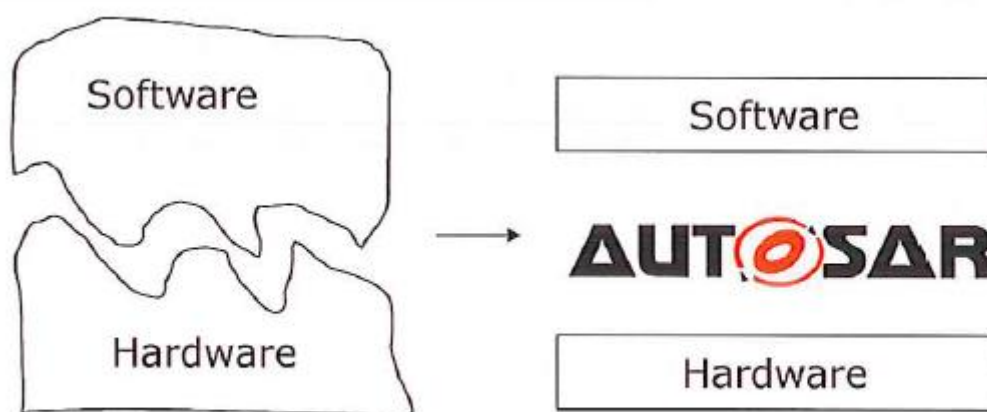
AUTOSAR (eng. *Automotive Open System Architecture*) je globalno partnerstvo proizvođača i dobavljača automobila i njihovih delova, kao i proizvođača poluprovodnika i alata.



Slika 2.1: AUTOSAR partneri

Ovo partnerstvo su 2003. godine osnovale kompanije kao što su BMW, Volkswagen, Daimler, Bosch i Continental, dok su im se vremenom priključile i mnoge druge kompanije (Ford Motor Company, General Motors, i drugi).

Osnovna ideja i cilj AUTOSAR partnerstva bila je standardizacija razvoja elektronskih komponenti i aplikacija korišćenih unutar automobila. Glavni razlog postojanja ovog cilja je veoma visok nivo složenosti modernog automobilskeg sistema. Izuzetno veliki broj funkcija predstavljao je potencijalan problem pri upotrebi komponenti drugih proizvođača ili dodavanju novih funkcionalnosti. Uska povezanost hardvera i softvera je u prošlosti otežavala implementaciju novih i ažuriranje postojećih funkcionalnosti. Zbog toga AUTOSAR standard uvodi skup specifikacija i standardizovanih rešenja za celu automobilsku intustriju, dok se konkurencija između kompanija održava na nivou implementacije istih. Time je postignuta apstrakcija hardvera i softvera potrebnih za razvoj i implementaciju softverskih komponenti. Još jedan pozitivan rezultat AUTOSAR standarda jeste modularnost komponenti i mogućnost njihovog ponovnog korišćenja u budućim projektima, nezavisno od platforme. Jedan od slogana AUTOSAR-a je „Saradnja na standardizaciji – konkurencija u implementaciji”.



Slika 2.2: Jedan od osnovnih ciljeva AUTOSAR standarda

## 2.2 AUTOSAR arhitektura

AUTOSAR standard karakteriše arhitektura koja se sastoji iz tri osnovna sloja:

1. Osnovni softver (eng. *Basic Software - BSW*)
2. Izvršno okruženje (eng. *Realtime Environment - RTE*)
3. Aplikativni sloj (eng. *Application layer*)

Standardizacija sprega između aplikacija i osnovnog softvera omogućava apstrakciju aplikacije od softvera i jednostavniju integraciju funkcionalnih modula. To se postiže postojanjem konfigurabilnog posredničkog softvera (eng. *middleware*). Njegov naziv je RTE (eng. *Runtime Environment*), i on je zadužen za komunikaciju između osnovnog softvera i aplikativnog sloja.

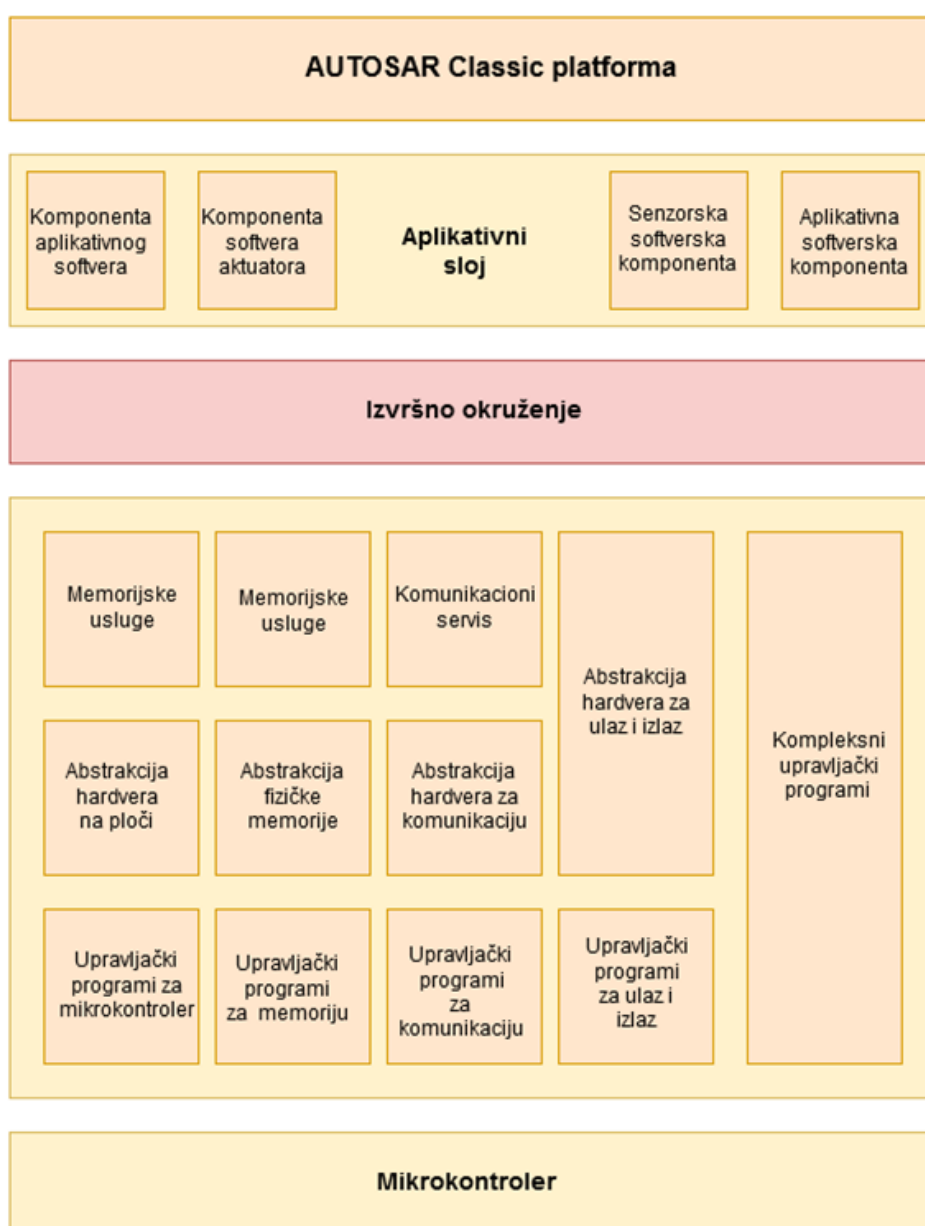


Slika 2.3: AUTOSAR arhitektura

Osnovni softver je sloj standardizovanog softvera čija je glavna svrha pružanje pristupa resursima potrebnim za funkcionisanje viših slojeva. Sastoji se iz mnoštva softverskih modula koji opisuju određena ponašanja i funkcionalnosti elektronskih upravljačkih jedinica (eng. *Electronic Control Unit - ECU*). Osnovni softver se dalje deli na: uslužni sloj, apstrakciju elektronske upravljačke jedinice, kompleksne rukovaoce i apstrakciju mikrokontrolera.

Izvršno okruženje ima formu virtuelne magistrale i služi kao posrednički sloj između osnovnog softvera i aplikativnog sloja. Osnovna uloga RTE sloja je da upravlja razmenom podataka između softverskih komponenti, njihovim raspoređivanjem (eng. *scheduling*), kao i komunikacijom između osnovnog softvera i aplikacija. Na taj način RTE sloj predstavlja komunikacionu spregu između ostalih delova AUTOSAR arhitekture, i pruža softverskim komponentama međusobnu nezavisnost. RTE sprega se generiše posebno za svaku ECU jedinicu.

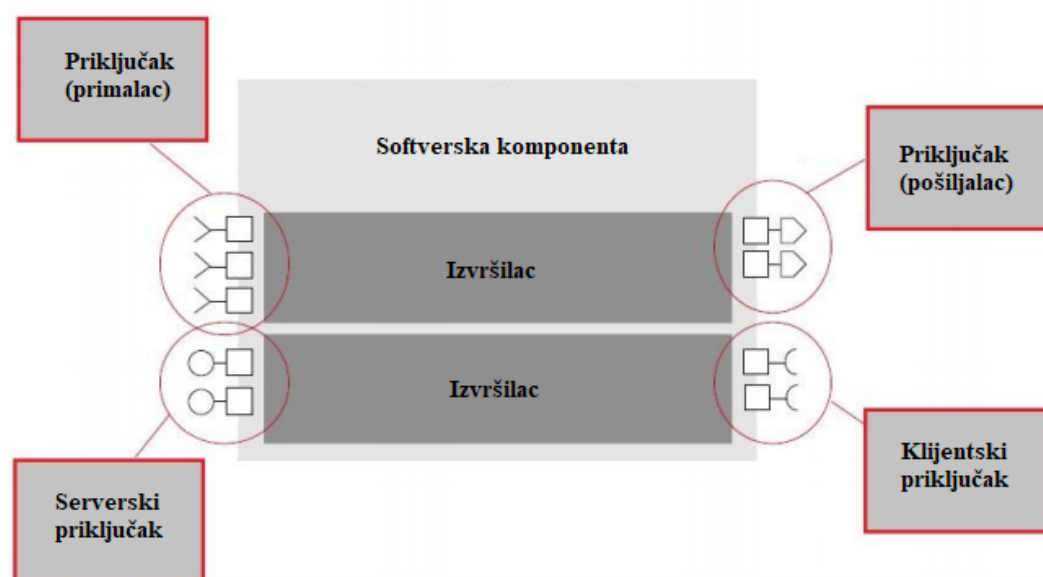
Ovakva troslojna arhitektura je jedna od osnovnih osobina AUTOSAR standarda i predstavlja AUTOSAR Classic platformu.



