Jedno rješenje analize i prikaza kontrolnih tačaka definisanih podešavanjem AUTOSAR nadzornog časovnika

Ivana Tešević, Branko Milošević, Dejan Bokan i Bogdan Pavković

***Apstrakt*— Razvojem automobilske industrije i softvera unutar nje kao tehnička posljedica javila se potreba za obaveznom integracijom zaštitnih mehanizama u ugrađenim jezgrima operativnog sistema. Jedan od osnovnih mehanizama za zaštitu sistema jeste nadzorni časovnik (eng. watchdog, WDG). Ova komponenta ima za cilj da nadgleda sve ostale komponente pokrenute od strane raspoređivača i time omogući bezbjedan rad sistema. Kako je probleme koje nadzorni časovnik prijavljuje relativno teško ispratiti i analizirati u stvarnom sistemu, došlo se do ideje da se oponaša rad komponente nadzornog časovnika na računaru sa istim ulaznim parametrima kao u živom sistemu. U ovom radu je dato rješenje za simulaciju mehanizama nadgledanja sistema definisane AUTOSAR arhitekture. Simulacijom je omogućeno da se minimalizuju odstupanja, predvide greške u sistemu i olakša sama analiza. Rad može doprinijeti bržem razvoju sistema jer omogućava da se prije implementacije predvide greške koje će se desiti u sistemu.**

Ključne riječi— AUTOSAR, WDG, WdgM, WdgIf, nadgledanje krajnjih rokova, logički nadzor, nadgledanje u realnom vremenu, najduže vrijeme izvršenja(WCET).

I. UVOD

AUTOMOBILSKA industrija je grana industrije koja se sveobuhvatno razvija u posljednjoj deceniji. Dizajn vozila u automobilskoj industriji tradicionalno se oslanja na diskretne hardverske komponente (elektronske upravljačke jedinice - ECU), sa vrlo malo potrebnog softvera. Sa poboljšanjem automobilske industrije, softver za nju se razvijao. Danas je softverski dio prevladao hardver[1][2] . Softverske komponente postale su komplikovanije i zahtjevnije od hardverskih komponenti. Danas automobili nude mnogo više mogućnosti, uključujući i autonomne funkcije pri vožnji[1]. Postoji pet nivoa automatizacije vožnje, dok je industrija trenutno na trećem nivou, očekujući da će dostići nivo četiri i pet do 2025. godine[3]. Vozači će moći bezbjedno da skrenu pažnju sa vožnje, npr. gledati film ili čitati knjigu.

Činjenica je da je sve više dobavljača u ovoj grani industrije, pa se pojavila potreba za standardizacijom proizvodnje softvera. Da bi se udovoljilo ovom zahtjevu, stvorena je platforma AUTOSAR (eng. *Automotive Open System Architecture*) [1].

Kako se ova industrija sve više širi i kako rastu softverski zahtjevi povećava se i potreba za raznim alatima za održavanje bezbjednosti sistema. Ovakvi sistemi moraju podlijegati raznim testovima i konstantno se nadgledati kako bi se u potpunosti otklonila mogućnost greške, jer i najmanja greška može imati fatalne posljedice.

Nadzorni časovnik je jedna od komponenti koja za cilj ima nadzor cijelog sistema. Ova komponenta kao takva sama po sebi mora imati maksimalni kvalitet koda i podlijegati najvećim provjerama. Bilo kakva greška primijećena od strane WDG komponente biće ispraćena reakcijom gašenja cijelog sistema. Ovakav vid zaštite u industriji otežava testiranje, predviđanje ali i pronalaženje greške u toku rada. Zato se javila potreba da se prikaže jedno rješenje za oponašanje sistema kako bi se moglo predvidjeti i upoznati sa greškama i načinima na koji dolazi do njih.

Ovaj rad prikazuje jedno rješenje analize i prikaza kontrolnih tačaka definisanih podešavanjem *AUTOSAR* nadzornog časovnika. Prikazaće se simulacija poremećaja u sistemu koji će nadzorni časovnik prepoznati pomoću nadzornih mehanizama. Namjerno izazivanje poremećaja i reakcija nadzornog časovnika na te poremećaje doprinijeće lakšem testiranju i predviđanju u stvarnom sistemu. Pomoću ovog rješenja nudi se mogućnost korišćenja stvarnih ulaznih parametara i testiranje raznih poremećaja i lanca događaja nakon namjerno izazvane greške. Mogućnost predviđanja i vizualni prikaz sistema nakon poremećaja jesu glavni doprinos ovog rada. Postojeća literatura na temu nadzornog časovnika[5][6] skoncentrisana je na unaprjeđenju mehanizama zaštite ili na načinu testiranja sistema i ispravnosti sprege nadzornog časovnika.

Drugo poglavlje će dati teorijske osnove o načinu rada svih modula, samom *AUTOSAR* standardu i vertikali nadzornog časovnika sa definisanim modulima.

U trećem poglavlju biće opisan način rada, pristup rješenju i podloga za nastavak i samu implementaciju.

