Alat za klasifikaciju parametara modela sustava

Hlavsa, Nikola

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:445016

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-08-17**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

ALAT ZA KLASIFIKACIJU PARAMETARA MODELA SUSTAVA

Diplomski rad

Nikola Hlavsa

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 07.12.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Nikola Hlavsa
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-982R, 30.09.2020.
OIB studenta:	16634214951
Mentor:	Doc.dr.sc. Zdravko Krpić
Sumentor:	Dr.sc. Ivan Vidović
Sumentor iz tvrtke:	Kristina Tošeski
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	Dr.sc. Ivan Vidović
Član Povjerenstva 2:	Izv. prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Alat za klasifikaciju parametara modela sustava
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Prikaz modela sustava se u većini slučajeva u okviru AUTOSAR (engl. AUTomotive Open System ARchitecture) metodologije svodi na prikaz u ARXML (engl. AUTOSAR EXtensible Markup Language) formatu kojeg je neophodno parsirati i prilagoditi ciljanim generatorima koda. Glavni motiv za ovaj međukorak je optimizacija, gdje se umjesto zahtjevnog parsiranja XML stabla za svaki generator pojedinačno, koriste objekti prilagođeni programskom jeziku u kojem je pisan generator, ili su prilagođeni formatu podataka samog generatora. Nedostaci ovakvog pristupa su opterećenje memorijskih i procesorskih resursa prilikom procesa prilagođavanja ARXML formata te ovakav pristup nije portabilan te je specifičan za zahtjeve korištenih generator
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Dobar (3)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 1 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	07.12.2020.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 23.12.2020.

Ime i prezime studenta:	Nikola Hlavsa
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-982R, 30.09.2020.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Alat za klasifikaciju parametara modela sustava**

izrađen pod vodstvom mentora Doc.dr.sc. Zdravko Krpić

i sumentora Dr.sc. Ivan Vidović

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Elektronika u vozilima dobiva sve više na značaju. Broj mikroupravljača u automobilu stalno se povećava. S povećanjem broja elektroničkih upravljačkih jedinica (engl. *Electronic Control Unit* - ECU), povećava se i količina i složenost funkcija koje inženjeri moraju dizajnirati, kodirati, testirati i implementirati. Virtualna ECU programska podrška emulira stvarni ECU u simulacijskom scenariju. Virtualni ECU može sadržavati komponente aplikacije i osnovne programske podrške (engl *Basic SoftWare* – BSW) te pruža funkcionalnosti usporedive s onima stvarnog ECU-a ili njemu namijenjenog koda programske podrške. Današnja ECU programska podrška sadrži brojne programske komponente (engl. *SoftWare Component* - SWC) s brojnim interakcijama. U velikim ECU mrežama često instaliranim u današnjim vozilima, broj SWC-a može lako doseći tisuće. Budući da zadatak razvoja komponenata ECU-a obično dijeli nekoliko odjela ili čak različite tvrtke, ne moraju se testirati i potvrditi samo SWC-ovi, već i interakcija između njih. Što se ranije u razvojnom procesu pronađu pogreške i nedosljednosti, to ih je brže i jeftinije ispraviti.

Automatizirano generiranje programskog koda temeljenog na modelu široko se koristi u razvoju i testiranju ECU-ova. Tijekom razvoja ECU-ova potrebna su opsežna testiranja implementiranih upravljačkih algoritama. Poznavanje unutarnjih vrijednosti i stanja programske podrške ECU-ova neophodno je za verifikaciju ispravne funkcionalnosti programske podrške složenih upravljačkih algoritama. Procesi mjerenja omogućuju promatranje vrijednosti interne memorije ECU-ova spremljene u varijablama koje nastaju iz funkcijskog modela. Nadalje, proces kalibracije omogućuje podešavanje upravljačkih parametara unutar ECU-a. Iz tih razloga su mjerenje i kalibracija široko primijenjeni unutar, ne samo automobilske, već i ostalih industrija. Stoga je integriranje mjerenja i kalibracije važan zadatak za inženjere koji rade razvoj ECU-ova temeljenih na modelu.

Prikaz modela sustava se u nekim slučajeva u okviru AUTOSAR (engl. *AUTomotive Open System ARchitecture*) metodologije svodi na prikaz u ARXML (engl. *AUTOSAR EXtensible Markup Language*) formatu kojeg je neophodno parsirati i prilagoditi ciljanim generatorima koda. Glavni motiv za ovaj međukorak je optimizacija, gdje se umjesto zahtjevnog parsiranja XML stabla za svaki generator pojedinačno, koriste objekti prilagođeni programskom jeziku u kojem je pisan generator, ili su prilagođeni formatu podataka samog generatora. Nedostaci ovakvog pristupa su opterećenje memorijskih i procesorskih resursa prilikom procesa prilagođavanja ARXML formata te ograničena prenosivost.

U okviru ovog diplomskog rada potrebno je migrirati serijalizirani ARXML model ECU-a (`model.zip`) na SQL bazu podataka te izraditi XCP generator u C++ programskom jeziku na temelju postojećeg generatora napisanog u Python programskom jeziku te provesti studiju izvodljivosti. XCP generator je alat koji služi za generiranje A2L datoteka iz ARXML modela ECU-a. Te datoteke služe za kalibraciju i komunikaciju s ECU-ovima preko XCP komunikacijskog protokola.

U drugom poglavlju je dan pregled znanstvenih radova koji koriste A2L generatore te je novi XCP generator uspoređen s postojećim XCP generatorom. Nadalje, u trećem poglavlju su opisani standardi i tehnologije korištene u diplomskom radu kako bi se bolje razumjela njihova uporaba u automobilskoj industriji. U četvrtom poglavlju je opisana realizacija XCP generatora koja je podijeljena na tri faze: Migriranje modela, izmjena postojećeg XCP generatora i izrada novog XCP generatora. Naposljetku je u petom poglavlju napisan zaključak u kojem su uzeti u obzir uspjesi i nedostatci diplomskog rada.

2. PREGLED LITERATURE I POSTOJEĆA RJEŠENJA

Kako bi se realizirao alat za klasifikaciju parametara modela, njegovu izvedbu je, zbog jednostavnije realizacije, potrebno podijeliti na model parser i XCP generator. Model parser zadužen je za čitanje datoteke prilagođenog ARXML ECU modela, izdvajanje bitnih informacija te njihovo zapisivanje u SQL bazu podataka. Funkcija XCP generatora je dohvaćanje podataka iz SQL baze podataka i generiranje A2L datoteka. U ovom poglavlju su opisani postojeći alati na kojima se temelji ovaj diplomski rad te postojeći radovi vezani uz ovu tematiku. U priloženoj literaturi autori navode korištenje MATLAB alata, TargetLinka i DaVinci Developera. Autori u [2] predlažu korištenje MATLAB skripti i TargetLink-a za generiranje A2L datoteka. U [3] autori generiraju A2L datoteke uz pomoć postojećih A2L datoteka i povezničke karte, dok autori u [4] generiraju A2L datoteke iz SIMULINK modela.

Autori u [1] predlažu novi pristup razvoju programske podrške ECU-ova koji se u potpunosti temelji na simulacijskom okruženju. Za generiranje A2L datoteka autori koriste vlastiti A2L generator koji je podijeljen na parser i spajatelj (engl. *merger*). Parser izvršava funkcije parsiranja i generiranja. Funkcija parsiranja čita C kod te traži CDL (engl. *Configuration Description Language*) specifičnu sintaksu, dok funkcija za generiranje interpretira sakupljenu CDL sintaksu te generira početnu A2L datoteku koja se sastoji od funkcija i grupnih modula prema ASAM (engl. *Association for Standarisation of Automation and Measuring Systems*) standardu. Merger je zadužen za spajanje (engl. *merging*) i generiranje. Funkcija spajanja za ulaz uzima početnu A2L datoteku i PMF (engl. *Project Managment File*) te ih spaja u konačnu A2L datoteka koja je kompatibilna s alatima za kalibraciju. A2L generator zatim izvodi funkciju generiranja pri kojoj generira C kod koji opisuje sučelje između ECU-a i korištenog aplikacijskog sustava. U diplomskom radu, kao i u radu [1], koristi se sličan koncept parsiranja modela, odnosno u slučaju diplomskog rada, nad prilagođenim ARXML ECU modelom.

U [2] autori za generiranje A2L datoteka koriste TargetLink automatski kod generator i MATLAB skup programskih alata. Pomoću SIMULINK-a, koje je MATLAB-ovo grafičko programsko okruženje za modeliranje i simulaciju, dizajnira se blok model sustava. Zatim TargetLink za ulaz uzima taj model te koristeći MATLAB skriptu generira A2L datoteku. Takav pristup ima svojstvo podesivosti te se u slučaju izmjene modela A2L datoteka može s lakoćom ponovno generirati. Sličan pristup se koristi i u ovom radu gdje se podaci u SQL bazi podataka mogu po potrebi dodati, izmijeniti ili obrisati te će alat generirati nove A2L datoteke.