Slika 2.4: Arhitektura AUTOSAR Classic platforme

Aplikativni sloj čine softverske komponente (eng. *Software Component - SWC*). One predstavljaju manje aplikacije koje imaju tačno definisanu određenu funkcionalnost. Kako se sva komunikacija softverskih komponenti i ostatka sistema dešava preko izvršnog okruženja, komponente su nezavisne od ECU jedinice na kojoj se nalaze, i međusobno se povezuju strogo definisanim priključcima (eng. *Port*). Jedna komponenta može istovremeno da ima više priključaka i da preko njih bude povezana sa više komponenti.

Za pojam softverskih komponenti je usko povezan i pojam izvršioca (eng. *Runnable*). Izvršilac je deo koda koji predstavlja implementaciju softverske komponente kojoj pripada, i koji može biti pokrenut na više različitih načina (periodično, po prijemu ili slanju podataka, prilikom greške u slanju, itd.). Jedna softverska komponenta može imati više izvršilaca, dok svaki izvršilac može imati samo jedan okidač izvršavanja. U AUTOSAR aplikativnom sloju razlikujemo više tipova softverskih komponenti, kao što su atomične, senzorske i aktuatorske, i kompozitne softverske komponente, koje se zapravo sastoje iz komponenti čije funkcionalnosti čine jednu logičku celinu.



Slika 2.5: Softverska komponenta

## 2.3 Testiranje softvera

Testiranje softvera je provera poklapanja načina funkcionisanja posmatranog programa sa očekivanim, kao i verifikacija da on ispunjava sve zahteve postavljene pre samog njegovog razvoja. Cilj testiranja jeste izvršavanje programa sa namerom nalaženja greške. Dobar test je onaj koji ima veću verovatnoću nalaženja neotkrivene greške. Zbog prirode razvoja softvera (ograničeno vreme, veliki obim projekata, i slično), nemoguće je testirati svaki mogući slučaj, te je potrebno odrediti konkretne testne slučajeve sa jasno određenim ciljevima. Da bi se izbeglo nepotrebno ponavljanje testiranja već proverenih osobina posmatranog softvera, potrebno je redovno ažuriranje postojećih testnih slučajeva. Takođe, pisanje novih testnih slučajeva je od presudne važnosti, kako bi se pokrilo što više različitih delova posmatranog softvera. Uz to, moraju se definisati tehnike i alati koji će se koristiti, i jasno podeliti uloge među zaduženim timovima. Sam process testiranja sastoji se iz više različitih faza:

- Planiranje i kontrola
- Analiza i projektovanje
- Implementacija i izvršavanje
- Procena i beleženje rezultata
- Završni postupci testiranja

Testiranje može vršiti sam programer, koji poznaje sistem i motivisan je krajnjim rezultatom projekta, ili nezavisni tester, koji mora da se upozna sa datim sistemom, ali je motivisan postizanjem većeg kvaliteta istog. Postoji više različitih tipova testiranja, kao što su test pristupačnosti, test pozadinskih mehanizama, test kompatibilnosti sa prethodnim verzijama, test usklađenosti, i drugo. Takođe postoje različiti nivoi testiranja (testiranje komponente, integraciono testiranje, testiranje sistema, testiranje prihvatljivosti), kao i različite metode testiranja (testiranje metodom crne kutije, testiranje metodom bele kutije, itd).



## 2.4 Proces testiranja integrisanih aplikacija unutar AUTOSAR softvera

Proces testiranja integrisanih AUTOSAR aplikacija definiše sve postupke potrebne za razvoj ECU sistema. Podeljen je na tri potprocesa: AAT (eng. *Application Acceptance Test*), AIT (eng. *Application Integration Test*) i SIT (eng. *Software Integration Test*).

Cilj testa prihvatljivosti aplikacije (AAT) je da proveriti da li su ispoštovani svi kriterijumi prihvatanja SW-C komponente, kako bi se mogla izvršiti njena uspešna integracija u ECU jedinicu. U te svrhe, snabdevač testira softversku komponentu na ECU platformi i beleži ulazne i izlazne podatke - vektore. Tada se vrši isporuka vektora i testne dokumentacije (AIT\_SW-C) integratoru. Pored AIT\_SW-C dokumenta, potrebno je da snabdevač dostavi i ostale potrebne dokumente i datoteke, sa odgovarajućom strukturom i nazivnim konvencijama (XML dokumenti, prethodno prevedene biblioteke, testni vektori i ostali podaci, u svojim respektivnim direktorijumima). Podaci rezultata izvršenog AAT testa nalaze se u posebnom dokumentu XML formata (eng. *Application Acceptance Test Report*), koji je od posebnog značaja za temu ovog rada.

Drugi potproces je test integracije aplikacije (AIT). Njegov zadatak je da verifikuje očekivano ponašanje komponente nakon njene integracije u ECU jedinicu. AIT izvršavaju i snabdevač i integrator za svaku komponentu ponaosob, tako što se na namensku platformu spušta samo jedna aplikacija, dok se ostale zaustavljaju. Snabdevač vrši proveru ponašanja komponente, beleži ulazne i izlazne vektore, i šalje ih integratoru, zajedno sa vektorom dozvoljenog odstupanja. Integrator tada vrši identične provere koristeći primljene ulazne, i beleži izlazne vektore. Na kraju, integrator poredi svoje izlazne vektore sa onima koje je dobio od snabdevača, i ako je odstupanje manje od dozvoljenog, rezultat testa je pozitivan.

Poslednji deo procesa testiranja integrisanih AUTOSAR aplikacija jeste test same integracije softvera (SIT), čija uloga u procesu podrazumeva utvrđivanje tačnosti integracije kompletnog softvera, koji se sastoji iz platformskog softvera, upravljačkog softvera i svih softverskih komponenti aplikacije. Druga uloga SIT-a jeste potvrda da ceo softver može da funkcioniše bez smetnji koje mogu biti prouzrokovane razlikama u korišćenom hardveru ili menjanjem konfiguracija u zavisnosti od tipa ciljnog vozila. Da bi ovaj test mogao da se izvrši, uz druge specifičnosti potrebni su i pozitivni rezultati prethodno izvršenih AAT i AIT testova, kao i pokretanje celog procesa izgradnje bez grešaka.

Ako su svi preduslovi ispunjeni, SIT se izvršava na kompletnom softveru, nakon spuštanja svih komponenti zajedno na namensku platformu, uključujući sve nezavisne aplikacije i upravljački softver. Rezultat testa je pozitivan ako su ispunjena sledeća tri uslova:

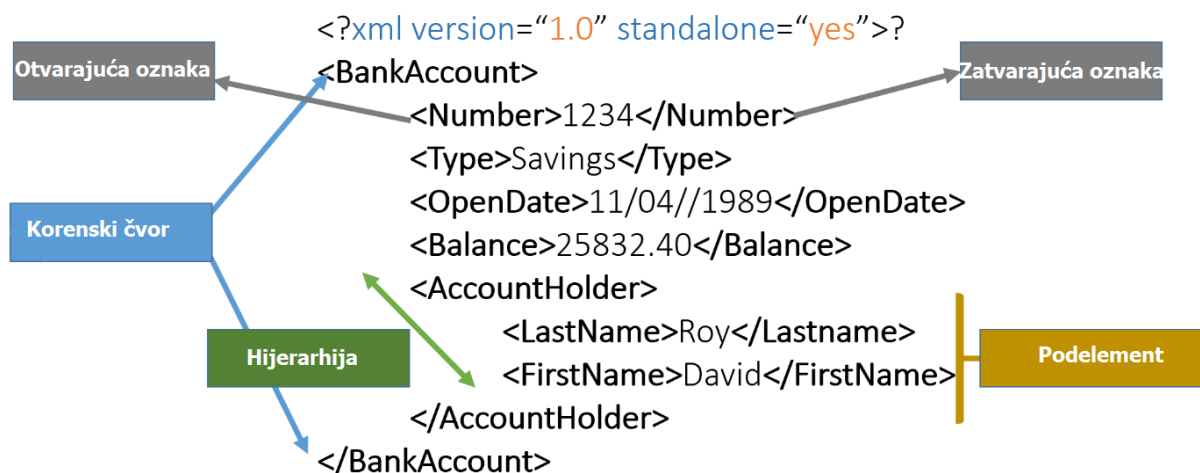
- Za svaku softversku komponentu integrisana je tačna verzija
- Za svaku softversku komponentu *init* metoda se poziva tačno jednom
- *main* metoda svake softverske komponente se poziva ciklično, u periodama definisanim u modelu.