U četvrtom poglavlju su opisani moduli, dato je programsko rješenje i sami postupci implementacije.

U petom poglavlju su prikazani rezultati rada, način testiranja, kao i svrha samog rješenja.

II. AUTOSAR STANDARD

Osnovan 2003. godine, AUTOSAR predstavlja međunarodno razvojno partnerstvo stranaka iz automobilske industrije. Cilj ove saradnje bio je stvaranje i uspostavljanje otvorene i standardizovane softverske arhitekture za osnovne elektronske jedinice autonomnog vozila nazvane ECU.

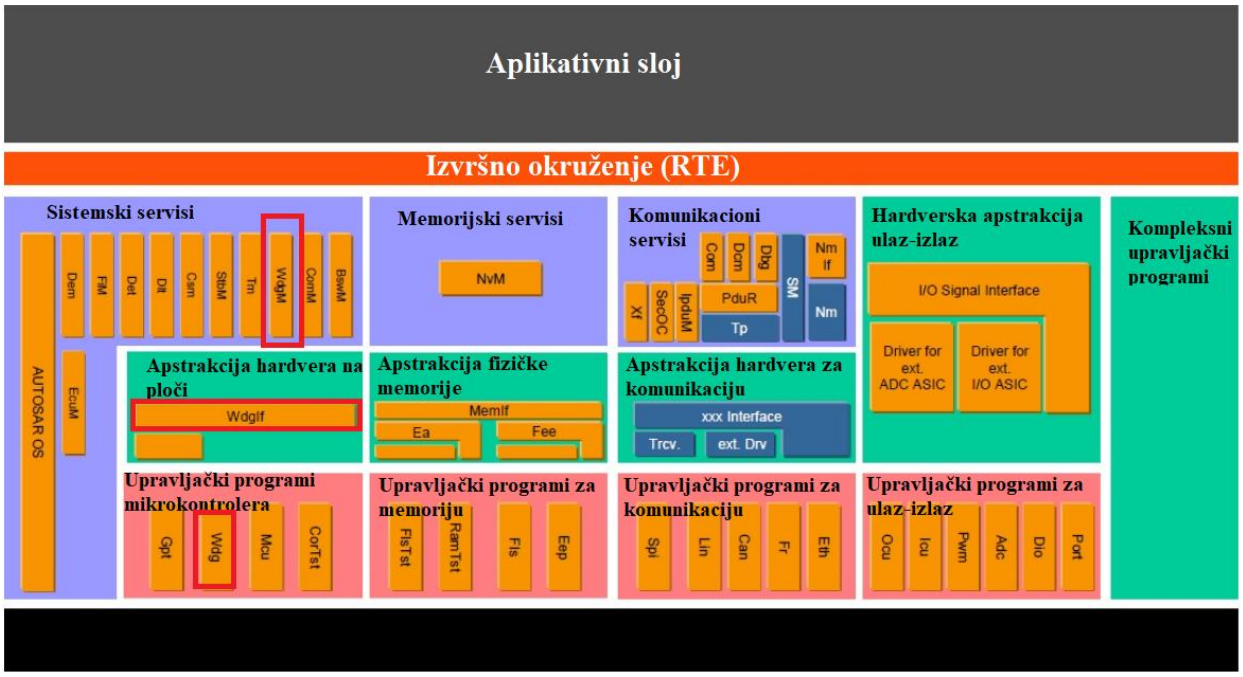
Ivana Tešević, RT-RK Institute for Computer Based Systems, Novi Sad, Srbija (e-mail: ivana.tesevic@rt-rk.com)

Branko Milošević, RT-RK Institute for Computer Based Systems, Novi Sad, Srbija (e-mail: branko.milosevic@rt-rk.com).

Bogdan Pavković, RT-RK Institute for Computer Based Systems, Novi Sad, Srbija (e-mail: bogdan.pavkovic@rt-rk.com)

Dejan Bokan, RT-RK Institute for Computer Based Systems, Novi Sad, Srbija (e-mail: dejan.bokan@rt-rk.com).

AUTOSAR standard daje set specifikacija koje opisuju funkcionalnosti softverskih modula i realizuje zajedničke metode daljeg razvoja na osnovu standardizovanog formata[1]. Arhitektura ovog standarda, odnosno AUTOSAR modela, na najvišem nivou apstrakcije prepoznaje tri različite softverske cjeline[4] (Slika 1.):



Sl. 1. AUTOSAR model [1]

• Osnovni softver (eng. *Basic software module*, *BSW*)- ovaj sloj se sastoji od modula koji su neophodni za funkcionisanje višeg softverskog sloja. Slojevi od kojih se sastoji osnovni softver su: sloj apstrakcije ECU, složeni upravljački programi, sloj apstrakcije mikrokontrolera (eng. MCAL).

• Izvršno okruženje(eng. Runtime environment, RTE)- realizuje komunikaciju između softverskih komponenti i osnovnog softvera.