Još jedan od načina generiranja A2L datoteka prikazan je u radu [3]. U ovom radu autori su generirali A2L datoteke iz već postojećih A2L datoteka uz pomoć Vectorovih DaVinci alata i A2L nadograditelja (engl. *A2L updater*). Vectorovi DaVinci alati, između ostaloga, služe za stvaranje konfiguracija RTE (engl. *Run-Time Enviroment*) i BSW komponenata. Prilikom stvaranja konfiguracije DaVinci alati također stvaraju odgovarajuće A2L datoteke koje se izvode iz standardne AUTOSAR ECU konfiguracijske datoteke. Spajanjem tih A2L datoteka nastaje glavna A2L datoteku koja se šalje u A2L nadograditelj gdje se kombinira s povezničkom kartom (engl. *Linker map*) koja sadrži stvarne adrese na ECU-u. Ovim procesom nastaje konačna A2L datoteka koja se može koristiti u alatima za kalibraciju.

U radu [4] autori koriste lanac alata (engl. *tool chain*) koji kao početni ulaz uzima SIMULINK model ECU-a te generira A2L datoteke u dva koraka. U prvom koraku A2L datoteke su generirane iz SIMULINK modela koje se zatim šalju dobavljačima komponenti BSW-a. U skladu s tim datotekama, dobavljači zatim šalju svoje A2L datoteke te se one u drugom koraku kombiniraju s generiranim A2L datotekama. Rezultat toga su konačne A2L datoteke spremne za korištenje u kalibracijskim alatima. Osim SIMULINK-a, za modeliranje ECU-a moguće je koristiti i ASCET alat (engl. *Advanced Simulation and Control Engineering Tool*) razvijen on strane ETAS grupacije koji su koristili autori u [5]. Unutar ASCET alata se dizajnira blok model ECU-a te se, ovisno o potrebama, generira C programski kod za simulaciju ili C programski kod za ugradbeno sklopovlje. Taj generirani C programski kod se zatim šalje prevoditelju (engl. *compiler*) koji generira A2L i HEX datoteke. Nakon toga se navedene datoteke šalju u ETAS-ove eHooks-Prep i eHooks-Dev alate koji u kombinaciji sa specifičnim konfiguracijama projekta generiraju konačne A2L datoteke.

Ovaj diplomski rad je velikim dijelom temeljen na postojećem XCP generatoru napisanom u Python programskom jeziku. Postojeći generator čita *pickle* serijalizirani ARXML model (model.zip) te generira A2L datoteke. Ovo rješenje ima nedostatke poput ograničene prenosivosti i nemogućnosti skrivanja programskog koda od krajnjeg korisnika. Stoga je u ovom radu model.zip migriran na SQL bazu podataka, a XCP generator je napisan u C++ programskom jeziku. Realizacijom ovog rješenja su riješeni nedostatci postojećeg rješenja, međutim nastali su i novi nedostatci. Zbog korištenja vektora i globalnih varijabli novi generator prilikom rada zauzme veliku količinu radne memorije. To je dijelom smanjeno spremanjem određenih podataka u tekstualne datoteke koje se kasnije čitaju i zapisuju u A2L datoteke. Drugi nedostatak ovog rješenja je pojava dvostrukih unosa u A2L datotekama zbog preklapanja tipova podataka. Ovaj nedostatak

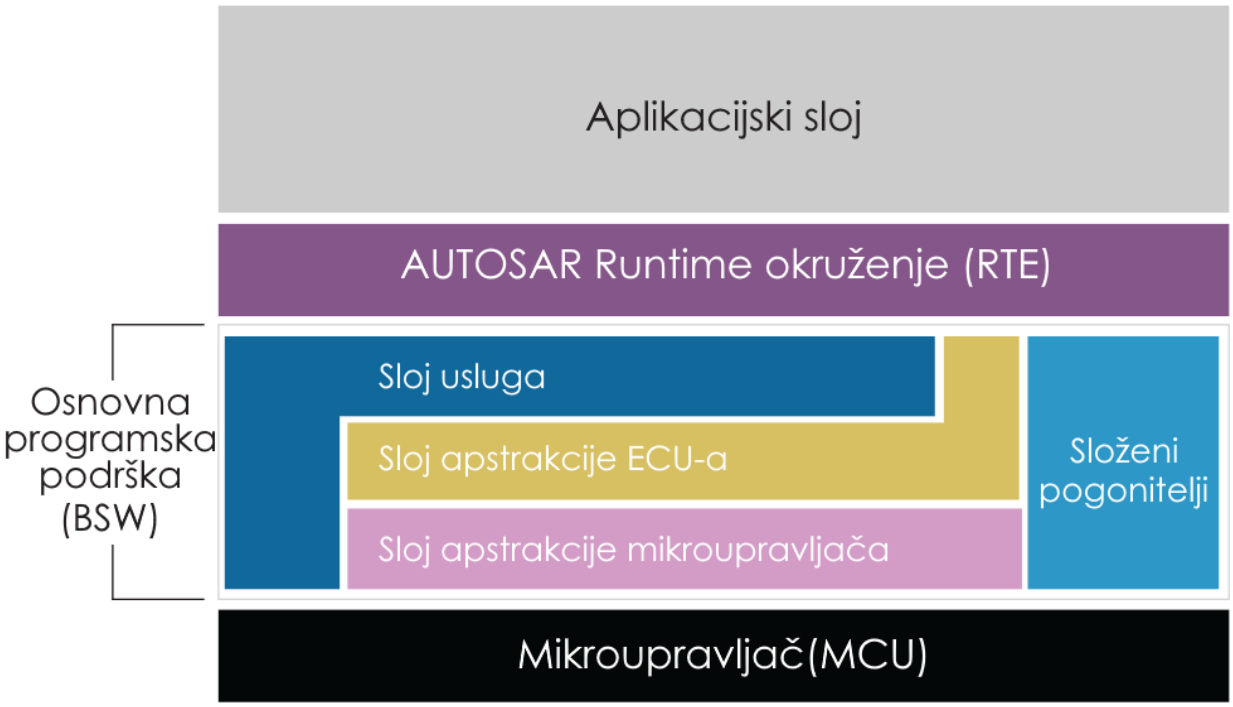
ne stvara probleme prilikom komunikacije s MotionWise pločom te su A2L datoteke funkcionalne unatoč dvostrukim unosima.

3. KORIŠTENI STANDARDI I TEHNOLOGIJE

Automobilska industrija se velikim dijelom oslanja na zajedničke standarde, platforme i alate kako bi se olakšao i ubrzao tehnološki razvoj te povećala proizvodnja novih industrijskih proizvoda. U ovom poglavlju su ukratko opisani standardi i alati korišteni u izradi alata za klasifikaciju parametara modela sustava.

3.1 AUTOSAR

AUTOSAR [6] se može se definirati kao zajednička platforma za cijelu automobilsku industriju koja je osmišljena kako bi povećala opseg primjene funkcionalnosti vozila bez utjecaja na trenutni operativni model. To je ujedno i svjetsko razvojno partnerstvo proizvođača vozila, dobavljača, pružatelja usluga i tvrtki iz automobilske elektronike, poluvodiča i softverske industrije. AUTOSAR arhitektura se sastoji od 3 glavna sloja: 1. Aplikacijskog sloja (engl. *Application layer*), 2. RTE sloja 3. BSW sloja. Arhitektura je prikazana na slici 3.1.



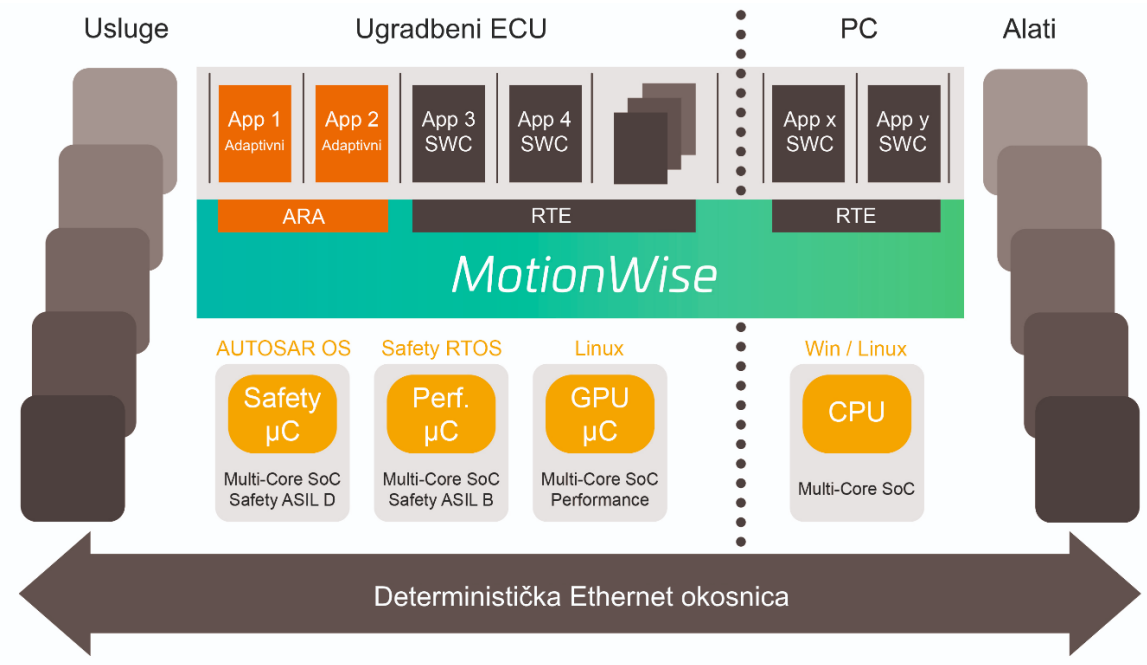
Slika 3.1. AUTOSAR arhitektura, preuzeto sa [7].