## 2.5 XML

U računarstvu, pojam označavanja dokumenta predstavlja proces upotrebe kodova kako bi se definisala struktura, vizuelna predstava, kao i značenje podataka unutar njega. XML (extensible markup language) je jezik za predstavljanje, označavanje, i prenos podataka unutar tekstualnih dokumenata, kroz različite informacione sisteme. Propisan od strane međunarodne organizacije za izradu internet standarda (W3C) 1998. godine, XML standard u osnovi koncepta korišćenja sadrži prostu ideju jednostavnosti korišćenja i čitljivosti, kako za korisnike tako i za računare.

XML dokument je samoopisujuća, platformski nezavisna tekstualna datoteka. Osnovni blokovi strukture XML dokumenta se zovu elementi. Jedan XML element (Slika 2.6) predstavlja podatke (*text*), koji se nalaze između početne, otvarajuće, i krajnje, zatvarajuće etikete (*tag*). Početna i krajnja etiketa jednog elementa predstavljaju njegov naslov, i moraju imati identičan naziv, s tim da krajnja etiketa kao prvi karakter sadrži znak "/". Elementu također može biti pridružen proizvoljan broj atributa, koji pružaju dodatne informacije o njemu, i sastoje se iz naziva atributa, znaka jednakosti i vrednosti atributa. Sve etikete, podaci i atributi su tekstualnog tipa, što omogućava proširivost XML jezika, i čitanje XML datoteka na različitim platformama koje imaju mogućnost čitanja tekstualne datoteke. Nazivi XML elemenata unutar etiketa moraju postojati, sadržaj elementa može biti prazan znakovni niz (string), dok su atributi u potpunosti opcioni. Sva polja unutar XML dokumenta su case sensitive. Struktura samog XML dokumenta je hijerarhijska, što znači da svaki element može sadržati jedan ili više 'podelemenata'. Dokument ima jedinstveni element na početku hijerarhije, koji predstavlja korenski element stabla. Svi ostali elementi su 'deca' korenskog elementa. Dobro oblikovana (eng. *well-formed*) XML datoteka poštuje skup veoma strogih pravila koja regulišu XML jezik. Ova pravila su sintaksne prirode, i odnose se na postojanje otvarajućih i zatvarajućih etiketa, karaktera dozvoljenih u naslovima i podacima elemenata, ugnježđenost elemenata, osetljivost na mala i velika slova, i slično. Ako je bilo koje od tih pravila prekršeno, datoteka ne može biti otvorena ni obrađivana kao XML datoteka. Za proveru oblika XML datoteke koristi se jedan od dva standarda preporučenih od strane W3C: DTD (eng. *Document Type Definition*) ili XSD (eng. *XML Schema Definition*). Oba standarda funkcionišu tako što upoređuju strukturu date XML datoteke sa prethodno definisanom šemom, koja sadrži informacije o potrebnoj hijerarhiji, tipovima i broju elemenata i njihovih atributa, a koja se nalazi u posebnoj datoteci.

Ako je posmatrana XML datoteka dobro oblikovana i konzistentna sa željenom šemom, tada je ona ispravna (eng. *valid*), dok je, ukoliko ne zadovoljava šemu, nevalidna (eng. *invalid*).



Slika 2.6: Izgled XML elementa

Za AUTOSAR standard veoma je bitno postojanje ARXML (AUTOSAR XML) datoteke. To je datoteka XML formata u kojoj se nalaze specifikacije posmatranih elektronskih upravljačkih jedinica. Pored osnovnih XML pravila, ARXML datoteka mora da ispoštuje i skup dodatnih pravila serijalizacije postavljenih AUTOSAR standardom, kako bi željeni model bio što vernije predstavljen. Između ostalog, ova pravila se tiču kodiranja, formatiranja i indentacije dokumenta, upotrebe imenskih prostora i sortiranja elemenata, i u nastavku su navedena samo neka od mnogih:

- Raščlanjivanje datoteka – Jedan AUTOSAR model može biti isporučen u vidu više ARXML opisnih datoteka
- Format datoteka – Sve ARXML datoteke moraju imati ekstenziju *.arxml*
- Dužina naziva datoteka – Najveća dozvoljena dužina naziva ARXML datoteka je 255 karaktera
- Kodiranje – Jedino dozvoljeno kodiranje karaktera je *UTF-8*.
- Verzija – Sve datoteke moraju biti u skladu sa XML verzijom 1.0
- Na početku svake opisujuće datoteke mora postojati deklaracija *UTF-8* kodiranja i XML verzije
- Opisujuće datoteke mogu sadržati komentare

Poštovanjem ovih pravila ARXML umanjuje broj mogućih varijacija XML datoteka, znatno olakšavajući implementaciju potrebnih alata, kao i razmenu podataka između različitih AUTOSAR partnera.

### 3. Koncept rešenja

U ovom poglavlju dat je detaljniji opis teme rada, kao i opis njegove realizacije. Potrebno je napraviti grafičku korisničku spregu, koja omogućava korisniku učitavanje postojećih ili kreiranje novih AAT rezultata u XML formatu, vršenje odgovarajućih izmena i čuvanje izmenjene datoteke na hard disku. Struktura trenutno otvorene XML datoteke prikazana je korisniku u vidu stabla. Takođe je potrebno realizovati opcije za dodavanje novih i uklanjanje postojećih elemenata stabla, čuvanje unesenih podataka u PDF formatu, kao i podešavanje podrazumevanih vrednosti nekih od parametara programa.

Izgled i uređenje aplikacije su zamišljeni sa jednostavnošću i jasnoćom korišćenja kao glavnim prioritetima. Glavni prozor aplikacije se sastoji iz tri panela, od kojih svaki ima svoju specifičnu namenu. Levi panel sadrži grafički element za prikaz i interakciju sa stablom čvorova XML dokumenta, kao i polje za pretragu testnih slučajeva, koji će redom selektovati sve testne slučajeve koji u svom nazivu sadrže uneseni tekst. U desnom panelu se nalaze tekstualna i polja sa izborom, koja sadrže podatke podčvorova selektovanog čvora. Prikaz podataka u desnom panelu se menja u zavisnosti od selekcije čvora stabla na levom panelu. Dubina prikaza je 1, što znači da će biti prikazani samo podčvorovi direktno ispod selektovanog čvora, dok će se umesto onih dubljih videti samo naziv njihovog respektivnog matičnog čvora. Panel sa donje strane prozora sadrži prikaz detaljnijeg opisa trenutno selektovanog čvora, ukoliko opis postoji. Paneli se nalaze unutar kliznih prozora, što znači da svaki od njih ima podesivu širinu, što može povećati preglednost kod stabala čiji čvorovi imaju veoma duge nazive. Takođe postoje i traka sa alatima i meni traka, koje se nalaze pri vrhu glavnog prozora, iznad levog i desnog panela, a ispod trake sa nazivom aplikacije, kao i statusna traka na samom njegovom dnu.

Uloga statusne trake je da prikaže korisniku informacije o raznim elementima prozora aplikacije (ukoliko one postoje) pri prelasku mišem preko njih, i da ga obavesti o uspešnosti zatraženih operacija (kao što su otvaranje i čuvanje datoteke, dodavanje i brisanje čvorova, itd.). Meni traka se sastoji iz *File*, *Edit*, *Options*, i *Help* menija, koji u sebi sadrže podmenije sa listama operacija koje su dostupne korisniku. Svaka operacija ima svoju prečicu u vidu komande sa tastature. Komanda se izvršava istovremenim pritiskom tastera *Ctrl* i tastera odgovarajućeg slova.

Opcije sa trake u gornjem delu glavnog prozora, njihove funkcionalnosti i odgovarajuće komande sa tastature su sledeće:

- *New File* – Opcija za kreiranje novog XML dokumenta pomoću praznog AATR šablona. Prečica: *Ctrl + N*
- *Open File* – Opcija koja omogućava korisniku učitavanje postojeće XML datoteke, nakon njenog izbora u dijalogu za biranje datoteka. Prečica: *Ctrl + O*
- *Save File* – Aktivacija ove opcije poziva jednu od dve funkcije, u zavisnosti od trenutnog stanja aplikacije: ukoliko trenutno obrađivana datoteka postoji na hard disku, čuva se novo stanje datoteke uz obaveštenje korisniku da je čuvanje uspešno (*Save*). Sa druge strane, ukoliko je datoteka tek kreirana i nije čuvana na disku, pojavljuje se dijalog za odabir lokacije i imena čuvane datoteke (*Save As...*). Ukoliko trenutno nije učitana ni jedna datoteka, opcija je nedostupna. Prečica: *Ctrl + S*
- *Create Issues* – Odabirom ove opcije pokreće se skripta za generisanje liste izveštaja. Prečica: *Ctrl + I*
- *Export to PDF* – Opcija koja omogućava čuvanje unesenih podataka u PDF formatu na odabranoj lokaciji, pomoću skripte koja sadrži šablon za kreiranje PDF dokumenta. Korišćenje opcije je moguće samo ako je trenutno učitana postojeća ili generisana nova XML datoteka. Prečica: *Ctrl + E*
- *Export all to PDF* – Poziva *Export to PDF* za svaki od XML fajlova unutar odabranog foldera ponaosob. Ova funkcionalnost se vrši u posebnoj niti programa, kako ne bi došlo do zaustavljanja cele aplikacije tokom njenog izvršavanja. Prečica: *Ctrl + Shift + E*

