Rule Based Car Diagnostic System

Marko Kopanja SW34-2015, Jovan Laković SW51-2015, Miloš Zrnić SW93-2017

Ideja je implementirati aplikaciju koja omogućava proces dijagnostike kvarova na automobilu upotrebom ekspertskog sistema vođenog pravilima (*rule-based expert system*). Sistem je namenjen samim vlasnicima automobila, u slučaju da je došlo do nekog sitnijeg kvara koji bi vlasnik automobila mogao da otkloni bez stručne pomoći odnosno automehaničara.

Deo sistema koji predlaže proces otklanjanja kvara je veoma bitan, iz razloga što vlasnici automobila uglavnom ne poseduju napredno stručno znanje iz ove oblasti. Sistem takođe poseduje i modul za praćenje kvarova, odnosno *monitoring* automobila (nešto poput online servisne istorije). Smatramo da je nepostojanje monitoringa nedostatak postojećih rešenja. Postojeći sistemi se uglavnom baziraju samo na trenutnoj dijagnostici, gde se ne prati stanje automobila na duže staze.

## Korisnici sistema

* Administrator - održavanje ekspertskog znanja i samog sistema
* Običan korisnik – vlasnik automobila

## Funkcionalnosti sistema

* Dijagnostika kvarova – rezoner otkriva koji je najverovatniji kvar na osnovu nuspojava (nekarakterističnih ponašanja) koje vozilo manifestuje i predlaže potencijalno rešenje za otklanjanje problema
* Otkrivanje potencijalnih velikih problema i sprečavanje istih, na osnovu monitoringa vozila kroz duži vremenski period (primer: akumulator se ispraznio 3 puta u roku od 10 dana --> verovatno postoji kvar na alternatoru (uređaju koji dopunjuje akumulator u vožnji, a ne na samom akumulatoru)

## Rad sistema

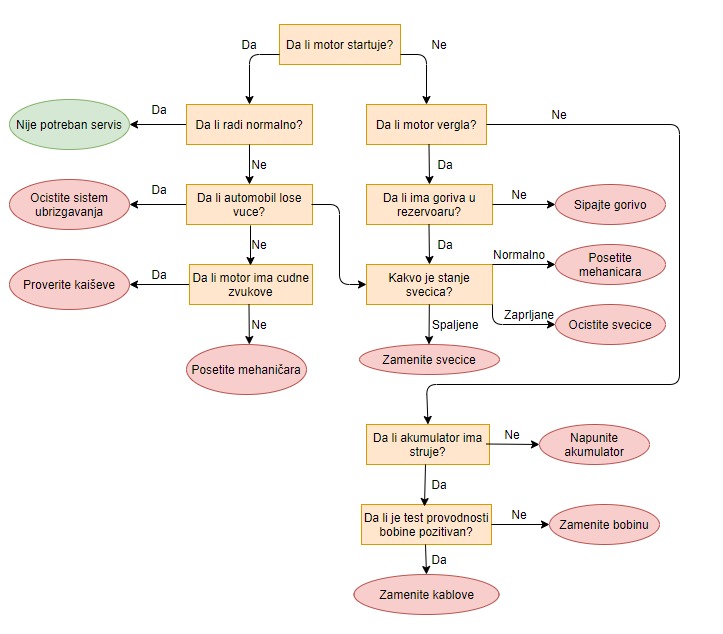
* Ulazni podaci: prolaskom kroz niz pitanja, korisnik unosi podatke o radu automobila I problemima koje je uočio
* Izlazni podaci: izveštaj o svim problemima koji su primećeni, potencijalno rešenje problema, kao i izveštaj o otklonjenom problemu
  + Izveštaj o otklonjenom problemu odnosi se na *monitoring* - radi se update modela sa ciljem da se predvidi neki veći problem koji postoji na vozilu, a koji je samo prividno uklonjen prethodnim rešenjem (konkretnije objašnjeno u primeru u nastavku)
* Konkretan primer:

Korisnik prolazi kroz pitanja i odgovara na njih (ulazni podaci):

* + Da li motor može da se pokrene? - NE
  + Da li se motor fizički okreće (vergla)? - NE
  + Da li akumulator ima napon? - NE

Nakon što je odgovorio na pitanja, pokreće se rezonovanje i korisnik dobija odgovor (izlazni podaci) :

* + Korisnik dobija povratnu informaciju: *Potrebno je napuniti akumulator.*
  + Sistem radi update servisne istorije i otkriva da je ovo već treći put u roku od 2 nedelje da je akumulator bio prazan.
  + Korisnik dobija novu povratnu informaciju: *Na osnovu ranijih problema, moguće je postojanje problema sa alternatorom*.