Aplikacijski sloj je prvi sloj arhitekture AUTOSAR i podržava implementaciju prilagođenih funkcionalnosti. Ovaj se sloj sastoji od specifičnih komponenata programske podrške i brojnih aplikacija koje izvode određene zadatke. RTE sloj je srednji sloj (engl. *Middlewar*e) programske arhitekture AUTOSAR između BSW-a i aplikacijskog sloja i pruža komunikacijske usluge za

aplikacijsku programsku podršku. Arhitektura BSW sloja se sastoji od velikog broja modula programske podrške strukturiranih u različite slojeve te je ta arhitektura zajednička svakom AUTOSAR ECU-u. To znači da dobavljač koji je dizajnirao BSW može podijeliti taj BSW s drugim dobavljačima koji rade na ECU-ovima motora, mjenjača itd.

3.2 MotionWise

MotionWise [8] je sigurnosna programska platforma namijenjena izvođenju i nadzoru autonomne vožnje, razvijena od strane TTTech grupacije. MotionWise platforma je namijenjena vozilima druge i treće razine autonomnosti, a predviđa se uporaba i u autonomnoj vožnji četvrte i pete razine. Platforma se sastoji od ugradbenog računalnog sustava i programske podrške koja sadrži skup alata potrebnih za nadzor elektroničkih upravljačkih jedinica automobila. Zadatak ugradbenog sustava je primanje informacija od senzora automobila, te na osnovu njih donošenje odluka o radu automobila u stvarnom vremenu. MotionWise platforma u potpunosti je kompatibilna s AUTOSAR standardom, te prema ISO 26262 ASIL D standardu osigurava nesmetan postupak integracije i sigurnosti. Na programskom dijelu MotionWise platforme izvode se AUTOSAR RTE i ARA (engl. *AUTOSAR Runtime for Adaptive Applications*) koji omogućavaju komunikaciju između elektroničkih upravljačkih jedinica. MotionWise platforma predstavlja srednji sloj između sklopovlja i operacijskog sustava vozila, kao što je prikazano na slici 3.2.

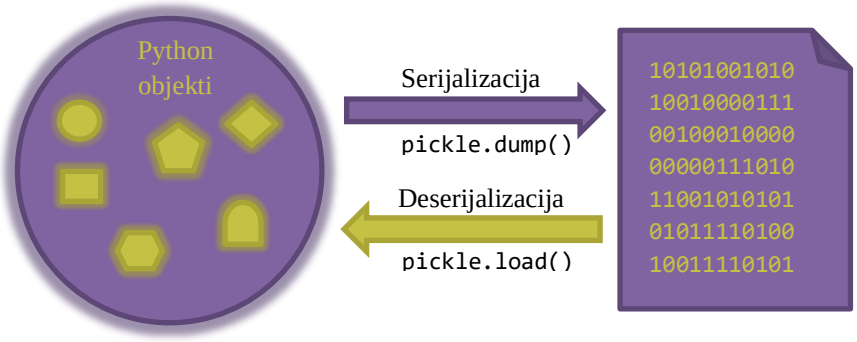


Slika 3.2. MotionWise platforma, preuzeto s [9].

Sklopovlje sustava sastoji se od ugradbenog računalnog sustava i elektroničkih upravljačkih jedinica vozila, dok se operacijski sustav sastoji od AUTOSAR OS-a, Safety RTOS-a (engl. *Real-time operating system*) i Linux operacijskog sustava. Model korišten u ovome radu namijenjen je MotionWise platformi.

3.3 Pickle modul

Pickle [10] modul implementira binarne protokole za serijaliziranje (engl. *pickling*) i uklanjanje serijalizacije (engl. *unpickling*) Python objektne strukture. *Pickling* je postupak kojim se Python hijerarhija objekata pretvara u tok bajtova (engl. *byte stream*), a *unpickling* je inverzna operacija, pri čemu se tok bajtova iz binarne datoteke pretvara natrag u hijerarhiju objekta. Dvije glavne funkcije koje se koriste su *pickle.dump()* i *pickle.load()*. Funkcija *pickle.dump()* zapisuje serijaliziranu reprezentaciju Python objekta u datoteku, dok funkcija *pickle.load()* čita reprezentaciju objekta iz datoteke te je pretvara nazad u hijerarhiju objekata. Shema rada se može vidjeti na slici 3.3., a primjer serijalizirane datoteke je prikazan na slici 3.4. U sklopu ovog diplomskog rada *pickle* modul je korišten za deserijaliziranje model.zip-a koji nastaje serijaliziranjem prilagođenog ECU modela u ARXML formatu.



Slika 3.3. *Pickle* modul shema.

```
 1 p1
 2 Example_filename
 3 p2
 4 ((S'Example_parent_1'
 5 p3
 6 S'Example_description_1'
 7 p4
 8 S'Example_child_1'
 9 p5
10 S'Example_child_of_child_1'
11 p6
12 S'Example_child_of_child_2'
13 p7
```

```
14 S'Example_child_2'
15 p8
16 S'Example_child_of_child_3'
17 p9
18 S'Example_child_of_child_4')
19 p10'
20 (S'Example_parent_2'
21 p11
22 S'Example_description_2
23 p12
24 S'Example_child_3'
25 p13
26 S'Example_child_3'
```

Slika 3.4 Primjer *pickle* serijalizirane datoteke.

3.4 A2L datoteka

Datoteka opisa A2L [11] pripada standardu ASAP2 koji definira ASAM, ranije poznat kao ASAP (njem. *Arbeitskreis zur Standardisierung von Applikationssystemen*). Sadrži sve informacije o relevantnim podatkovnim objektima u ECU-u kao što su karakteristike (parametri, karakteristične krivulje i mape), stvarne i virtualne mjerne varijable i ovisnosti o varijantama. Za svaki od ovih objekata su potrebne informacije, kao što su memorijska adresa, struktura memorije, vrsta podataka i pravila za njihovo pretvaranje u fizičke jedinice. Uz to, A2L sadrži i parametre za komunikaciju između alata i ECU-a. Primjer takvog alata je CANape koji je u ovom radu korišten zajedno s generiranim A2L datotekama čiji je primjer prikazan na slici 3.2.

```
/begin CHARACTERISTIC D_oZFDEV3_dn_ab_soll_lim.InitStatus "-"
    VALUE 0x3CB59 ui8 0 Pios_OInitStatus 0 15
    /begin IF_DATA CANAPE_EXT
        100
        LINK_MAP "D_oZFDEV3_dn_ab_soll_lim.InitStatus" 0x3CB59 0 0 0 1 0x87 0
        DISPLAY 0x0 0 15
    /end IF_DATA
    DISPLAY_IDENTIFIER D_o2FDEV3_dn_ab_soll_lim.InitStatus
/end CHARACTERISTIC

/begin CHARACTERISTIC D_oZFDEV3_dn_ab_soll_lim.idx_Filter "-"
    VALUE 0x3CB5D ui8 0 Pios_Filter 0 8
    /begin IF_DATA CANAPE_EXT
        100
        LINK_MAP "D_oZFDEV3_dn_ab_soll_lim.idx_Filter" 0x3CB5D 0 0 0 1 0x87 0
        DISPLAY 0x0 0 8
    /end IF_DATA
    DISPLAY_IDENTIFIER D_oZFDEV3_dn_ab_soll_lim.idx_Filter
/end CHARACTERISTIC

/begin CHARACTERISTIC D_oZFDEv3_dn_ab_soll_lim.uthvalue "-"
    VALUE 0x3C860 si32 0 CONV_DEFAULT 0 65535
    /begin IF_DATA CANAPE_EXT
        100
        LINK_MAP "D_oZFDEV3_dn_ab_soll_lim.uthValue" 0x3CB60 0 0 0 1 0xDF 0
        DISPLAY 0x0 0 65535
    /end IF_DATA
    DISPLAY_IDENTIFIER D_o2FDEV3_dn_ab_soll_limuthValue
/end CHARACTERISTIC
```

Slika 3.2. Primjer A2L datoteke.