- *Add Child* – Ovaj alat dodaje novi podčvor trenutno selektovanom čvoru stabla. Različiti čvorovi mogu imati različit broj i vrste podčvorova, te se u slučaju više mogućih podčvorova za dodavanje, pomoću iskaćućeg menija korisniku pruža lista dostupnih mogućnosti. Prečica za ovaj alat ne postoji.
- *Delete item* – Alat koji vrši uklanjanje odabranog elementa iz stabla. Kako samo određeni elementi stabla mogu biti uklonjeni, dostupnost ovog alata zavisi od prirode obeleženog čvora. Nakon odabira ovog alata pojavljuje se dijalog koji od korisnika traži potvrdu brisanja elementa. Prečica za ovaj alat ne postoji.
- *Settings* – Alat koji omogućuje korisniku menjanje pojedinih podrazumevanih parametara programa. To se postiže pomoću pokretanja skripte `prefs.py`, koja kreira poseban dijalog sa mogućnošću odabira podrazumevanih putanja koje se korisniku nude pri učitavanju i čuvanju dokumenta, kao i pri stvaranju liste izveštaja. Dijalog za svaku podrazumevanu vrednost sadrži grafički element za prikaz trenutno odabrane putanje i dugme za odabir nove putanje preko dijaloga za izbor lokacije. Takođe postoji dugme za čuvanje izmena putanja kao i dugme za odustajanje od izmena i povratak na prethodno podešene vrednosti. Podešene i sačuvane putanje se upisuju u datoteku `config.ini`, iz koje se pri svakom pokretanju respektivnih funkcija učitavaju. Prečica: *Ctrl + P*
- *Quit* – Opcija koja, uz dodatno odobrenje putem dijaloga za potvrdu, zatvara aplikaciju. Prečica za ovaj alat ne postoji.
- *About* – Ova opcija nema ulogu u vršenju bilo kakvih izmena, već pruža kratak pregled informacija o autoru programa, godini završetka projekta, licenci, itd. Prečica: *F1*

Radi jednostavnijeg korišćenja aplikacije, operacije koje podrazumevaju izmenu trenutnog dokumenta (*Add Child*, *Delete Item*) mogu se izvršiti i pomoću iskakajućeg menija kome se pristupa desnim klikom na željeni čvor stabla.

Pre bilo koje operacije koja može da prouzrokuje gubitak unesenih podataka (kao što su *New File*, *Open File* i zatvaranje aplikacije) korisniku se, pomoću upitnog prozora, nudi mogućnost da sačuva dosadašnje izmene. Informacija o tome da li je trenutno otvoren dokument od momenta učitavanja menjan ili ne, data je korisniku u vidu zvezdice (\*) koja se, pored putanje do dokumenta (ili reči '*Untitled*', ukoliko je reč o tek kreiranom dokumentu) dodaje na naslov aplikacije na samom vrhu glavnog prozora.

## 4. Programsko rešenje

### 4.1 Korišćeni alati i okruženje

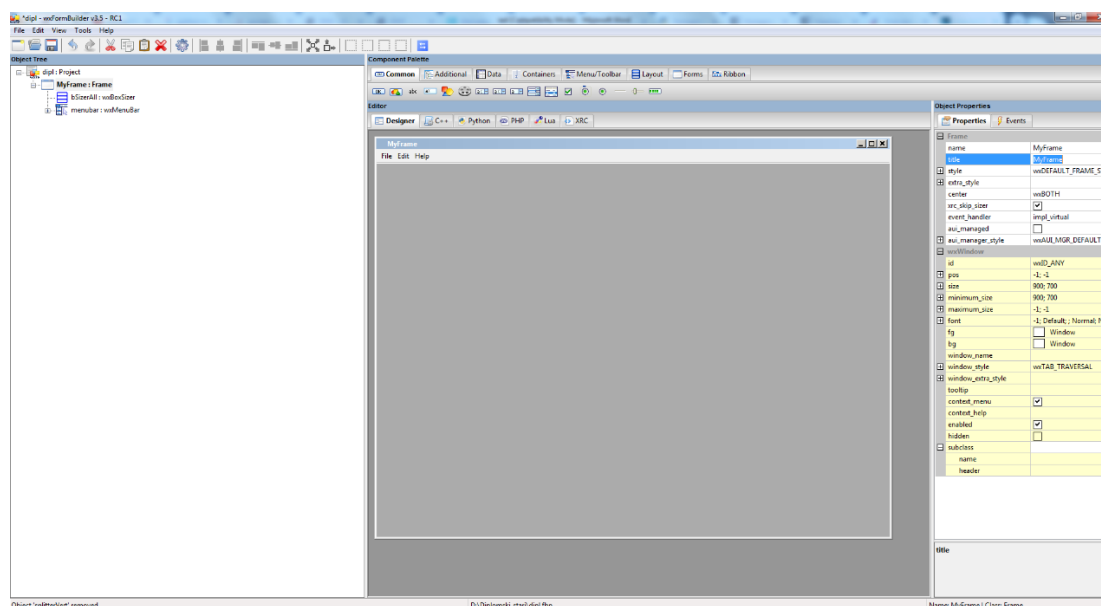
U ovom poglavlju dat je detaljniji opis strukture i programske realizacije projekta. Za realizaciju je korišćena verzija 2.7.15 programskog jezika *Python* uz praćenje PEP8 standarda za pisanje *Python* koda, koji sadrži određena pravila sa ciljem poboljšanja čitljivosti i konzistentnosti koda.

Kao glavni alat za unos koda i otklanjanje grešaka odabran je *Microsoft Visual Studio Code*. Ovaj kompaktni alat se pokazao kao veoma pogodan za potrebe projekta, pre svega zbog raznovrsnosti funkcionalnosti koje nudi i visokog stepena prilagodljivosti. Da bi se podesio interpreter ili prevodilac željenog programskog jezika, dovoljno je odabrati odgovarajuću nadogradnju iz ugrađenog menija za pretragu. Veoma veliki broj dostupnih održavanih nadogradnji umnogome je olakšao izbor alata na početku izrade projekta.

Grafička korisnička sprega implementirana je uz pomoć *wxwidgets* biblioteke. Ova biblioteka pruža korisničko sučelje za kreiranje grafičke sprege aplikacija. *wxwidgets* biblioteka je napisana u programskom jeziku C++, ali se može koristiti i sa raznim drugim jezicima, kao što su C#, Perl i Python (u kom slučaju je naziv biblioteke *wxpython*). Veliki izbor dostupnih klasa koje predstavljaju elemente grafičke sprege (eng. *widgets*) i alata za korišćenje znatno olakšava i skraćuje proces razvoja aplikacija.



Pored još nekoliko dostupnih *Python* biblioteka za izradu grafičkog prikaza, kao što su *Tkinter* i *PyQt*, *wxpython* se nametnuo kao najlogičniji izbor, jer pruža dobar balans između pozitivnih i negativnih strana korišćenja svake od alternative. Iako upoznavanje sa njim nije jednostavno kao sa *Tkinter*-om, i ne pruža toliko mogućnosti kao *PyQt*, *wxpython* je besplatna biblioteka otvorenog koda, sa mogućnošću pokretanja aplikacija na više platformi. Jedna od najpozitivnijih strana ovog rešenja jeste dostupnost alata *wxFormBuilder* (Slika 4.1), koji uklanja potrebu za određivanjem apsolutnih pozicija elemenata grafičkog prikaza direktno u kodu. Alat *wxFormBuilder* omogućava precizno postavljanje elemenata i jasan pregled grafičkog prikaza aplikacije bez prethodnog generisanja i pokretanja samog koda. Time je znatno ubrzano i olakšano kreiranje prikaza i raspoređivanje njegovih elemenata, a samim tim i izrada same aplikacije. Sami grafički elementi dostupni tokom razvoja se kreću od najjednostavnijeg dugmeta ili tekstualnog prikaza, do onih komplikovanijih, kao što su element grafičke sprege za prikaz podataka u vidu stabla ili liste, element za biranje datuma putem iskakajućeg kalendara, HTML okvir za prikaz koda, i drugi.



Slika 4.1: Izgled alata *wxFormBuilder*

Za učitavanje, parsiranje, i čuvanje XML dokumenata i korišćenje operacija nad njihovim čvorovima potrebno je učitati *Python* module *jinja2* i *xmltodict*.

*Jinja2* je jezik namenjen radu sa šablonima u *Python*-u, napravljen po uzoru na mehanizam za rad sa šablonima koji koristi *Django*.