•Aplikativni sloj(eng. Application Layer)-funkcionalnost elektronskih kontrolnih jedinica je implementirana u obliku pojedinačnih softverskih komponenti.

A. *Nadzorni časovnik*

Za automobilske sigurnosne sisteme kritično je pitanje zadovoljavanja zahtjeva u realnom vremenu na deterministički način. Da bi se udovoljilo vremenskim ograničenjima, razvijeni su različiti mehanizmi praćenja, kao što su nadzorni hardver ECU jedinice[7], nadgledanje krajnjih rokova[8][9], nadgledanje vremena izvršenja 0. Ovakav vid nadzora kreiran je kako bi se osigurao tačan raspored zadataka0 .

Vertikalu nadzornog časovnika u AUTOSAR slojevitoj arhitekturi čine rukovodilac nadzornog časovnika(nalazi se u servisnom sloju eng. *Service Layer*), sprega nadzornog časovnika(smještena u ECU sloju apstrakcije) i upravljač nadzornog časovnika(smješten u sloju apstrakcije mikrokontrolera)[12] Ovi moduli pružaju usluge za praćenje vremena i ispravnosti izvršenja entiteta u aplikaciji i osnovnom softveru.

B. *Rukovodilac nadzornog časovnika*

Rukovodilac nadzornog časovnika (eng. *watchdog manager*, *WdgM*) je osnovni softverski modul u servisnom nivou koji nadgleda tok programa[13]. Kada se otkrije narušavanje unaprijed definisanih vremenskih ili logičkih ograničenja u programskom toku, potrebno je evidentirati grešku i preći u bezbjedno stanje nakon vremenskog kašnjenja. Sigurno stanje se postiže ponovnim pokretanjem ili izostavljanjem aktiviranja modula nadzornog časovnika.

Po *AUTOSAR* definiciji, tačke u kontroli toka nadgledanog entiteta gdje se aktivnost prijavljuje rukovodiocu nadzornog časovnika su kontrolne tačke.

Polja koja opisuju kontrolnu tačku su:

• *ID* kontrolne tačke

• Lokalni početak, lokalni kraj

• Globalni početak, globalni kraj

Lokalni prelazi predstavljaju prelaze između dvije kontrolne tačke unutar istog nadgledanog entiteta.

Globalni prelazi su prelazi između dvije kontrolne tačke koje pripadaju različitim entitetima.

Nadgledani entitet predstavljen je kontrolnim tačkama kojih može biti jedna ili više. Svaki nadgledani entitet može imati jedno ime i jedno stanje.

Kada se govori o mehanizmima nadgledanja u *WdgM* modulu pominju se tri tipa nadgledanja[14]:

• Nadgledanje u realnom vremenu (eng. *Alive Supervision*) – prati frekvenciju izvršavanja određenog softverskog dijela. To znači da rukovodilac provjerava da li se nadgledani entitet javlja suviše često ili suviše rijetko.

• Nadgledanje krajnjih rokova (eng. *Deadline Supervision*) – nadgleda vrijeme potrebno za izvršavanje nadgledanog entiteta. Glavna svrha je provjera vremenskog, dinamičkog ponašanja entiteta.

• Logički nadzor (eng. *Logical Supervision/Program Flow check*) – nadgleda tok izvršavanja u programu.

Dva su ključna pojma koja treba pomenuti kada je u pitanju nadgledanje i reakcija na greške, a to su vrijeme otkrivanja greške i vrijeme reakcije na grešku.

Vrijeme otkrivanja greške (eng. *Fault Detection*) traje od pojave greške do trenutka kada je ta greška otkrivena i prijavljena sistemu.

Vrijeme reakcije na grešku (eng. *Fault Reaction*) traje od trenutka otkrivanja greške do ponovnog pokretanja sistema. *WdgM* reakcija na grešku:

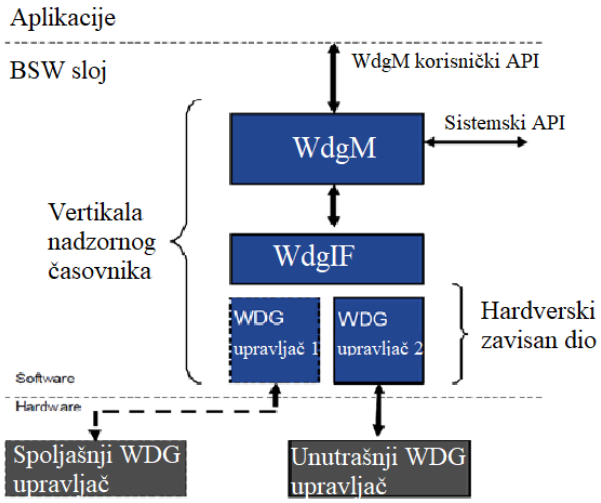
- Obavještenje iz funkcije povratnog poziva
- Ponovno pokretanje sistema
- Stopiranje okidanja nadzornog časovnika

C. *Sprega nadzornog časovnika*

Sprega nadzornog časovnika (*WdgIf*) je dio ECU apstraktnog sloja. Uvijek se nalazi ispod rukovodioca i iznad upravljača nadzornog časovnika. Sprega komunicira sa upravljačkim programima ispod. Implementacija sprege zavisi od broja upravljača[14].