CANape [12] je programski alat za optimizaciju parametrizacije (kalibracije) ECU-a. Služi za kalibriranje vrijednosti parametara i istovremeno pribavljanje mjernih signala tijekom rada sustava. Fizičko sučelje između CANape alata i ECU-a može se napraviti putem CAN (engl. *Controller Area Network*) sabirnice s CCP (engl. *CAN Calibration Protocol*) ili XCP (engl. *Universal Measurement and Calibration Protocol*) protokolima.

CCP [13] je protokol za kalibraciju i prikupljanje podataka iz ECU-a. U prošlosti su se koristila različita tehnička rješenja za razvoj, kalibraciju, proizvodnju i servis ECU sklopovlje i programske podrške. Cilj CCP-a je stvoriti zajednički alat za sve faze razvoja ECU-a koji je kompatibilan s različitim vrstama sklopovlja i programske podrške. CCP predstavlja aplikacijski sloj koji komunicira i izvodi čitanje i pisanje u memoriju ECU-a preko fizičkog sloja CAN-a.

XCP [14] je protokol nastao kao daljnji razvoj CCP-a. Podržava više prijenosnih medija, ali nije kompatibilan s CCP-om. Omogućuje pristup čitanju i pisanju varijabli i memorijskog sadržaja sustava mikroupravljača tijekom izvođenja programa. Cijeli skupovi podataka mogu se pribaviti ili simulirati sinkronizirano s događajima koje pokreću tajmeri ili radni uvjeti. Uz to, XCP također podržava programiranje brze memorije.

4. MIGRIRANJE MODELA I IZRADA XCP GENERATORA

Postojeći XCP generator je napisan u Python programskog jeziku te za ulaz koristi *pickle* serijalizirani model koji zatim deserijalizira i generira A2L datoteke. Model (*model.zip*) nastaje iz ARXML datoteka koje model generator prevodi u oblik upotrebljiv za TTTech-ove alate, što je u slučaju ovog diplomskog rada u oblik pogodan za XCP generator. Nedostaci ovog rješenja su navedeni u drugom poglavlju. Uzimajući u obzir te nedostatke kreirane su slijedeće specifikacije na temelju kojih je izrađeno novo rješenje:

1. Migrirati model na SQL bazu podataka
2. Zamijeniti ulazne parametre XCP generatora napisanog u Python programskom jeziku
3. Zamijeniti TTTech biblioteke MWC sa SQLite bibliotekom
4. Izraditi XCP generator koristeći C++ programski jezik

Da bi se model migrirao na SQL bazu podataka potrebno je izraditi model parser koji čita prilagođeni ARXML ECU model te ga zapisuje u SQL bazu podataka. Početni i željeni tok rada programa prikazani su na slikama 4.1. i 4.2.

Slika 4.1. Početni tok rada programa.

Slika 4.2. Željeni tok rada programa.

4.1 Prevođenje modela u SQL bazu podataka

4.1.1 Definiranje strukture baze podataka

Prvi korak u realizaciji alata za klasifikaciju je izrada SQL baze podataka. Struktura baze i izgled tablica su definirane prema klasama i listama u postojećem generatoru. Definirano je 7

tablica, od kojih je prva tablica (`ECU_1`) sa SWC-ovima i pripadajućim prototipovima. Tablica se koristi pri određivanju koji se SWC-ovi i njihovi KVS-ovi (engl. *Key-Value-Storage*) trebaju nalaziti u glavnoj A2L datoteci te pri generiranju A2L datoteka za pojedine SWC-ove. *Key-Value-Storage* je paradigma za pohranu podataka dizajnirana za spremanje, dohvaćanje i upravljanje asocijativnim nizovima i strukturama podataka koja je danas poznatija kao rječnik ili *hash* tablica. Rječnici sadrže zbirku predmeta ili zapisa, koji u sebi imaju više različitih polja koja sadrže podatke. Ti se zapisi pohranjuju i dohvaćaju pomoću ključa koji jedinstveno identificira zapis i koristi se za pronalaženje podataka u bazi podataka.

U drugoj tablici (`ECU_2`) se nalaze prototipovi portova svih SWC-ova te informacija koriste li se ti portovi za slanje ili prijem. Treća tablica (`Software_Component`) sadrži podatkovne elemente svih portova, njihov opis, tip poruke i tip podataka. U četvrtoj i najvećoj tablici (`Elements`) se nalaze imena svih KVS-ova te također njihov opis, tip poruke i tip podataka. Treća i četvrta tablica se koriste u izradi KVS unosa u A2L datoteke. Peta tablica (`Msg_Type_P`) sadrži tipove podataka, njihove metode računanja, kategorije, gornje i donje granice, veličinu pri alociranju te ostale podatke.

KVS-ovi su podijeljeni prema 2 tipa poruke (tipova podataka): kompozitne i primitivne. Tip podataka je skup vrijednosti koje karakteriziraju njihova svojstva i operacije nad tim vrijednostima. Kompozitni tipovi podataka „*niz*“ i „*zapis*“ omogućuju izgradnju novih tipova podataka. Moguće je kombinirati ovo dvoje, tako da niz može biti element zapisa, a na isti način zapis može biti elementa niza. Moguće je i ugnježđivanje ovih vrsta podataka. Niz može sadržavati N elemenata koji su svi istog tipa. Kada se želi pristupiti elementu niza unutar SWC-a koristi se indeks elementa koji se kreće od 0 do N-1. Zapis (struktura) je opisan kao neprazan skup objekata od kojih svaki ima jedinstveni identifikator s obzirom na tip zapisa i tip podataka. Ime svakog objekta mora biti jedinstveno u ovom skupu objekata, a tip podataka može biti bilo koji tip podataka unutar AUTOSAR-a.

Primitivni tipovi podataka ne mogu se rastaviti u druge tipove podataka te se dijele na: *Integer, Float, Char, Boolean, String* i *Opaque.* Tipovi podataka su definirani u ARXML datotekama te se primjer definicije primitivnog tipa podatka može vidjeti na slici 4.3. Šesta tablica (`Msg_Type_A`) sadrži tipove podataka koje koriste nizovi i tipove na kojima se temelje. Primjerice, tip podatka „*Array_uint16*“ se temelji na „*uint16*“ tipu podatka. U posljednjoj tablici (`Enumerations`) se nalaze posebni tipovi podataka koji služe za nabrajanje.

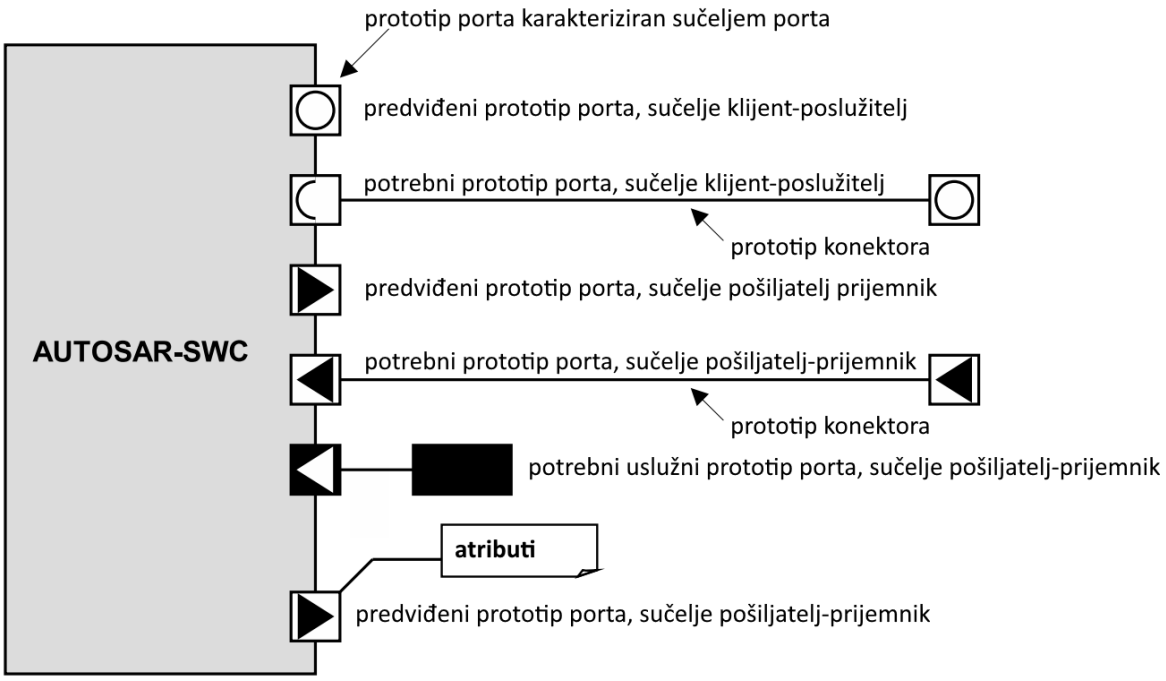
```
<AR-PACKAGE>
    <SHORT-NAME>ApplicationDataTypes</SHORT-NAME>
    <ELEMENTS>
        <APPLICATION-PRIMITIVE-DATA-TYPE>
            <SHORT-NAME>MyApplicationStringType</SHORT-NAME>
            <CATEGORY>STRING</CATEGORY>
            <SW-DATA-DEF-PROPS>
                <SW-DATA-DEF-PROPS-VARIANTS>
                    <SW-DATA-DEF-PROPS-CONDITIONAL>
                        <SW-TEXT-PROPS>
                            <ARRAY-SIZE-SEMANTICS>VARIABLE-SIZE</ARRAY-SIZE-SEMANTICS>
                            <SW-MAX-TEXT-SIZE>50</SW-MAX-TEXT-SIZE>
                            <BASE-TYPE-REF DEST="SW-BASE-TYPE" BASE="default">BaseTypes/
                                MyTextBaseType</BASE-TYPE-REF>
                        </SW-TEXT-PROPS>
                        <INVALID-VALUE>
                            <APPLICATION-VALUE-SPECIFICATION>
                                <CATEGORY>STRING</CATEGORY>
                                <SW-VALUE-CONT>
                                    <SW-VALUES-PHYS>
                                        <VT>inv</VT>
                                    </SW-VALUES-PHYS>
                                </SW-VALUE-CONT>
                            </APPLICATION-VALUE-SPECIFICATION>
                        </INVALID-VALUE>
                        <SW-RECORD-LAYOUT-REF DEST="SW-RECORD-LAYOUT" BASE="default">
                            RecordLayouts/StringDescriptor</SW-RECORD-LAYOUT-REF>
                    </SW-DATA-DEF-PROPS-CONDITIONAL>
                </SW-DATA-DEF-PROPS-VARIANTS>
            </SW-DATA-DEF-PROPS>
        </APPLICATION-PRIMITIVE-DATA-TYPE>
    </ELEMENTS>
</AR-PACKAGE>
```