Pomoću *Jinja2* šablona realizovana je funkcionalnost čuvanja izmenjenih datoteka na disku. Šablon (Slika 4.2) je izradjen po uzoru na postojeće AATR XML dokumente, a sadrži tekst sa praznim mestima za promenljive podatke, koje korisnik unosi preko polja za unos teksta.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="A-AcceptTestReport.xsl"?>
<ns2:A-AcceptTestReport xsi:schemaLocation="http://stefan.stojanovic.com TestReportScheme.0.9.1.xsd http://stefan.stojanovic.com/Supplier"
  <ns2:Release>{{ release_da.release }}</ns2:Release>
  <ns2:Platform>{{ release_da.platform }}</ns2:Platform>
  <ns2:SWC desc="Identification of SWC Release. This information will be used to check version consistency with embedded libraries (ve
    <SWC_Name>{{ swc_row.swc_name }}</SWC_Name>
    <SWC_Version desc="Release Version format must be NNN. BuildTimestamp format must be YYYYMMDDHHMMSS">
      <ReleaseKind>{{ swc_row.releasekind }}</ReleaseKind>
      <ReleaseVersion>{{ swc_row.releaseversion }}</ReleaseVersion>
      <IF-SET_Variant>{{ swc_row.ifset_variant }}</IF-SET_Variant>
      <IF-SET_Major>{{ swc_row.ifset_major }}</IF-SET_Major>
      <IF-SET_Minor>{{ swc_row.ifset_minor }}</IF-SET_Minor>
      <Platform_Major>{{ swc_row.platform_major }}</Platform_Major>
      <Platform_Minor>{{ swc_row.platform_minor }}</Platform_Minor>
      <Platform_BuildTimestamp>{{ swc_row.platform_buildtimestamp }}</Platform_BuildTimestamp>
      <SWC_Major>{{ swc_row.swc_major }}</SWC_Major>
      <SWC_Minor>{{ swc_row.swc_minor }}</SWC_Minor>
      <SWC_BuildTimestamp>{{ swc_row.swc_buildtimestamp }}</SWC_BuildTimestamp>
    </SWC_Version>
    <SWC_ASIL_Level>{{ swc_row.swc_asil_level }}</SWC_ASIL_Level>
    <SWC_Host>{{ swc_row.swc_host }}</SWC_Host>
    <Description>{{ swc_row.description }}</Description>
  </ns2:SWC>
  <ns2:Comment>{{ swc_da.comment }}</ns2:Comment>
  <ns2:DocumentInfo desc="Description of the document (document version, document author, document status)">
    <DocumentId>{{ swc_da.documentid }}</DocumentId>
    <DocumentVersion>{{ swc_da.documentversion }}</DocumentVersion>
    <DocumentStatus>{{ swc_da.documentstatus }}</DocumentStatus>
    <DocumentDescription>{{ swc_da.documentdescription }}</DocumentDescription>
    <DocumentAuthor desc="Please add author contact information. He/She releases the delivery and is the first contact person for qu
      <Name>{{ swc_da.name }}</Name>
      <Company>{{ swc_da.company }}</Company>
```

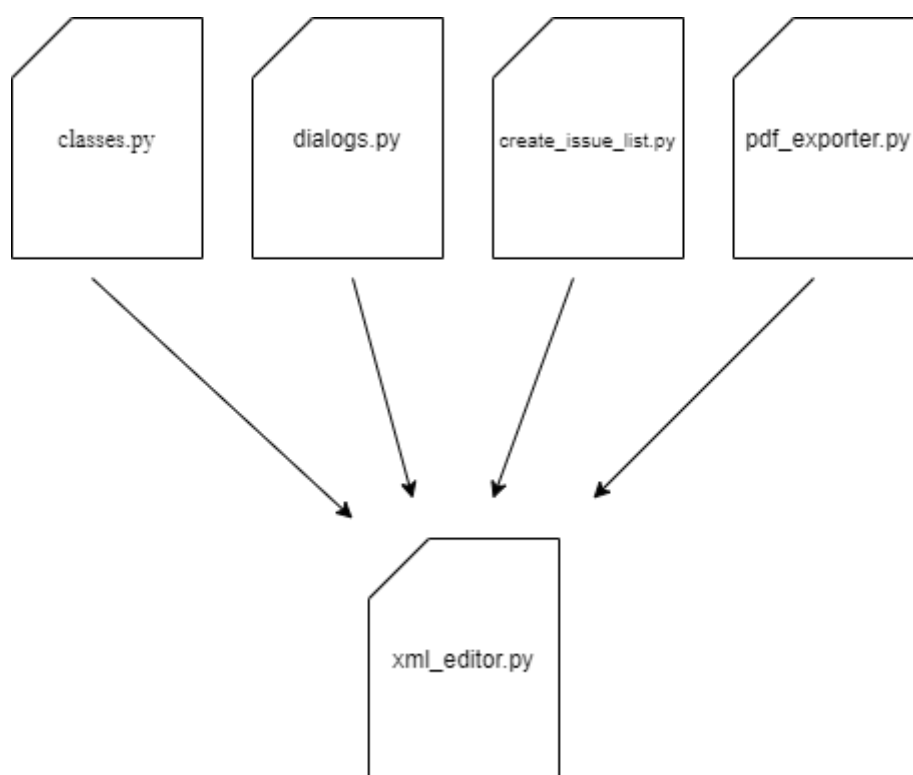
Slika 4.2: *jinja2* šablon za čuvanje podataka

*xmldict* je *Python* biblioteka namenjena olakšavanju rada sa podacima iz XML datoteka. Sadrži funkcije za učitavanje XML datoteke sa diska u formatu python rečnika (eng. *dictionary*), kao i funkcije za konverziju u suprotnom smeru, tj. pravljenje XML strukture od *Python* rečnika. Pomoću funkcije *parse* iz biblioteke *xmldict* realizovano je učitavanje postojeće XML datoteke u memoriju, nakon čega se XML elementima može pristupiti sintaksno identično kao i *Python* rečniku, što je u velikoj meri pojednostavilo implementaciju njihove dalje obrade.

Za potrebe uvezivanja modula u izvršnu (.exe) datoteku, i njenu jednostavniju distribuciju, korišćen je alat *PyInstaller*.

## 4.2 Struktura projekta

U ovom poglavlju detaljnije je opisana struktura samog koda projekta. Radi veće čitljivosti i modularnosti, kod se sastoji iz više logičkih celina, koje su implementirane u vidu *Python* modula (Slika 4.3). Neki od realizovanih modula su *xml\_editor.py*, *pdf\_exporter.py*, *classes.py*. Takođe postoji i datoteka *config.ini*, u kojoj se čuvaju podrazumevane vrednosti koje korisnik može menjati u bilo kom trenutku, pomoću skripte *prefs.py*, pokretanjem opcije *Settings*. U nastavku su pojašnjeni pojedinačni moduli i njihove metode.



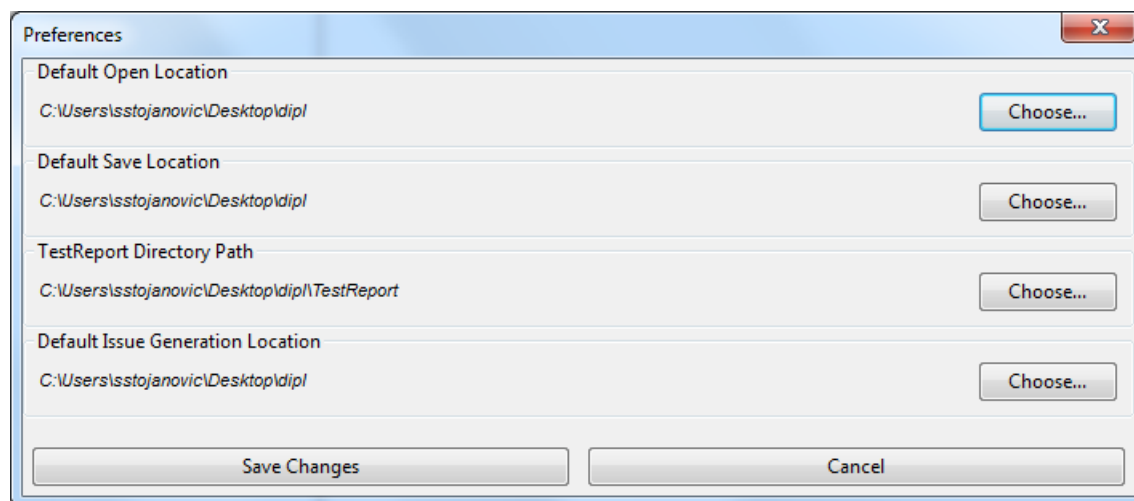
Slika 4.3: Organizacija *Python* modula koda projekta

### 4.2.1 Modul *xml\_editor.py*

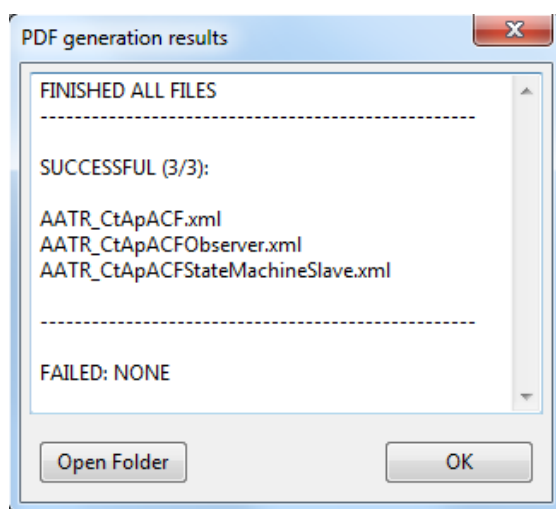
Ovaj modul je glavni deo projekta, i u njemu je implementirano najviše funkcionalnosti potrebnih za uspešno obavljanje zadatih operacija nad XML datotekom. Sadrži implementacije svih metoda zaduženih za samo kreiranje i prikaz grafičkih elemenata, i učitavanje i obradu podataka. Za glavni prozor aplikacije korišćen je objekat klase *wx.Frame*, koja predstavlja glavni okvir aplikacije. Unutar okvira se nalaze svi ostali elementi aplikacije. Klase *wx.Panel* i *wx.SplitterWindow* korišćene su za realizaciju glavnih panela i kliznih prozora. Svaki grafički element se nalazi unutar posebnog objekta klase *wx.Sizer*; *wxpython* klase koja služi za elegantnije uređenje i prikaz elemenata na željenim lokacijama unutar prozora aplikacije. Zadatak objekta klase *wx.Sizer* je da se pobrine da svi elementi unutar njega imaju odgovarajuće veličine i pozicije, bez obzira na promenu veličine glavnog prozora. Za prikaz okvira i naziva svakog od elemenata korišćena je podklasa *wx.StaticBoxSizer*. Levi panel sadrži objekte klase *wx.TreeCtrl* (za prikaz stabla) i *wx.SearchCtrl* (za prikaz polja za pretragu testnih slučajeva). Za prikaz samih podataka sa desne strane prozora i unos željenih izmena korišćena je klasa *wx.TextCtrl*, koja predstavlja polje za unos teksta, ili *wx.Choice*, grafički element za izbor iz liste vrednosti, ukoliko je reč o onim elementima koji mogu imati samo jedno od prethodno određenih vrednosti (npr. čvor *document\_status* može primiti samo vrednosti 'released', 'draft', ili 'in\_work', od kojih svaka označava trenutno stanje učitanoog dokumenta). Za potencijalno veoma dugotrajne operacije (kao što je izvoz podataka iz svih XML dokumenata u PDF format) korišćena je metoda *start\_new\_thread* modula *thread*, kako bi se omogućilo izvršavanje u posebnoj programskoj niti, i time sprečilo zamrzavanje prozora aplikacije za vreme njegovog trajanja.

### 4.2.2 Modul *dialogs.py*

Modul *dialogs.py* sadrži klase za kreiranje i prikaz posebnih prozora, koji su zbog svojih specifičnih namena morali biti redefinisani. Svaki od ovih prozora nasleđuje klasu *wx.Dialog*, uz redefinisane potrebne karakteristike, kao što su veličina prozora, broj i raspored grafičkih elemenata unutar njega, natpisi i funkcije tastera, i slično. Za potrebe ovog projekta implementirana su dva ovakva prozora: *PDFDialog* (Slika 4.4), koji služi da obavesti korisnika o detaljima uspešnosti operacije čuvanja svih dokumenata u PDF formatu, i *PrefsDialog* (Slika 4.5), čiji je zadatak da korisniku omogući izbor podrazumevanih putanja za učitavanje i čuvanje dokumenata, i generisanje liste izveštaja, i sačuva izabrane putanje u *config.ini* datoteku.



Slika 4.4: Namenski prozor za podešavanje podrazumevanih putanja



Slika 4.5: Namenski prozor za prikaz rezultata generisanja PDF dokumenata

### 4.2.3 Modul *pdf\_exporter*

U modulu *pdf\_exporter.py* realizovane su metode potrebne za kreiranje novog PDF dokumenta. Pomoću ove opcije, podaci iz svakog AATR XML dokumenta se mogu predstaviti u PDF formatu, koji može biti umnogome pregledniji i jednostavniji za čitanje. Generisani PDF dokument (Slika 4.6) je struktuiran po uzoru na postojeće AATR PDF dokumente generisane prethodno korišćenim alatom. Osnovni podaci iz XML dokumenta prikazani su na prvoj strani, dok je većina ostalih podataka prikazana tabelarno. Na drugoj strani dokumenta se nalazi njegov sadržaj. U zaglavlju i podnožju svake stranice nalaze se informacije o nazivu projekta za koji je posmatrani dokument vezan, tipu XML dokumenta (AIT, AATR, itd.), autoru, datumu i autorskim pravima kreiranja PDF dokumenta. Za implementaciju ove opcije korišćena je *FPDF Python* biblioteka, koja sadrži funkcije za iscrtavanje linija, ispisivanje teksta, i postavljanje željenih slika na određenu poziciju unutar stranice. Šablon osnovnog oblika PDF dokumenta se popunjava podacima unetim u stablo podataka putem polja za unos teksta i izbor vrednosti. Ovakvom implementacijom postignuto je da je svaki generisani PDF dokument istog oblika, dok su jedina razlika uneti podaci, što znatno povećava nivo preglednosti generisanih dokumenata.







## 1. Application Acceptance Test Result

This chapter documents the overall test results of the performed Application Acceptance Test.

### 1.1 SW-C Overall Test Result & Integration Recommendation

SWC Name	Version	Integration recomm.
Software Component Name	4.20	passed

Table 1 Application Acceptance Test Result & Integration Recommendation

	Passed, integrate all test cases were without errors or only priority C failures occurred
	Failed, do not integrate at least one priority A failure occurred
	Failed, but still integrate at least one priority B failure occurred
	Not Executed

### 1.2 Statistics

	Priority A		Priority B		Priority C		Total	
Overall number of test cases	2	100%	1	100%	1	100%	4	100%
Executed test cases	2	100.00%	1	100.00%	0	0.00%	3	75.00%
Not executed test cases	0	0.00%	0	0.00%	1	100.00%	1	25.00%
Passed test cases	1	50.00%	0	0.00%	0	0.00%	1	25.00%
Failed test cases	1	50.00%	1	100.00%	0	0.00%	2	50.00%

Table 2 Statistics

Slika 4.6: Jedna od stranica generisanog PDF dokumenta

#### 4.2.4 Modul *create\_issue\_list.py*

Modul *create\_issue\_list.py* je preuzeti modul koji učitava podatke iz postojećih AATR XML datoteka i u zavisnosti od njihovih vrednosti ispisuje niz poruka u posebno generisanu tekstualnu datoteku (Slika 4.7). Jedan od bitnih ulaznih parametara ovog modula jeste putanja na kojoj će se generisati tekstualna datoteka, a koju korisnik unosi u config.ini datoteku pomoću opcije *Settings*.