D. *Upravljač nadzornog časovnika*

Upravljač je zadužen za pristup samoj periferiji direktno[15] (unutrašnjem i spoljašnjem nadzornom časovniku) i nalazi se u sloju apstrakcije mikrokontrolera. Modul za spoljašnji nadzorni časovnik koristi druge module za pristup spoljnom uređaju.



Sl. 2 Slojevita struktura WDG 1

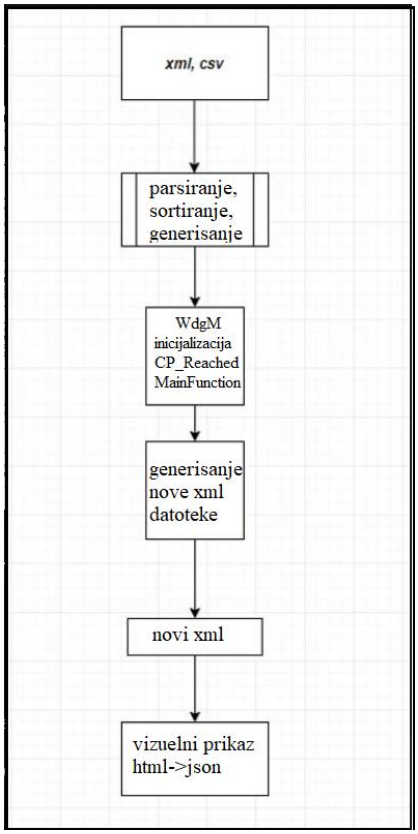
III. KONCEPT RJEŠENJA

Uključen nadzorni časovnik u stvarnom sistemu može stvarati velike probleme prilikom analiziranja nekog problema. Stalno gašenje i ponovno pokretanje jedan su od pokazatelja zašto je to tako. Kako bi se nastavila analiza neke greške na projektima se obično podliježe gašenju nadzornog časovnika i nakon toga se nastavlja sa analiziranjem. Ovaj rad predstavlja jedan pomoćni alat prilikom te analize koji je omogućio da pomoću komandne linije unese željeni poremećaj i isprati lanac događaja koji slijede nakon njega.

Na osnovu već generisanog operativnog sistema odrađeno je parsiranje redoslijeda zadataka i AUTOSAR generisane konfiguracije za stek modul nadzornog časovnika. Parsirani podaci bili su neophodni za grafički prikaz raspoređivača. Grafička predstava kontrolnih tačaka odrađena je tako da se vodilo računa o redoslijedu, kao i koja je kontrolna tačka dodijeljena kom zadatku. Pomenuti grafički prikaz omogućava da se jednostavno uoči poremećaj, tako da je grafički prikaz jedan od osnovnih alata koji su korišteni prilikom analize i testiranja. Ulazni parametri za parsiranje su definisani u csv i arxml formatu (*SE WCET*, *SE period*, *WCET neto*, *WCET abs*).

Sledeći korak jeste simuliranje nadgledanja sistema definisane AUTOSAR arhitekture. Osnovni cilj ovog koraka jeste da prikaže što približnije slijed događaja i grešaka na računaru, kao što je očekivano i u stvarnom sistemu. Glavna razlika je ta što greške koje vidimo na računaru nemaju nikakvu bezbjednosnu posljedicu po izvršenje, već služe isključivo u svrhu analize i rezultiraće ispisima i informativnim porukama, umjesto gašenjem sistema.

Nakon implementacije grafičkog prikaza i simulacije mehanizama, pristupa se analizi sistema, praćenju ponašanja sistema pod uticajima raznih poremećaja koji su namjerno izazvani i koji se odnose na mehanizme nadgledanja. Namjernim izazivanjem grešaka olakšava se analiziranje istih, očekivanih u procesu rada na stvarnom projektu. Data je mogućnost da se predvide različiti lanci događaja, kao i da se smanji vrijeme koje bi bilo potrošeno na pokušaje analize problema uslijed stalnog gašenja sistema.



Sl. 3 Dijagram rješenja

A. *Rukovodilac nadzornog sistema u višejezgarnom sistemu*

Rukovodilac može biti korišten u jednojezgarnim i višejezgarnim sistemima. U ovom radu obrađen je rukovodilac u višejezgarnom sistemu.

Svaka instanca treba da bude nezavisna jedna od druge i mora biti inicijalizovana njenom sopstvenom konfiguracijom. Poziv *Main* funkcije je odvojen. U stvarnom sistemu se softverske komponente izvršavaju paralelno i vremenski nezavisno. Svako jezgro ima svoje sopstveno vrijeme.