Slika 4.3. Definicija primitivnog tipa podataka.

4.1.2 Parsiranje modela i zapisivanje u bazu podataka

Model je serijaliziran *pickle* modulom, zato ga je potrebno deserijalizirati da bi se pristupilo podacima. Postojeći generator već obavlja tu funkciju, stoga je zbog jednostavnosti korišten kao osnovica u pisanju model parsera. Prvo je potrebno napraviti parsiranje i zapisivanje u bazu podataka imena KVS-ova koja imaju oblik definiran u AUTOSAR SWC predlošku (engl. *Software Component Template*). SWC predložak [15] je stvoren kako bi se podržala ponovna upotrebljivost na razini aplikacijske programske podrške tj. kako bi bila moguća ponovna upotreba postojećih artefakata za stvaranje daljnjih elemenata modela, umjesto da se prisiljava izrada svakog pojedinog elementa od nule. Između ostalog, ovaj predložak omogućuje stvaranje hijerarhijskih struktura SWC-ova proizvoljne složenosti. Međutim, stvaranje hijerarhijskih struktura samo po sebi nema utjecaja na ponašanje cjelokupnog sustava već je stvarno ponašanje definirano unutar pojedinih SWC-ova. Aplikacijska programska podrška unutar AUTOSAR-a je organizirana u samostalne

SWC-ove koji inkapsuliraju različite funkcionalnosti i ponašanja. SWC-ovi posjeduju prototipove portova (engl. *Port Prototype*) koji im služe za međusobnu komunikaciju i interakciju. Prototipovi portova komuniciraju preko sučelja za slanje i prijem (engl. *Sender-Receiver Interface*) koje definira podatkovne elemente korištene u komunikaciji. Primjer SWC-a i njegovih prototipova portova prikazan je na slici 4.4.

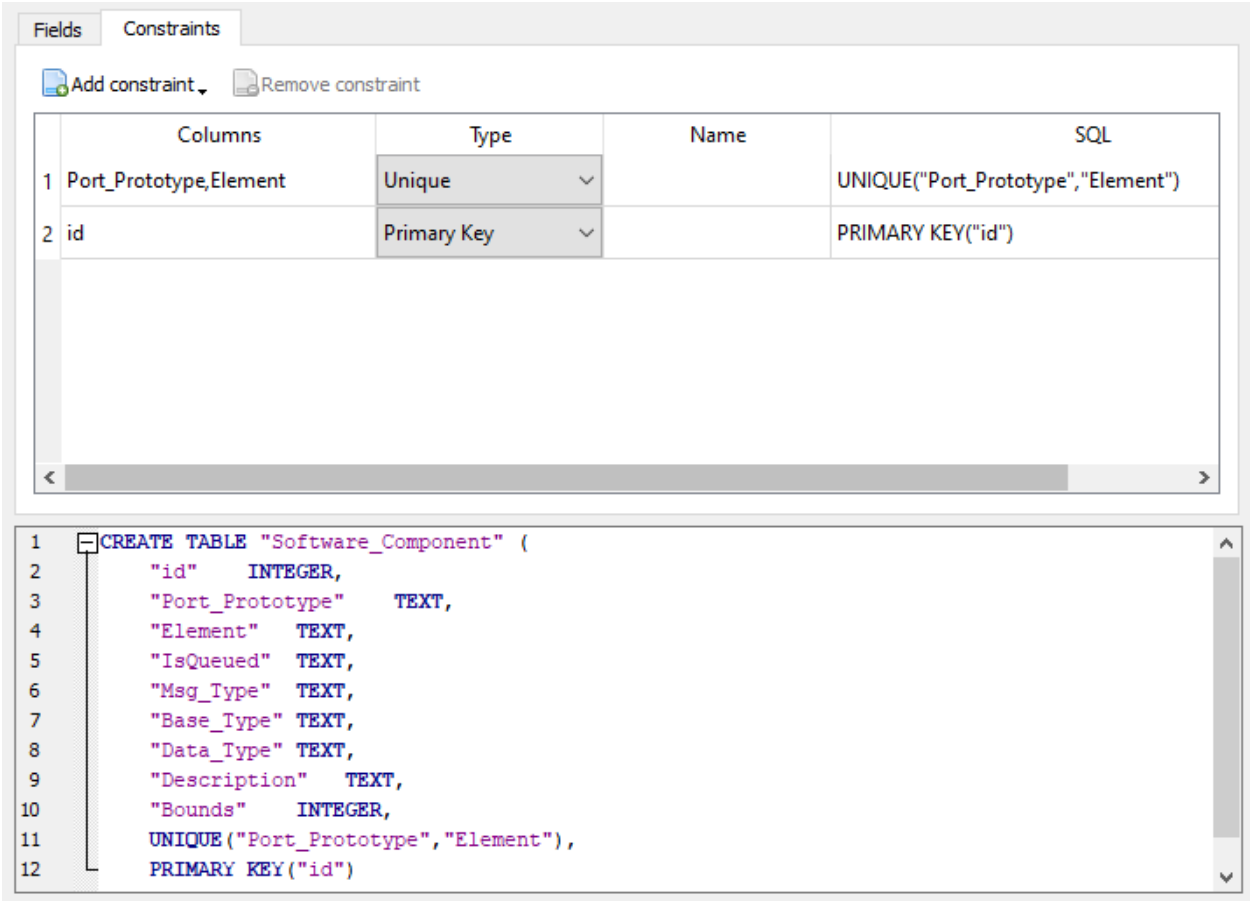


Slika 4.4. AUTOSAR SWC shema, preuzeto s [15]

Prema tome su imena KVS unosa zapisana u obliku „`SWC.Port_prototip.Element.Podelement_1.Podelement_2.Podelement_3`“ pri čemu je broj podelemenata promjenjiv pa u imenu mogu biti od 1 do 3 podelementa. Prva tri dijela imena („`SWC.Port_prototip.Element`“) su složena unutar funkcije za stvaranje KVS ulaza (`KVS_entry_creator`), koja zatim poziva jednu od tri funkcije za rukovanje tipovima porukama (`structure_msg_handler`, `array_msg_handler`, `primitive_msg_handler`). Ta funkcija dodaje ime podelementa („`Podelement`“) u ime KVS ulaza te ona također poziva jednu od funkcija za rukovanje tipovima porukama. Svaki podelement može imati svoj podelement koji se dodaje dok se ne pozove funkcija za rukovanje primitivnim tipovima porukama (`primitive_msg_handler`) koja dovršava stvaranje KVS unosa i sprema ga u listu. Prema tome je potrebno dizajnirati tablicu u koju se pohranjuju veze između prototipova portova, elemenata i podelemenata. Različiti prototipovi portova i elementi mogu imati

podelemente istog imena pa su zbog toga svi elementi i podelementi u tablici povezani s dva do četiri roditeljska elementa.