```
##PV_ISSUE_SMC_NAME##CTapACF##PV_ISSUE_MESSAGE##_AAT_

##PV_ISSUE_SMC_NAME##CTapACF##PV_ISSUE_MESSAGE##_AAT_comment_It is recommended to integrate since SMC is buildable and no PRIO_A fails.

##PV_ISSUE_SMC_NAME##CTapACF##PV_ISSUE_MESSAGE##Proceed with AIT.

##PV_ISSUE_SMC_NAME##CTapACFObserver##PV_ISSUE_MESSAGE##_AAT_
- Asils are different; found in releaseNotes [QM], expected from Architecture [B] [ASIL level check] [PRIO_C]
- 59.00% of RTE interfaces not used, 41.00% used. [RTE interface usage check] [PRIO_C]
- Not allowed flags found! [Check allowed compiler flags ] [PRIO_A]
- RequiredLevel is different than zero: 427 [MISRA level check] [PRIO_B]
- List of not used mandatory compiler flags: -pedantic-errors, -c List of not used mandatory linker flags: -llicore, -LC:/data/vx7-Industrial/workspace/qmrtpl../vsb/user/lib/common, -pedantic-errors

##PV_ISSUE_SMC_NAME##CTapACFObserver##PV_ISSUE_MESSAGE##_AAT_comment_It is recommended to integrate since SMC is buildable.

##PV_ISSUE_SMC_NAME##CTapACFObserver##PV_ISSUE_MESSAGE##Proceed with AIT.

##PV_ISSUE_SMC_NAME##CTapACFStateMachineSlave##PV_ISSUE_MESSAGE##_AAT_
- Release notes missing important information [Release notes check] [PRIO_B]
- AITR missing important information [Supplier's AITR check] [PRIO_B]
- AITR's limits not equals to expected limits [Supplier's AITR limits check] [PRIO_B]
- 55.00% of RTE interfaces not used, 45.00% used. [RTE interface usage check] [PRIO_C]
- Not allowed flags found! [Check allowed compiler flags ] [PRIO_A]
- RequiredLevel is different than zero: 22 [MISRA level check] [PRIO_B]
- Some hex files found in 05_data folder cannot be found in Release Notes. [Availability of dataset files check] [PRIO_B]

##PV_ISSUE_SMC_NAME##CTapACFStateMachineSlave##PV_ISSUE_MESSAGE##_AAT_comment_It is recommended to integrate since SMC is buildable.