Sl 4. Isječak iz arxml fajla

B. *Sortiranje kontrolnih tačaka prema rasporedu*

Parametri koji su značajni i koji opisuju izvršenje sistema u vremenu su zadati u xml datoteci, a potrebni entiteti su u csv datoteci. Ove dvije ulazne datoteke moraju biti međusobno povezane i predstavljaju jednu cjelinu. Parsiranjem ovih datoteka dobijeni su svi podaci potrebni za simulaciju nadgledanja. Ti parametri su: naziv, perioda, trajanje, vrijeme početka, prioritet i ID. Isječak iz arxml fajla je prikazan na Sl 4.

Nakon izvlačenja pomenutih podataka sve kontrolne tačke sortirane su po vremenu i po prioritetu. Kontrolna tačka koja ima manji prioritet će biti prekinuta ako se u toku njenog trajanja javi neka druga tačka većeg prioriteta.

Nakon sortiranja sve kontrolne tačke kreću sa izvršenjem, baš kao u stvarnom sistemu, istim redoslijedom kako je zahtijevano u ulaznoj datoteci i po prioritetu. Ono što je takođe bilo bitno prikazati jeste vrijeme trajanja koje je oponašano na osnovu ulaznih informacija.

C. *Simulacija nadgledanja*

Nakon uspješno obavljene inicijalizacije, sortiranja i prozivanja kontrolnih tačaka potrebno je simulirati rad prethodno opisanih načina nadgledanja.

Rukovodilac *Main* je sastavni dio izvršenja i on se prozove po zadatom intervalu od 10 milisekundi i tada se vrši provjera nadgledanje u realnom vremenu, nadgledanje krajnjih rokova, logički nadzor.

Vrijeme u sistemu nadzornog časovnika je predstavljeno u tikovima. Potrebno je simulirati vrijeme tako da odgovara vremenu iz stvarnog sistema.

Na osnovu tog vremena se provjeravaju nadgledanja. Provjeru nadgledanja u realnom vremenu treba obaviti tako da se u slučaju da se kontrolna tačka ne javi u očekivanom vremenskom intervalu na konzoli dobijemo ispis o grešci koja se desila.

IV. PROGRAMSKO RJEŠENJE

A. *Parsiranje rasporeda i sortiranje*

Za ulazne podatke iskorišćene su xml i csv datoteke iz stvarnog sistema. Parsiranje je rađeno u programskom jeziku *Python*, svi podaci koji su izvučeni iz tih datoteka su generisani i urađeno je prozivanje funkcije *WdgM_CheckpointReached()*.

Osnovni problem koji se javio prije prozivanja ove funkcije bio je sortiranje kontrolnih tačaka prema rasporedu. Sortiranje je takođe odrađeno u programskom jeziku *Python* i korištene su funkcije:

- *expiry_points()* - funkcija koja sortira podatke koji su izvučeni iz ulaznih datoteka parsiranjem. Sortira kontrolne tačke po vremenu njihovog javljanja i po jezgrima.
- *priority_sort()* - Prethodno sortirana lista po vremenu javljanja se sortira i po prioritetima . Ako se desi da dvije kontrolne tačke počinju istovremeno prednost će imati tačka sa većim prioritetom, tačka manjeg prioriteta ostaje da čeka svoje red. Entitet može biti prekinut i u toku izvršenja, ako se desi da je došlo do javljanja izvršioca sa većim prioritetom, trenutni entitet ostaje u stanju čekanja sve dok mu se ne signalizira da je prioritetniji entitet završio sa radom.

Nakon sortiranja je generisano kojim se redoslijedom vrši pozivanje *WdgM_CheckpointReached()* funkcije.

B. *Implementacija rukovodioca*

Prvi korak koji je odrađen jeste postupak inicijalizacije. Svaki zadatak inicijalizovan je pomoću funkcije *WdgM_Init()*.

U stvarnom sistemu WDG zadatak ponavlja se kružno na svakih 10 milisekundi. To znači da se poziv funkcije *WdgM_MainFunction()* ponavlja svakih 10 milisekundi.

Ova funkcija ima ključnu ulogu jer se u njoj vrše provjere ispravnosti. Trajanje jednog ciklusa naziva se hiper period, Na osnovu trenutnih ulaznih parametara koji su obrađeni u ovom primjeru koji će biti opisan hiper period je 80 milisekundi i nakon toga se završava jedan ciklus nadzora.

Nakon izvršenja *WdgM_MainFunction()* očekuje se neka od reakcija rukovodioca.

- Ako dođe do greške u nadgledanju u realnom vremenu greška će biti detektovana na kraju nadgledanog referentnog ciklusa (eng. *Alive supervision reference cycle*).
- U slučaju nadgledanja programskog toka ako dođe do greške ona će biti detektovana na kraju svakog nadgledanog ciklusa.
- Ako je greška u nadgledanju krajnjih rokova ona će biti detektovana na kraju svakog nadgledanog ciklusa, nastavak kršenja ovog vida nadgledanja detektuje se na kraju svakog krajnji rok nadgledanog entiteta.