Unutar postojećeg generatora je integriran model parser koji upisuje podatke u bazu podataka na svim mjestima gdje se koristi model i gdje će se koristiti u budućem generatoru. Za upisivanje se koristi „*INSERT OR IGNORE*“ SQL naredba koja dodaje retke u odabranu postojeću tablicu unutar baze u slučaju da unos ne krši ograničenje. Svaka tablica unutar baze podataka sadrži parove ili grupe stupaca s „*UNIQUE*“ ograničenjem, što znači da su redci jedinstveni tj. ne mogu postojati redci s istim parovima ili grupama stupaca unutar tablice. Primjer ograničenja se može vidjeti na slici 4.5. To ograničenje se koristi zbog toga što generator više puta prolazi kroz iste elemente te bi bez ograničenja unio iste retke više puta, što bi stvorilo zalihost pri generiranju A2L datoteka. Također, ovo ograničenje sprječava unošenje istih redaka pri ponovnom pokretanju parsera radi ažuriranja baze podataka. Da ovog ograničenja nema, parser bi, umjesto dodavanja samo novih redaka, unio cijelu tablicu ponovo.

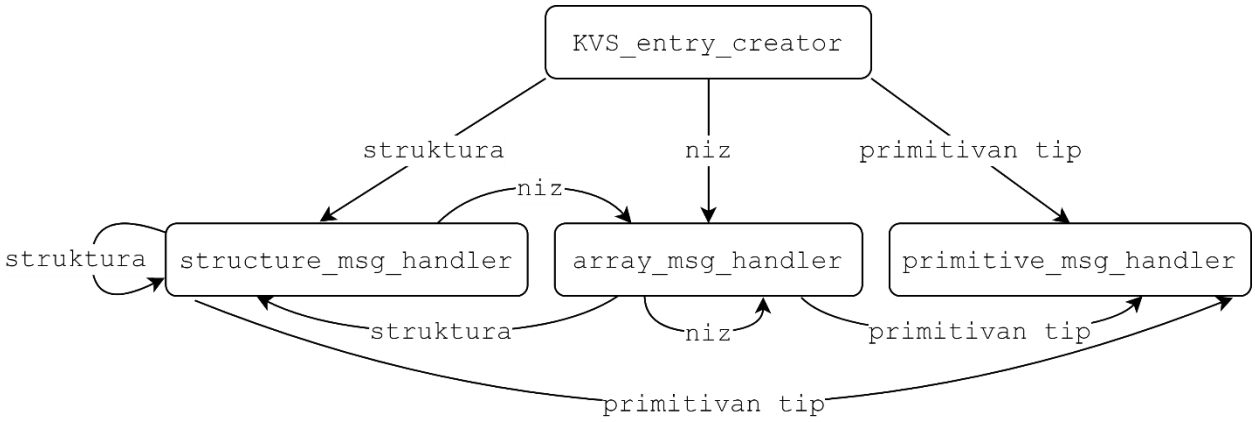


Slika 4.5. Tablica s prikazom ograničenja.

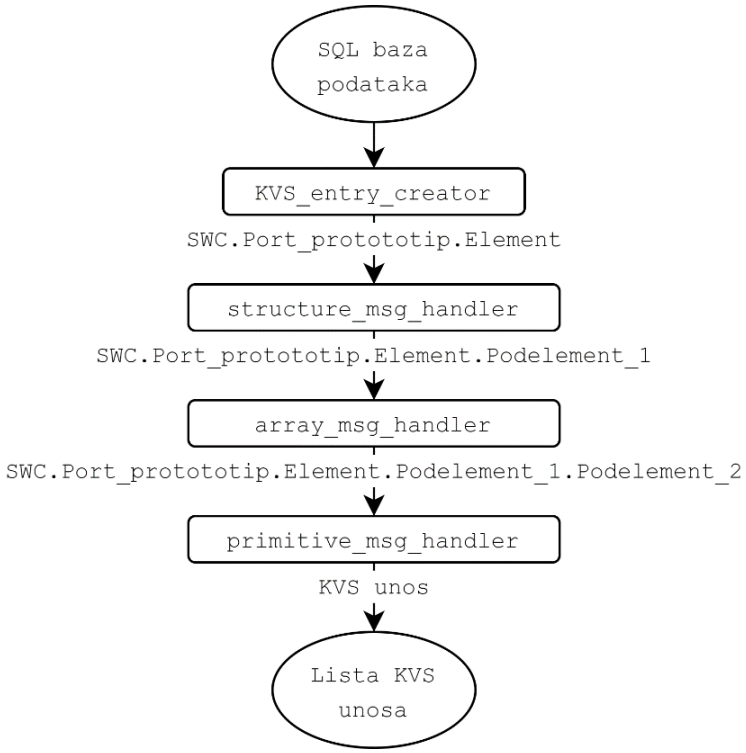
4.2 Zamjena ulaznih parametara XCP generatora

Drugi korak u realizaciji alata za klasifikaciju je prilagodba postojećeg generatora i zamjena ulaznih parametara. Postojeći generator koristi *pickle* modul i serijalizirani model komprimiran u model.zip koje je potrebno zamijeniti SQLite bibliotekom i SQL bazom podataka. Također, u postojećem generatoru su korištene klase, druge biblioteke i datoteke vezane uz *pickle* modul koje se moraju ukloniti ili izmijeniti kako bi bile kompatibilne sa SQL bazom podataka. Za dohvaćanje podataka iz baze se koristi „*SELECT*“ naredba koja vraća određene atribute redaka iz jedne ili više tablica. Kako bi se pristupilo željenim atributima, naredba može imati jedan ili više uvjeta. Na primjer, za pristup prototipu porta unutar tablice s prototipovima i SWC-ovima je potreban samo jedan uvjet (ime SWC-a), dok je za pristup elementima i podelementima unutar prototipa porta potrebno četiri uvjeta (dva prethodna elementa, prototip porta i SWC).

Prva velika izmjena u glavnoj datoteci se nalazi unutar funkcije za stvaranje KVS unosa. SELECT naredbom se pristupa SWC-ovima i njihovim portovima unutar tablica `ECU_1` i `ECU_2` koji se dodaju u listu traženih KVS-ova ako zadovoljavaju uvjete. Ta se lista koristi pri stvaranju KVS unosa. Pristupa se elementima iz `Software_Component` tablice koji pripadaju portovima unutar liste. Od tih podataka se složi ime KVS-a u formi „`SWC.Port_prototip.Element`“ i šalje jednoj od tri funkcije za rukovanje tipovima poruka gdje se nalaze druge velike izmjene. Unutar tih funkcija se pristupa podelementima iz `Elements` tablice koji pripadaju poslanom elementu ili podelementu te ih se dodaje u ime KVS-a. Te funkcije zatim također pozivaju jednu od funkcija za rukovanje tipovima poruka. Funkcija može pozvati i samu sebe ako podelement ima isti tip poruke kao i element. Taj proces se ponavlja dok se ne pozove funkcija za rukovanje primitivnim tipovima podataka. Unutar te funkcije se dodaje posljednji podelement u ime KVS-a. Nakon toga se stvaraju pripadajuća metoda računanja iz tablice `Msg_Type_P` i ključna riječ za konverziju metode računanja. Zatim se dovršava KVS unos te se sve dodaje u odgovarajuće liste. Cijeli postupak pozivanja funkcija i stvaranja KVS unosa prikazan je na slikama 4.6 i 4.7. Kada prođe kroz sve SWC-ove i pripadajuće elemente, `KVS_entry_creator` vraća sve liste koje se čitaju u `main` funkciji i zapisuju u A2L datoteke prema određenom predlošku.



Slika 4.6. Dijagram pozivanja funkcija.



Slika 4.7 Dijagram stvaranja KVS unosa.

Problem koji je nastao zbog ovih izmjena je postojanje dvostrukih KVS unosa. Prilikom pozivanja `array_msg_handler` funkcije na ime podelementa se dodaje indeks („`[i]`“) koji je u tom obliku parasiran i zapisan u tablici `Elements` kao niz. Problem nastaje ukoliko je podelement koji se nalazi u nizu primitivnog tipa. Nakon zapisivanja u tablicu se poziva i funkcija `primitive_msg_handler` koja ovaj put upisuje taj element bez indeksa i kao primitivan tip. Zbog toga se prilikom generiranja A2L datoteka stvaraju dvostruki KVS unosi ukoliko sadrže te podelemente. Međutim, ti dvostruki unosi ne javljaju nikakve greške te se generirane A2L datoteke uspješno pokreću na ploči. Primjeri ekvivalentnih unosa u ARXML datoteci, *Elements* SQL tablici i A2L datoteci prikazani su na slikama 4.8., 4.9. i 4.10.

```
<AR-PACKAGE>
      <SHORT-NAME>ComponentTypes</SHORT-NAME>
      <ELEMENTS>
        <APPLICATION-SW-COMPONENT-TYPE UUID="A2BEA3A8-D96D-698F-42A2-7FAC382CC77E">
          <SHORT-NAME>SWC_Example</SHORT-NAME>
          <PORTS>
            <P-PORT-PROTOTYPE UUID="DF6DD954-57EE-6492-D2EF-E3A9347E66C8">
              <SHORT-NAME>PpExample</SHORT-NAME>
              <PROVIDED-COM-SPECS>
                <NONQUEUED-SENDER-COM-SPEC>
                  <DATA-ELEMENT-REF DEST="VARIABLE-DATA-
PROTOTYPE">/PortInterfaces/PortInterface_Example/DeExample</DATA-ELEMENT-REF>
                  <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                  <INIT-VALUE>
                    <CONSTANT-REFERENCE>
                      <CONSTANT-REF DEST="CONSTANT-
SPECIFICATION">/Constants/Constant_Example</CONSTANT-REF>
                    </CONSTANT-REFERENCE>
                  </INIT-VALUE>
```