Slika - Rezonovanje jednostavnih kvarova

Napomene:

Trenutni graf toka aplikacije će biti proširen sa još pravila. Ovo je samo osnovna ideja, da bi se stekao utisak šta želimo da postignemo sistemom. Takođe, biće dodato još kompleksnih pravila koja se odnose na monitoring i na otkrivanje nekih većih problema sa vozilom ukoliko je ustanovljeno da se neki kvarovi ponavljaju u određenom vremenskom intervalu. Moguće je i da kombinacija različitih ‘jednostavnih kvarova’ u određenom vremenskom intervalu rezultuje nekim kompleksnijim kvarom koji se na prvi pogled ne može otkriti.

## Zakljucak

Dakle, kao sto je prvobitno rečeno, naša aplikacija bi pored jednostavnih rešenja za popravku vozila, sadržala i deo gde bi se sve prijave kvarova čuvale radi kreiranja istorije kvarova za konkretno vozilo. Upravo istorija kvarova bi pomogla da se kreiraju kompleksnija pravila.

Pored već pomenutog problema, gde bi recimo korisnik prijavio prazan akumulator, i aplikacija bi mu ponudila rešenje kako da napuni akumulator, ova prijava bi bila evidentirana u istoriju kvarova. U slucaju da korisnik3 ili vise puta u roku od mesec dana prijavi prazan akumulator, aplikacijabi kroz istoriju kvarova detektovala da se pomenuti problem desava vec 3 put i skrenula pažnju korisniku da problem možda nije sam akumulator, većda neki drugi potrošač u sistemu “vuče” struju što dovodi do pražnjenja akumulatora.

Još jedan kompleksniji problem je onaj gde bi više različitih kvarova, prijavljenih u različitim trenucima, mogli da ukažu na ozbiljniji problem na vozilu.Primera radi, korisnik prijavi da mu se smanjio nivo rashladne tečnosti motora, aplikacija ga posavetuje da proveri moguća curenja na crevima I posudi za tečnost, zatim da dospe rashladnu tečnost a zatim i zabeleži ovu prijavu u istoriju kvarova.

Nekoliko dana nakon toga, korisnik prijavi pojavu belog dima na auspuhu vozila. U tom trenutku aplikacija “povezuje” da je korisnik nekoliko dana pre toga prijavio problem sa rashladnom tečnošću, i da ova dva simptoma, kada su istovremeno prisutna ukazuju na to da motor “troši” rashladnu tečnost, odnosno da postoji kvar na glavi motora.

**Dopuna vezana za implementaciju kompleksnijih pravila 🡺**

**Model:**

1. Komponenta koja predstavlja trenutno stanje auta (potential problems)
2. Komponenta koja predstavlja istoriju problema(delova) sa autom (past problems)

**Rule Layer 1** (potential problems):

* Kolekcija pravila koja na osnovu unetih simpoma sa fronta pronalazi potencijalne probleme I updajtuje model (verovatno Car, mozda i posebnu kolekciju potential problems).

Symptoms -> Potential problems -> update Model(Car)

( Za neke jednostavne simptome/probleme mozda cak I da se ne mora ulazi u **Layer 2** (updatuje model) vec se odmah preskace na Fix, ako taj simptom/problem upucuje uvek na jednu istu stvar).

**Layer 2** (*Actual problems and fixes*)

* Kolekcija pravila koja sadrzi kombinacije past problems I potential problems I na osnovu njih odredjuje sta je najverovatnije **Actual Problem** I nudi Fix za njega.

**Forward Chaining 2**

* Nadamo se da samo ova podela na layere pokriva ovaj deo price. *Layer 1* vrsi update modela na osnovu unetih simptoma zbog kojeg se okidaju pravila iz *Layer 2. (vec implementirano)*

**Prioriteti okidanja pravila:**

Zbog slucajeva da vise pravila mogu da se okinu na osnovu jedne kombinacije past/potential problems, radi izbora Actual Problema pravilima se zadaju prioriteti:

Ne znamo da li moze doci do konflikta ako 2 pravila koja treba da se okinu imaju isti prioritet.

Primer 🡺

================================================================================

*Potential problem = Battery is empty*

----------------------------------------------------------------

**Rule “Battery is empty”**

*Salience = 1*

When batt.isEmpty() == True

Then: Fix => “**Recharge** **battery**”

*Past problem = Battery is empty (x5)*

-----------------------------------------------

**Rule “Battery is damaged”**

*Salience = 2*

When num(batt.isEmpty()) > 5

Then: Fix => “**Change battery**

================================================================================

Nakon ponudjenog fixa korisnik ima opciju ***Yes/No*** na pitanje *Did the fix solve your problem*?

1. **Yes** -> potential problem se dodaje u istoriju (postaje past problem)
2. **No** -> nudi se drugi *Actual problem/fix*