Ponašanje sistema nakon uočavanja neke od pomenutih grešaka zavisi od konfiguracije i tipa poremećaja.

V. PROVJERA ISPRAVNOSTI

A. *Opis testiranja*

U svrhu testiranja korišteni su ulazni parametri sa stvarnog sistema. Ovakav pristup omogućio je poređenje sa stvarnim sistemskim greškama i utvrditi ispravnost samog rada.

Kako je stvarni sistem čiji si ulazni parametri iskorišćeni sadržao 3 jezgra, a ona u sistemu rade u paraleli. U ovom rade sva tri jezgra su testirana istovremeno i softverski spojena u jednu cjelinu, prikazan je njihov paralelizam. Na računaru se pomoću komandne linije prati ispis rezultata, kao rezultat testiranja dobijaju se poruke o prekršajima.

Prilikom pokretanja radi se inicijalizacija sistema.

Korisnik treba da odabere jezgro na kome će nanijeti poremećaj kao i vrstu poremećaja koju želi, promjena *WCET* vremena ili greška u nadgledanju u realnom vremenu.

Ako je u sistemu sve prošlo bez greške prilikom izvršavanja rukovodioca prozvaće se *TriggerWindow* funkcija na osnovu čega je testirana ispravnost. Vizualni prikaz sistema bez greške dat je na Sl.5

B. *Poremećaj nastao promjenom WCET vremena*

Najgore vrijeme izvršenja predstavlja ukupno vrijeme koje je dato jednom zadatku da se izvrši. Vrijednost ovog parametra predstavljena je sa dva termina *bruto* i *neto WCET*. Kada je riječ o ukupnom *bruto* vremenu možemo reći da je to vrijeme koje protekne od početka do kraja zadatka sa uračunatim svim prekidima od strane prioritetnih zadataka. Dok je *neto WCET* vrijeme od početka do kraja ali predstavlja samo sabrane vremenske trenutke u kojima je aktivan dati zadatak.

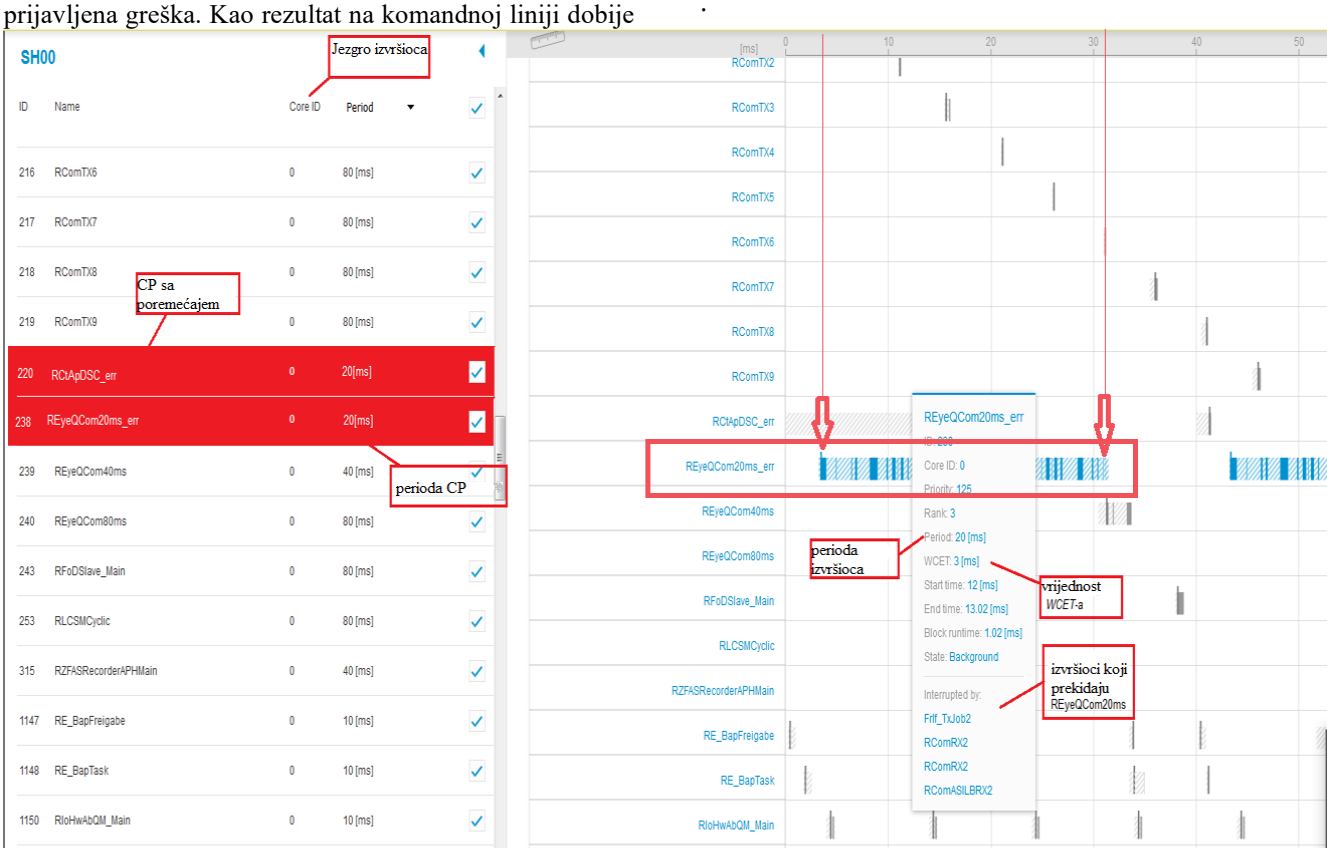
Vrijednost *WCET* vremena koja je konfigurisana za određeni nadgledani entitet i data u rasporedu očekivana je vrijednost u sistemu sa kojom svi nadzori rade ispravno. Nakon što se napravi neka promjena *WCET* vremena i ispisivanja poruka o ispravnosti sistema generiše se nova datoteka *arxml* koja predstavlja ulazni parametar za vizualizaciju sistema. Ako je neka promjena unesena na grafičkom prikazu promijenjena kontrolna tačka mijenja boju sto se vidi na Sl 5. Greška u sistemu vidljiva je na slici gdje su crvenom bojom markirani izvršioci kod kojih je prijavljena greška. Kao rezultat na komandnoj liniji dobije se poruka o svim prekršajima koje je izazvala promjena *WCET* vremena na željenom entitetu.