Slika 4.8. Primjer unosa u ARXML datoteci

id	Fifth Element	Fourth Element	Third Element	Second Element	First Element	Msg Type	Data Type
4017	-	SWC_Example	PpExample	DeExample	SubDeExample	Msg_Type_P	DtExample

Slika 4.9. Primjer unosa u Elements tablici

```
/begin CHARACTERISTIC
SWC_Example.PortPrototype_Example.DataElement_Example.SubDataElement_Example
      VALUE 0x5A72E __UBYTE_S 0
SWC_Example.PortPrototype_Example.DataElement_Example.SubDataElement_Example.DtExample.
CONVERSION 0 255
      ECU_ADDRESS_EXTENSION 0x50
      EXTENDED_LIMITS 0 255
/end CHARACTERISTIC
```

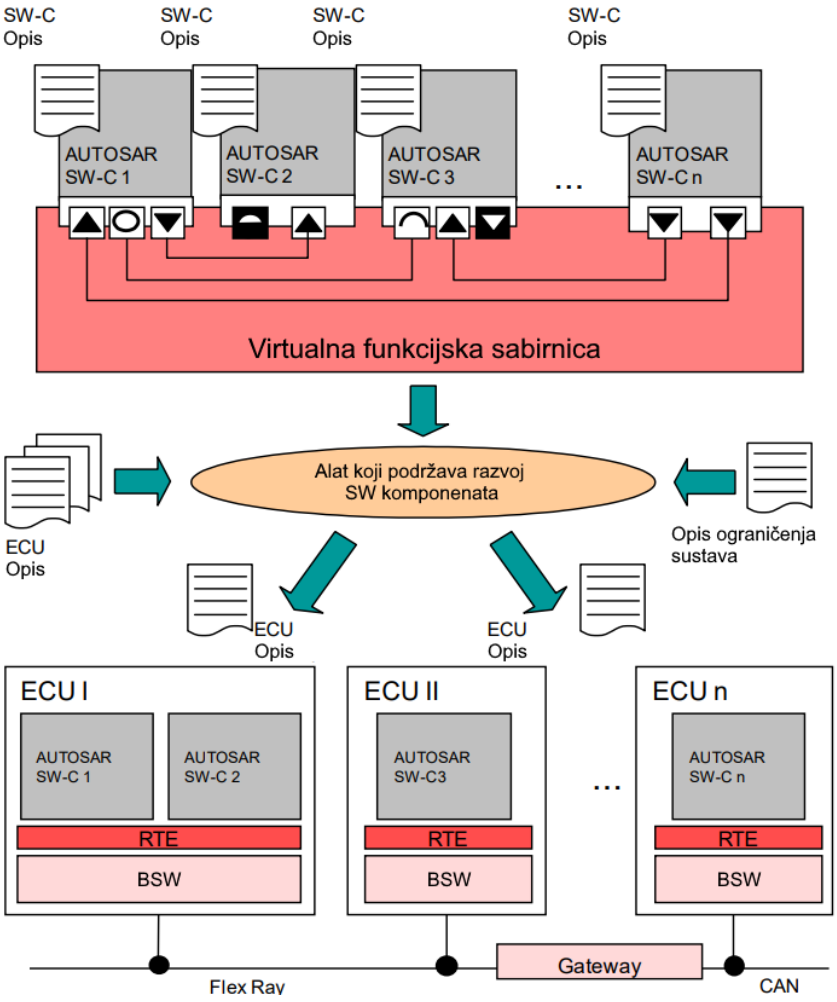
Slika 4.10. Primjer unosa u A2L datoteci

4.3 Izrada XCP generatora koristeći C++

Treći korak u realizaciji alata za klasifikaciju je prevođenje izmijenjenog generatora u C++ programski jezik. Postojeći generator je potrebno podijeliti po funkcijama i klasama te svaku od njih proučiti i pronaći optimalan pristup za prevođenje. Prva funkcija koju je potrebno prevesti je `main` funkcija. Ona poziva funkciju za parsiranje konfiguracijske XML datoteke, učitava RTE funkcije poziva funkciju `KVS_entry_creator` te na kraju generira A2L datoteke. U konfiguracijskoj XML datoteci se nalaze informacije za generiranje A2L datoteka od kojih su najbitnije: putanja i ime predloška A2L datoteke, imena ciljanih *hostova* (*SafetyHost* i

PerformanceHost), ime projekta, proširenje ECU adresa i ime korištene tehnologije transportnog sloja (npr. *Ethernet*). Predložak A2L datoteke se sastoji od zaglavlja, podnožja te mjesta između njih rezerviranog za generirane KVS unose, njihove metode računanja i tablice za konverziju metoda računanja. Sve generirane A2L datoteke imaju isto zaglavlje i podnožje pri čemu je samo generiranje olakšano te je proces upisivanja unosa između njih automatiziran.

Potom se učitavaju RTE funkcije parsiranjem C++ datoteke zaglavlja koja je stvorena pomoću RTE generatora. RTE generator [16] je jedan od skupa alata koji stvaraju AUTOSAR virtualnu funkcijsku sabirnicu (engl. *Virtual Functional Bus* – VFB) za ECU na temelju informacija iz opisa konfiguracije ECU-a. Virtualna funkcijska sabirnica [17] je komunikacijski mehanizam koji omogućuje interakciju između SWC-ova. U koraku dizajniranja sustava nazvanom „Konfiguriranje sustava“ (engl. *Configure System*) SWC-ovi se preslikavaju na ECU-ove. Na taj način se virtualne veze između SWC-ova preslikavaju na lokalne veze unutar jednog ECU-a ili na komunikacijske mehanizme (npr. na CAN ili FlexRay okvire). U takvom sustavu se ECU-ovi mogu pojedinačno konfigurirati. Proces konfiguriranja sustava prikazan je na slici 4.11.



Slika 4.11. Proces konfiguriranja sustava, preuzeto sa [17].

Generator RTE odgovoran je za stvaranje funkcija API-ja koje povezuju SWC-ove s operacijskim sustavom i upravljaju međusobnom komunikacijom između SWC-ova te komunikacijom između SWC-ova i BSW modula. RTE funkcije služe za kopiranje kalibracijskih parametara iz programske i brze memorije u radnu memorije te za korištenje metode dvostrukih pokazivača. Metoda dvostrukih pokazivača je dodjeljivanje adrese jednog pokazivača drugom te se koristi kako bi se pristupilo vrijednosti varijable izvan poziva funkcije.

RTE funkcije se dijele na funkcije za čitanje i pisanje te postoje za većinu podatkovnih elemenata. Prilikom parsiranja RTE datoteke zaglavlja funkcije se zapisuju u dvije liste ovisno o tome služe li za čitanje i pisanje. Tijekom generiranja KVS unosa poziva se funkcija koja uzima ime podatkovnog elementa, prolazi kroz liste funkcija, izdvaja one koje sadrže podatkovni element te ih pridodaje tom elementu. U slučaju da neki element nema određenu funkciju, na mjestu funkcije mu se upisuje „*NULL_PTR*“.

Zatim se prevodi funkcija `KVS_entry_creator` gdje se pojavljuje prva „*SELECT*“ naredba. U Python generatoru pri dohvaćanju vrijednosti iz baze podataka „*SELECT*“ naredba sprema sve vrijednosti u liste koje imaju promjenjivu veličinu ovisno o tome koliko elemenata sadrže. Da bi se te vrijednosti na isti način mogle spremiti u C++-u potrebno je koristiti vektore, koji, za razliku od nizova, imaju promjenjivu veličinu, ali zato zauzimaju više memorije. Sljedeće funkcije koje je potrebno prevesti su funkcije za rukovanje tipovima poruka. Zbog načina na koji se pozivaju funkcije i njihovog međusobnog ulančavanja potrebno je koristiti globalne vektore koji zauzimaju još više memorije. To stvara problem pri generiranju jer zbog velikog broja KVS unosa generator zauzme svu radnu memoriju te program ne može nastaviti s radom. Da bi se to izbjeglo, dovršeni KVS unosi, njihove metode računanja i tablice za konverziju metoda računanja se zapisuju u tekstualne datoteke koje `main` funkcija čita te generira A2L datoteke. Ostale funkcije koje se moraju prevesti su pomoćne funkcije koje učitavaju predloške, konfiguracijske datoteke, određuju oblik KVS unosa i standardne Python funkcije za koje ne postoje ekvivalentne funkcije u C++-u.