##PV_ISSUE_SMC_NAME##CTapACFStateMachineSlave##PV_ISSUE_MESSAGE##Proceed with AIT.
```

Slika 4.7: Primer generisane liste izveštaja

#### 4.2.5 Modul *classes.py*

Ovaj modul sadrži deklaracije i definicije svih klasa korišćenih u procesu čuvanja unetih podataka u XML datoteku pomoću *jinja2* šablona. Pre samog čuvanja, podaci iz AATR dokumenta se raspoređuju po atributima različitih klasa, koje se kasnije uvezuju sa promenljivama iz *jinja2* šablona. Ovaj korak pruža sigurnost da će svaki podatak koji je potrebno sačuvati završiti na svom odgovarajućem mestu unutar šablona. Opisivanje svakog atributa svake klase ne bi bilo smisleno za koncept ovog rada, te će biti navedeni samo nazivi nekih od klasa: *ReleaseData*, *SWCData*, *TestCase*, *TestCaseTable*, *TableCell*, *Reference*. Nazivi atributa direktno preslikavaju nazive XML elemenata koje predstavljaju, a klase su nazivane po logičkim celinama koje predstavljaju njihovi atributi.



## 4.3 Tok programa

Pri učitavanju XML datoteke poziva se funkcija *file\_open\_on\_menu* u kojoj se, nakon zatvaranja prozora za odabir datoteke, pozivanjem ugrađenih funkcija *loads* i *dumps* iz modula *json*, parsira XML dokument. Time se svi njegovi elementi i njihove vrednosti prevode u *Python* objekat.

Za ovu svrhu odabran je *Python* rečnik (eng. *dictionary*), zbog toga što očuvava hijerarhiju podataka, i time omogućava jednostavan prenos XML elemenata i njihovih atributa i vrednosti bez gubitka podataka i njihovih pozicija unutar datoteke. *Python* rečnik je struktura podataka specifična za programski jezik *Python*. Funkcioniše po principu ključne reči i vrednosti, gde, u konkretnom slučaju ovog rada, ključna reč predstavlja naziv XML elementa (*tag*), a vrednost ono što se nalazi u njemu (*text*). Elementima rečnika je tada moguće pristupiti indeksiranjem po nazivu elementa (*ime\_rečnika[ime\_elementa] = vrednost*). Vrednost elementa rečnika može biti bilo kog tipa, pa i tipa samog rečnika, čime se dobija ugnježdeni rečnik. Ova struktura se pokazala najpogodnijom za predstavu hijerarhije podataka XML datoteke, zbog intuitivnog pristupa elementima pri učitavanju i upisivanju podataka.

Ukoliko se odabere kreiranje novog XML dokumenta, istim postupkom parsira se prazan AATR XML šablon (metoda *create\_blank\_file*), što rezultuje *Python* rečnikom koji sadrži ispravnu hijerarhiju, ali prazne vrednosti elemenata. Sledeće, u funkciji *tree\_fill* se kreirani objekat klase stabla popunjava podacima iz dobijenog ugnježdenog rečnika, postavlja se proširen prikaz korenskog čvora stabla, i automatski se na njega postavlja odabir. Postojanje imenskih prostora (eng. *namespace*) u XML dokumentu može dodatno poremetiti parsiranje i rad sa njegovim elementima. Da bi se to eliminisalo, funkcija *del\_namespace* se koristi za uklanjanje imenskog prostora iz naziva elemenata, ukoliko on postoji. Nakon ovog postupka nazivi svih elemenata ugnježdenog rečnika predstavljaju same nazive odgovarajućih XML elemenata. Imenski prostor se ponovo dodaje na nazive XML elemenata *jinja2* šablonom, prilikom čuvanja datoteke, čime se sprečava njegov gubitak. Na kraju se omogućava pristup alatima za dodavanje i uklanjanje čvorova, generisanje liste izveštaja i PDF dokumenta, dodaje se naziv učitane XML datoteke na naslov prozora aplikacije, i čeka se unos korisnika.

Pored toga što je adekvatno rešenje za grafički prikaz podataka, objekat klase *wx.TreeCtrl* je pogodan i za čuvanje izmena na čvorovima direktnim uvezivanjem čvora i odgovarajuće vrednosti. Naime, svo čitanje i pisanje podataka u učitanoj datoteku realizovano je preko samog objekta stabla. Time je postignuto da je objekat stabla podataka osnova za sve operacije vezane za izmenu i unos podataka – prilikom svake izmene tekstualnog polja sa desne strane prozora aplikacije, u funkciji *on\_edit* odmah se uvezuju novi podaci sa odgovarajućim čvorom stabla. Uz pomoć imena čvora, koje se dobija pozivom metode objekta klase stabla, *GetItemText*, pristupa se čvoru stabla i stara vrednost podatka čvora zamenjuje se novom, pozivom metode *SetItemData*. Ukoliko je menjano polje sa nazivom 'Name' jednog od čvorova sa zajedničkim nazivom (TestCase, Acronym, Reference itd.), funkcijom *update\_name* se nazivu čvora u stablu sa leve strane prozora dodaje uneseni naziv između srednjih zagrada '[ ]', čime se ostvaruje lakši pregled čvorova stabla sa identičnim nazivom. U slučaju jednog od čvorova sa nazivom 'Revision' ova funkcija uzima vrednost polja 'Date'.

U sledećoj tabeli dat je pregled metoda za dodavanje praznih elemenata u stablo. Tabela sadrži naziv elementa za dodavanje, naziv metode, naziv *jinja2* šablona korišćenog za učitavanje strukture novog elementa (formata *.template*), i matičnog elementa na koji se novi element dodaje. Sve navedene metode funkcionišu isto kao već opisana metoda *tree\_fill*.

Element	Metoda	Matični čvorovi	Šablon
<TestCase>	<i>add_test_case</i>	<Test>	test_case
<TestCaseTable>	<i>add_test_case_table</i>	<Test>	tct
<TableRow>	<i>add_table_row</i>	<TestCaseTable>	/
<TableCell>	<i>add_table_cell</i>	<TableHeader>	/
<Acronym>	<i>add_acronym</i>	<Acronyms>	acronym
<Reference>	<i>add_reference</i>	<References>	reference
<TestArtefact>	<i>add_ta</i>	<IntegratorsTestArtefacts> <SuppliersTestArtefacts>	test_artefact
<AdditionalTool>	<i>add_tool</i>	<AdditionalSoftwareUsed>	add_tool
<SuppliersAdditionalInfo>	<i>add_info</i>	<SupplierAdditionalInfo>	/
<KnownLimitation>	<i>add_known_lim</i>	<A-AcceptTestReport>	/
<Revision>	<i>add_revision</i>	<Revisions>	revision

Tabela 4.1: Metode za dodavanje novih čvorova u stablo podataka

Kada se odabere opcija *Save* (ili *Save As...*) i unese željena putanja čuvanja datoteke, poziva se funkcija *save\_data*. U ovoj funkciji se, iteracijom kroz sve čvorove, iz stabla svi podaci raspoređuju u objekte odgovarajućih klasa, za šta se takođe koriste pomoćne funkcije *save\_class* (koja popunjava objekte jedinstvenih klasa) i *save\_instance* (koja popunjava objekte koji su instance jedne iste klase, kao što su *TestCase*, *Acronym*, *Reference*, itd.). Na kraju funkcije *save\_instance* popunjena instanca klase se dodaje u odgovarajuću listu objekata jedne klase, radi lakšeg kasnijeg iteriranja. Klase i njihovi atributi su modelovani po uzoru na strukturu AATR XML datoteke.

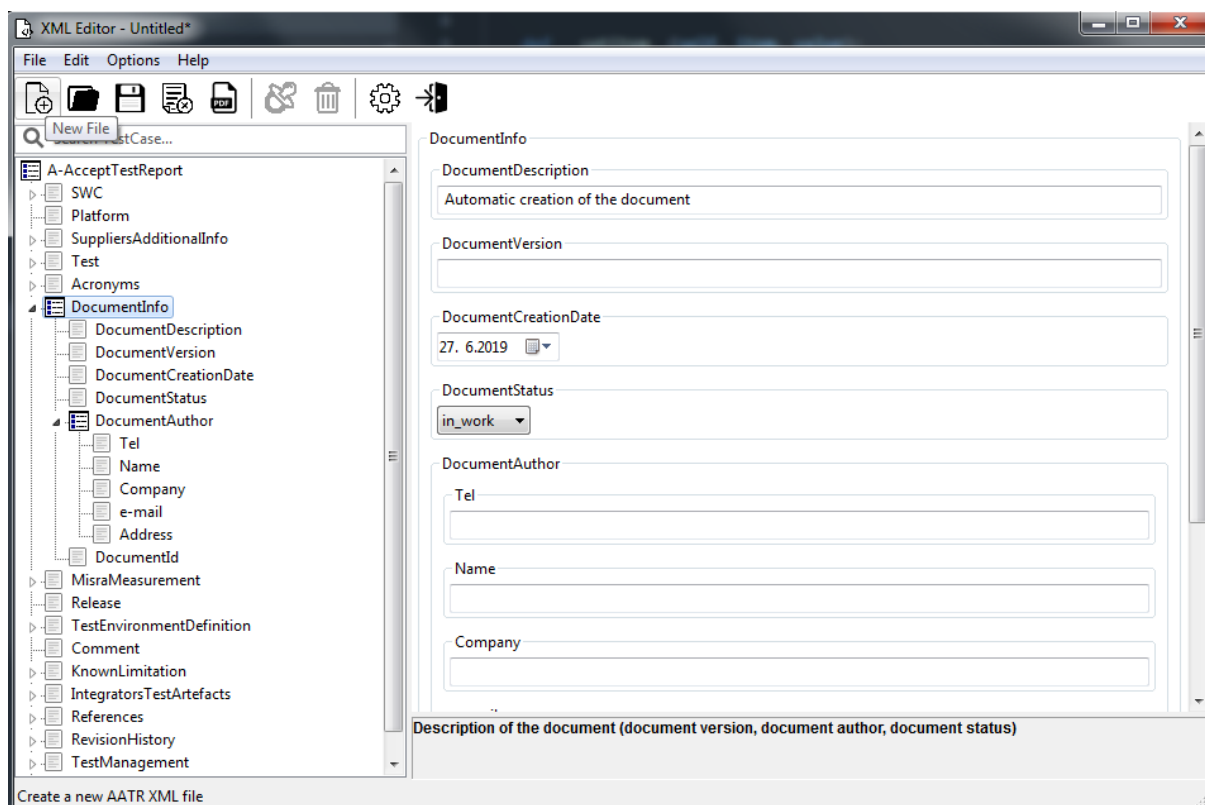
Samo čuvanje u datoteku na disk se obavlja u funkciji *save\_to\_disk*, unutar koje se prave potrebni objekti klase za funkcionisanje *jinja2* šablona (*FileSystemLoader*, *Environment*, *template*), kao i promenljiva *context*, čija je uloga mapiranje objekata i promenljivih iz python skripte na promenljive u *jinja2* šablonu. Za elemente sa identičnim nazivima koriste se odgovarajuće liste instanci klase, od kojih svaka predstavlja po jedan element. Konačno, pozivom funkcije *render* klase *template*, i kreiranjem objekta datoteke i pozivom funkcije *write*, *jinja2* popunjava zadati šablon unesenim podacima iz stabla, čime se na odabranoj putanji na disku dobija rezultujuća XML datoteka.