Na osnovu vizualizacije omogućeno je lakše praćenje dešavanja u sistemu, način na koji se nakon bilo koje promjene izmiješaju kontrolne tačke. Simulirana je vremenska osa i kontrolne tačke u vremenu.

C. *Poremećaj u nadgledanju realnog vremena*

Kada se nanese ovaj vid poremećaja u sistemu dolazi do situacije u kojoj se određena kontrolna tačka ne prozove. Tada sistem detektuje grešku u nadgledanju u realnom vremenu. Primjer iz tabele takođe pokazuje grešku nad *RCtApDSC* koja ima period 20 milisekundi. Možemo uočiti da je došlo do problema kada je rukovodilac prepoznao da se u prvih 20 milisekundi nije javila ova kontrolna tačka, a mehanizam nadzora u realnom vremenu očekivao je da će doći do njenog javljanja. Nakon otkrivanja problema brojač se nije uvećao i kao status vraćena je vrijednost “nije uspjelo”(eng. FAILED).

Kada se poveća WCET u ovom slučaju nad nadgledanim entitetom pod nazivom *ReyeQCom20ms* koji je prikazan u tabeli (T 1.) desi se poremećaj u prvih 10 milisekundi. Očekivana vrijednost je 2.7 milisekundi jer je ovo nadgledani entitet koji je niskog prioriteta i isprekidan je od strane ostalih koji imaju veći prioritet. Prilikom testiranja promijenili smo vrijednost na 3 milisekunde. Zbog poremećaja na jednoj kontrolnoj tački i greške u nadgledanju krajnjeg roka u prvih 10 milisekundi očita se greška, ali se prozove i funkcija *TriggerWindow*.



Sl. 5 Greške na jezgru 0 nakon promjene *WCET* vremena nadgledanog entiteta *ReyeQCom20ms*

Na Sl 5 može se uočiti kako će *ReyeQCom20ms* zadatak imati uticaj na sistem kada se njemu nanese vremenski poremećaj. Takođe primjetna je lista izvršilaca koji prekidaju pomenuti entitet zbog većeg prioriteta, pri čemu će svaki od pomenutih prekinuti izvršenje. trenutnog. Puna linija predstavlja izvršenje *ReyeQCom20ms* dok je isprekidanim poljima predstavljen

period kada je posmatrani entitet u pozadini i čeka na izvršenje zadatka sa većim prioritetom.

jezgra / poremećaj	JEZGRO 0	JEZGRO 1	JEZGRO 2
sistem bez poremećaja	TriggerWindow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)
greška na kontrolnoj tački ReyeQCom 20ms WCET=3ms	*ReyeQCom20ms Execution flow violation*	TriggerWiudow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)
	TriggerWindow	TriggerWindow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)
	RBAQM10ms Execution flow ERROR	TriggerWindow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)
	SetResetReasone		
Alive monitoring greska nad RCtApDSC	*RCtApDSC Execution flow ERROR*	TriggerWindow (za svaki zadatak koji pripada tom jezgru)	TriggerWindow (za svaki zadatak koji pripada tom jezgru)
	SetResetReasone		

T 1. Ponašanje sistema nakon poremećaja 1

VI. ZAKLJUČAK

U okviru ovog rada prikazano je rješenje i način praćenja poremećaja prijavljenih od strane nadzornog časovnika. Česte su situacije da se u radu na realnoj platformi nailazi na poteškoće po pitanju očekivanog ponašanja hardvera na određene zahtjeve iz softvera. Kada govorimo o samom nadzornom časovniku i reakciji fizičkog upravljača nekada sa sigurnošću ne možemo da tvrdimo šta je uzrok okidanja greške i gašenja sistema. Sigurnosno gašenje može značajno usporiti proces analiziranja nekog problema koji sam po sebi ne mora biti vezan isključivo za nadzorni časovnik. Takav vid poteškoća je moguće pratiti samo isključenjem upravljača. Simulacijom je omogućeno da se minimalizuju ta odstupanja, predvide greške u sistemu i olakša analiza sistema. Greška koja se desi na jednoj kontrolnoj tački može da prijavi grešku tek na sledećoj kontrolnoj tački koja je u redu. Rad omogućava da se isprati i predvidi takav vid prekršaja.