Nedostaci ovog generatora su znatno veće vrijeme izvođenja programa, veća potrošnja radne memorije te dvostruki unosi koji se pojavljuju i kod izmijenjenog Python generatora. Ovi nedostaci bi se mogli ukloniti tako da se umjesto izravnog prevođenja funkcija i klasa, one prilagode i optimiziraju C++-ovom načinu spremanja i pristupanju podataka.

5. ZAKLJUČAK

Povećanjem broja ECU-a u automobilima dolazi do veće razine kompleksnosti sustava. Ta kompleksnost sa sobom povlači i složeniju ECU programsku podršku koja se sastoji od brojnih komponenata programske podrške. Pri stvaranju ECU programske podrške, razmjenjuju se datoteke i meta-informacije između nekoliko strana i u različitim kontekstima. Na primjer, datoteke i meta-informacije razmjenjuju se između arhitekta sustava i programera, između alata za dizajn sustava i alata za modeliranje ponašanja, između različitih projekata u istom ili različitim alatima za dizajn sustava, između različitih projekata u istom ili različitim alatima za modeliranje ponašanja. Stoga je potrebno koristiti zajednički predložak. U ovom radu je izrađen XCP generator koji koristi *Software Component* predložak.

Generator je baziran na već postojećem rješenju te je njegova izrada podijeljena u 3 koraka. U prvom koraku je izrađena SQL baza podataka temeljena na *pickle* serijaliziranom modelu. Baza je popunjena pomoću model parsera napisanog unutar postojećeg rješenja zbog olakšanog pristupa serijaliziranom modelu. U drugom koraku se SQL bazom zamijenio model.zip kao ulazni parametar postojećeg XCP generatora u Python programskom jeziku. U ovom koraku su izmijenjene funkcije i klase postojećeg generatora kako bi se prilagodile SQL bazi podataka. A2L datoteke su uspješno generirane i bez problema pokrenute na MotionWise ploči, međutim pojavljuju se dvostruki unosi u slučajevima kada se u imenu KVS ulaza nalazi niz. Dvostruki unosi nisu stvarali probleme pri testiranju te su nakon neuspjelih pokušaja uklanjanja zanemareni. U trećem koraku je izrađen XCP generator u C++ programskom jeziku. A2L datoteke generirane ovim rješenjem su identične datotekama iz drugog koraka, što znači da su uspješno pokrenute na ploči, ali još uvijek imaju dvostruke unose. Drugi nedostatak ovog rješenja je preveliko zauzeće radne memorije pri generiranju koje je dijelom smanjeno zapisivanjem nekih podatka u tekstualne datoteke.

Izrada alata za klasifikaciju parametara modela sustava je uspješno odrađena te su generirane A2L datoteke, izuzev dvostrukih unosa, identične onima iz postojećeg XCP generatora. Nedostatci postoje i ima mjesta za napredak te bi se u budućnosti moglo optimizirati korištenje radne memorije i ukloniti dvostruki unosi. Kako je ovaj diplomski rad ujedno bio i studija izvodljivosti dokazano je da je ovaj koncept izvodljiv.

LITERATURA

[1] K. Kianer, U. Finis, T. Pusch, i S. Slopianka, „Open Interfaces for Bridging the Steps in the Chain of a Totally Simulation-based Software Development“, ožu. 2001.

[2] L. Juhasz, A. Samiee, i W. Engelbrecht, „XCP Service Integration for Model-Based, Automatic Production Code Generation“, u IEEE EUROCON 2019 -18th International Conference on Smart Technologies, Novi Sad, Serbia, srp. 2019, str. 1–6

[3] R. E. Lotoczky i M. Schwager, „New Methods of Debugging and Testing Improve the Software Quality of AUTOSAR ECUs“, *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.*, sv. 6, izd. 1, str. 180–185, tra. 2013

[4] A. Arenz i S. Potrykus, „Model-based algorithm development: Automated from the idea to the production control unit“, *ATZ Worldw*, sv. 112, izd. 1, str. 18–22, sij. 2010

[5] C. Luckeneder, H. Kaindl, i M. Korinek, „Automated Unit Testing in Model-based Embedded Software Development“:, u *Proceedings of the 12th International Conference on Software Technologies*, Madrid, Spain, 2017, str. 427–434

[6] AUTOSAR, „AUTOSAR Introduction“, 2020. [Mrežno]. Dostupno na: https://www.autosar.org/fileadmin/ABOUT/AUTOSAR_EXP_Introduction102020.pdf, [Pristup ostvaren: 04.12.2020].

[7] Anshul Saxena, „Understanding AUTOSAR and its Applications in the Automotive Industry“, 2020. [Mrežno]. Dostupno na: https://www.einfochips.com/blog/autosar-in-automotive-industry/, [Pristup ostvaren: 04.12.2020].

[8] TTTech Auto, „MotionWise“, [Mrežno]. Dostupno na: https://www.tttech-auto.com/products/automated-driving/motionwise/, [Pristup ostvaren: 04.12.2020].

[9] TTTech Auto, „TTTech Introduces Modular Platform for Automated Driving“, [Mrežno]. Dostupno na: https://www.tttech-auto.com/tttech-introduces-modular-platform-for-automated-driving/, [Pristup ostvaren: 04.12.2020].

[10] Python, „pickle — Python object serialization“, [Mrežno]. Dostupno na: https://docs.python.org/3/library/pickle.html/, [Pristup ostvaren: 04.12.2020].

[11] Vector, „Data Description for ECU Calibration“, [Mrežno]. Dostupno na: https://www.vector.com/int/en/products/application-areas/ecu-calibration/data-description/, [Pristup ostvaren: 04.12.2020].

[12] Vector, „ECU Calibration with CANape“, [Mrežno]. Dostupno na: https://www.vector.com/int/en/products/products-a-z/software/canape/, [Pristup ostvaren: 04.12.2020].

[13] Kvaser, „CCP/XCP“, [Mrežno]. Dostupno na: https://www.kvaser.com/about-can/higher-layer-protocols/ccpxcp/, [Pristup ostvaren: 04.12.2020].

[14] Vector, „Measurement and Calibration Protocol XCP – Fundamentals“, 2020. [Mrežno]. Dostupno na: https://www.vector.com/int/en/know-how/technologies/protocols/xcp-measurement-and-calibration-protocol/, [Pristup ostvaren: 04.12.2020].

[15] AUTOSAR, „Software Component Template“, 2011. [Mrežno]. Dostupno na: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-2/AUTOSAR_TPS_SoftwareComponentTemplate.pdf, [Pristup ostvaren: 04.12.2020].

[16] AUTOSAR, „Specification of RTE“, 2011. [Mrežno]. Dostupno na: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-0/AUTOSAR_SWS_RTE.pdf, [Pristup ostvaren: 04.12.2020].

[17] AUTOSAR, „Virtual Functional Bus“, 2011. [Mrežno]. Dostupno na: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_VFB.pdf, [Pristup ostvaren: 04.12.2020].

SAŽETAK

Tehnološkim napretkom automobilske industrije elektronika u automobilu postaje sve značajnija, a time se povećava i broj elektroničkih upravljačkih jedinica. A2L datoteke sadrže relevantne informacije o objektima ECU-a kao što su karakteristike i mjerne varijable. U sklopu ovog diplomskog rada je bilo potrebno izraditi XCP generator u C++-u baziran na postojećem rješenju u Pythonu koji generira te datoteke. Generator je realiziran u 3 koraka: prvo su izrađeni SQL baza podataka i model parser, zatim je zamijenjen *pickle* serijalizirani model (model.zip) sa spomenutom SQL bazom podataka, te je izrađen XCP generator u C++ programskom jeziku. Nakon izrade novog rješenja generirane su A2L datoteke te su pokrenute na MotionWise ploči i komunikacija je uspješno uspostavljenja.

Ključne riječi: AUTOSAR, XCP generator, A2L, ECU, pickle, SQL, model

ABSTRACT

A tool for classification of system model parameters

With the technological progress of the automotive industry, electronics in the car is becoming increasingly important, and thus the number of electronic control units is increasing. A2L files contain relevant information about ECU objects such as characteristics and measurement variables. As part of this thesis, it was necessary to create an XCP generator in C++ based on the existing solution in Python that generates these files. The generator was realized in 3 steps: first the SQL database and the parser model were created, then the *pickle* serialized model (model.zip) was replaced with the mentioned SQL database, and finally the XCP generator in the C++ programming language was created. After creating a new solution, A2L files were generated and launched on the MotionWise board and the communication was successful.

ŽIVOTOPIS

Nikola Hlavsa je rođen 20.11.1995. godine u Našicama. Osnovnu školu Matije Gupca završava u Magadenovcu. Zatim upisuje prirodoslovno-matematičku gimnaziju u Srednjoj školi Isidora Kršnjavog u Našicama. Završetkom srednje škole upisuje se na preddiplomski studij Računarstva na Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku, tadašnji Elektrotehnički fakultet. Preddiplomski studij Računarstva završava 2018. godine i iste godine upisuje diplomski studij Računalno inženjerstvo.