U bilo kom momentu moguće je odabrati opciju *Settings* sa trake sa opcijama. Tada se pozivom skripte *dialogs.py* kreira poseban prozor, koji pruža korisniku mogućnost izbora podrazumevanih putanja za učitavanje i čuvanje datoteka, kao i putanju prema *TestReport* direktorijumu, koja je neophodna za nesmetano funkcionisanje opcija *Create Issues* i *Export all to PDF*.

Izabrane putanje se na kraju upisuju u datoteku *config.ini* pomoću objekta klase *ConfigParser* (iz istoimenog modula), iz koje se ponovo učitavaju pri sledećem pokretanju programa. Funkcionisanje aplikacije je koncipirano tako da je direktorijum *TestReport* podrazumevano mesto čuvanja svih XML datoteka koje je potrebno menjati, čuvati u PDF formatu, ili za koje je potrebno generisati listu izveštaja. Ukoliko putanja do ovog direktorijuma nije tačna, program korisniku prikazuje grešku, i automatski otvara *Settings* prozor, kako bi se mogla odabrati validna *TestReport* putanja.

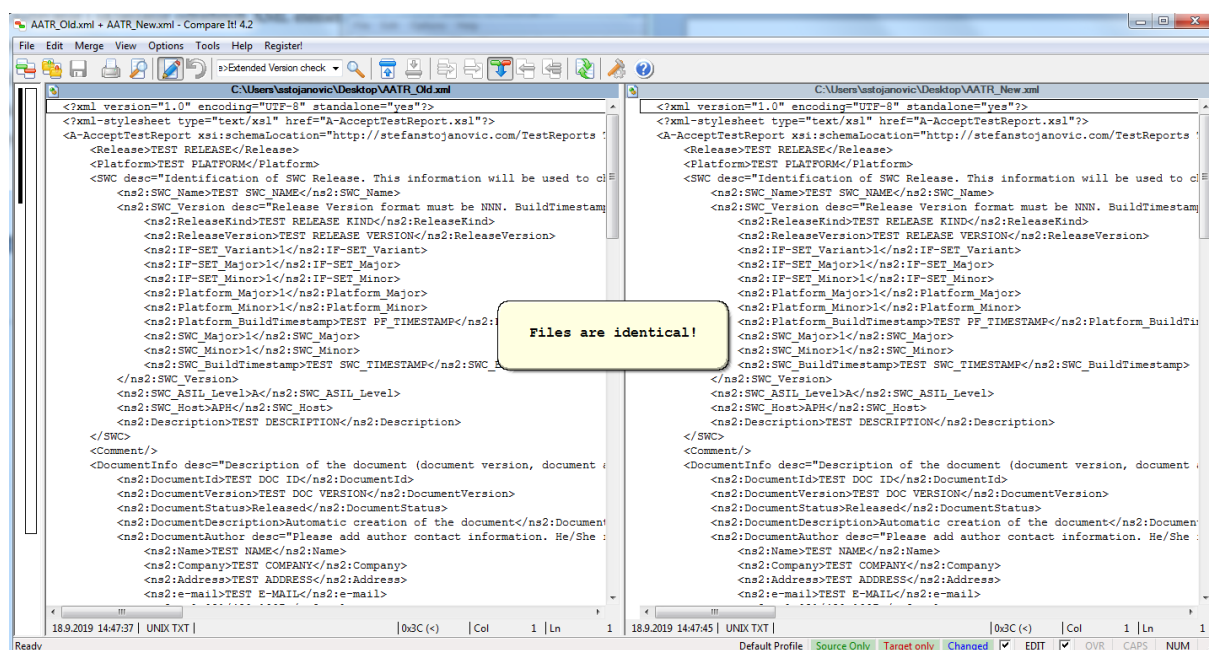
## 5. Rezultati

Za uvezivanje svih modula sa finalnim kodom u izvršnu (.exe) datoteku korišćen je *PyInstaller*, besplatni alat za distribuciju *Python* aplikacija. Pomoću njegovih funkcionalnosti moguće je podesiti parametre samog uvezivanja, kao što su: otvaranje komandnog prozora zajedno sa aplikacijom, uključivanje svih datoteka koje predstavljaju zavisnosti aplikacije, podešavanje željene ikonice same izvršne datoteke i slično.



Slika 5.1: Izgled finalne aplikacije tokom korišćenja

Gotova aplikacija (Slika 5.1) je testirana na više različitih računara sa različitim verzijama *Windows* operativnog sistema (7, 8 i 10). Osnovni test ispravnosti finalne aplikacije vršen je po sledećem principu: za isti ulaz, novorazvijena aplikacija mora da pruži iste izlazne rezultate kao i prethodno korišćena aplikacija. Tačnije, ukoliko korisnik pomoću obe aplikacije unese identične izmene u dva (takođe identična) XML dokumenta, potrebno je da nakon unosa te dve datoteke nemaju nikakve razlike. Za poređenje rezultujućih datoteka nastalih testiranjem aplikacije, korišćen je program *Compare It!*, alat za poređenje i spajanje datoteka raznih formata. Kako su datoteke dobijene pri unosu istih podataka identične, zaključeno je da je osnovni test gotove aplikacije uspešan (Slika 5.2).



Slika 5.2: Rezultat poređenja izlaznih datoteka prethodno korišćene aplikacije i završne aplikacije projekta

Nakon provere tačnosti osnovnih funkcionalnosti, istim postupkom testirane su i ostale implementirane funkcionalnosti. Zaključeno je da dobijene liste izveštaja nemaju razlike u odnosu na one generisane pomoću prethodno korišćene aplikacije. Isto važi i za PDF dokumente, kreirane bilo kojom od dve dostupne opcije (samo za trenutno učitane podatke ili za sve XML dokumente unutar *TestReport* direktorijuma). Kreiranje i učitavanje konfiguracione datoteke *config.ini*, u kojoj se čuvaju podrazumevane putanje unesene putem posebnog prozora opcije *Settings*, je takođe u skladu sa očekivanjima.

Na svakom od računara korišćenim za testiranje obavljene su sve navedene provere gotove aplikacije, sa pozitivnim rezultatima. Na osnovu njih, zaključeno je da su uspešno i tačno implementirane sve funkcionalnosti koje finalna aplikacija nudi. Među njima su:

- Kreiranje nove AATR datoteke u XML formatu
- Učitavanje postojeće i čuvanje izmenjene AATR datoteke na hard disku
- Izvoz unesenih podataka u prethodno definisani dokument PDF formata za trenutno otvorenu ili sve XML dokumente unutar *TestReport* direktorijuma
- Generisanje liste izveštaja i njeno čuvanje u tekstualnu datoteku na hard disku
- Dodavanje i uklanjanje određenih XML elemenata (čvorova stabla podataka)
- Podešavanje podrazumevanih putanja ponuđenih pri otvaranju i čuvanju datoteka, kao i pri generisanju liste izveštaja, i njihovo čuvanje u konfiguracionoj datoteci radi kasnijeg ponovnog učitavanja
- Prikazivanje informacija o verziji, godini izrade i autoru aplikacije.

Gotova aplikacija je u potpunosti prenosiva, uz nekoliko zavisnosti. Da bi se mogli učitavati, proširivati i čuvati XML dokumenti, potrebno je da direktorijum sa odgovarajućim *jinja2* šablonima bude smešten unutar direktorijuma finalne aplikacije. Na ovoj lokaciji treba da se nalazi i direktorijum sa svim slikama korišćenim u svrhu prikazivanja ikonica unutar aplikacije. Takođe, potrebno je da postoji *TestReport* direktorijum, koji će sadržati AATR XML dokumente koji se obrađuju.

## 6. Zaključak

U ovom radu dat je opis izrade i funkcionisanja grafičke korisničke sprege za uređivanje dokumenata za prikaz AATR rezultata u XML formatu. Implementirana je aplikacija koja omogućava korisniku sve osnovne operacije koje su potrebne za uspešnu izmenu AATR XML dokumenata, uključujući učitavanje dokumenata, njihovo čuvanje u različitim formatima, kao i dodavanje i uklanjanje željenih XML elemenata. Time je pokazano je da se u *Python* grafičkom okruženju može napraviti aplikacija koja po broju funkcionalnosti i brzini izvršavanja uspešno može da parira aplikacijama napisanim u programskim jezicima kao što su *C#* ili *Java*.

Za implementaciju korišćen je programski jezik *Python*, zajedno sa bibliotekom za elemente grafičke sprege *wxpython*, metodama biblioteke *json* za učitavanje i *jinja2* šablonom za čuvanje podataka. Za razvojno okruženje odabran je *Microsoft Visual Studio Code* zbog svoje dostupnosti i širokog spektra mogućnosti. Iz istih razloga, pri izradi grafičke predstave aplikacije korišćen je alat *wxFormBuilder*.

Realizovana aplikacija pruža sve funkcionalnosti za osnovne potrebe izmene i čuvanja AATR XML dokumenata. Brzina izvršavanja kao i preglednost aplikacije su povećane u odnosu na prethodno korišćeni alat. Još jedna prednost ovog rešenja jeste prenosivost, jer njegovo pokretanje i korišćenje ne zahteva nikakva prethodna podešavanja okruženja.

Takođe, postoji veliki broj mogućih proširenja datog rešenja zadatka. Sama priroda aplikacije za izmenu dokumenata bilo kog formata nosi sa sobom naizgled neograničen broj potencijalnih dodatnih funkcionalnosti i opcija. Neki od njih su: omogućavanje korisniku rad sa XML datotekama bilo kog oblika pored AATR (npr. AIT i SIT), dodavanje komande za poništavanje poslednjeg unosa (eng. *Undo*), implementacija mogućnosti čuvanja podataka u više različitih formata, funkcionisanje aplikacije na više različitih platformi, i drugo.

## 7. Literatura

- [1] Al Sweigart: *Automate the Boring Stuff with Python*, 2015.
- [2] Mark Lutz: *Programming Python*, 2014.
- [3] Noel Rappin, Robin Dunn: *wxPython in Action*, 2006.
- [4] Mike Driscoll: *WxPython Recipes: A Problem - Solution Approach*, 2017.
- [5] Cody Precord: *WxPython 2.8: Application Development Cookbook : Quickly Create Robust, Reliable and Reusable WxPython Applications*, 2010.
- [6] wxPython API documentation <https://wxpython.org/Phoenix/docs/html/>,  
datum pristupa: 16.8.2020.
- [7] wxPyWiki <https://wiki.wxpython.org/>, datum pristupa: 16.8.2020.
- [8] Python tutorials <https://pythonspot.com/>, datum pristupa: 16.8.2020.