Kao što je prethodno pomenuto za ovaj rad je korišten već postojeći raspored zadataka koji sam po sebi predstavlja preduslov za početak simulacije. Budući rad obuhvatiće unaprjeđenje postojećeg rješenja simulacijom operativnog sistema, gdje će se voditi računa i o simulaciji raspoređivanja zadataka kako bi se poremećaj mogao nanijeti direktno u raspoređivanju i ispratiti cijeli proces. Ovaj vid unaprjeđenja značajno bi mogao poboljšati analizu i predviđanje grešaka.

LITERATURA

[1] AUTOSAR, "*Layered Software Architecture*," *[Online]*, https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf., 2017-12-08 [Accessed July 2021]

[2] B. Catalin-Virgil, F. Ioan and F. Heininger, "*A new trend in automotive software: AUTOSAR concept*" SACI, IEEE 8th International Symposium, Timisoara, Romania, May 2013.

[3] Techemergence, "*The Self-Driving Car Timeline –Predictions from the Top 11 Global Automakers*", 2018.

[4] H. Fennel et al. "*Achievements and exploitation of the AUTOSAR development partnership"* SAE Technical Paper Series 2006-21-0019, SAE International, October 2006,[Accessed July 2021]

[5] Mazen Ahmed, Mona Safar, "*Formal Verification of AUTOSAR Watchdog Manager Module Using Symbolic Execution*", IEEE 30th International Conference on Microelectronics, 2018

[6] Mazen Ahmed, Mona Safar , *"Symbolic Execution based Verification of Compliance with the ISO 26262 Functional Safety Standard"*, IEEE 14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era, 2019

[7] J. Ganssle, "*Watching the Watchdog*", Embedded World, 2003.

[8] *AUTOSAR* , "*Specification of Communication*" *[Online]*, https://www.autosar.org/fileadmin/user_upload/standards/classic/3-2/AUTOSAR_SWS_COM.pdf [Accessed July 2021]

[9] Michael Kunz, "*OSEK OS"*, March 18, 2009, http://www.uni-obuda.hu/users/schuster.gyorgy/rtos/OSEK.pdf [Accessed July 2021]

[10] The AUTOSAR Consortium, "*AUTOSAR Specification of Operating System"*, pp. 33-35, 2006.

[11] Nahmsuk Oh, P. Shirvani, E. McCluskey, "*Control-Flow Checking by Software Signatures*", IEEE Transaction on Reliability, vol. 51, Mar-2002

[12] Texas Instruments MCUSW, "*Wdg Design Document* " *[Online]*, *http://software-dl.ti.com/jacinto7/esd/processor-sdk-rtos-jacinto7/latest/exports/docs/mcusw/mcal_drv/docs/drv_docs/design_wdg_top.html* [Accessed July 2021]

[13] *AUTOSAR*, "*AUTOSAR SWS Watchdog Manager*" *[Online]*, https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_WatchdogManager.pdf , *2017-12-08* [Accessed July 2021]

[14] *AUTOSAR*, " *Specification of Watchdog Interface*" https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_WatchdogInterface.pdf, [Accessed July 2021]

[15] *AUTOSAR*, "*Specification of Watchdog Driver* "*[Online]* https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_SWS_WatchdogDriver.pdf , [Accessed July 2021]

ABSTRACT

With the development of the automotive industry and software within it, as a technical consequence, there was a need for mandatory integration of protection mechanisms in the embedded cores of the operating system. One of the basic mechanisms for system protection is the watchdog. This component aims to monitor all other components initiated by the scheduler and thus enable the safe operation of the system. As the problems reported by Watchdog are relatively difficult to track and analyze in a real system, the idea came up to simulate the operation of the Watchdog component on a computer with the same input parameters as in a living system. This paper provides a solution for simulating the system monitoring mechanisms of the defined AUTOSAR architecture. The simulation makes it possible to minimize deviations, predict errors in the system and facilitate the analysis itself. Work can contribute to faster system development because it allows to predict errors that will occur in the system before implementation.

ONE SOLUTION FOR ANALYZING AND DISPLYING CHECKPOINTS IN THE AUTOSAR WATCHDOG CONFIGURATION

Ivana Tešević, Branko Milošević Dejan Bokan, Bogdan Pavković