УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

# ПРЕДЛОГ АРХИТЕКТУРЕ СРЕДЊЕГ СЛОЈА СОФТВЕРА ЗА РАЧУНАРСКИ СИСТЕМ У ВОЗИЛИМА

ДОКТОРСКА ДИСЕРТАЦИЈА

Ментор:
проф.др Милан З. Бјелица

Кандидат:
Милена Милошевић

Нови Сад, 2022. године

**УНИВЕРЗИТЕТ У НОВОМ САДУ** **ОБРАЗАЦ – 5а**
**ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА[1]

| | |
|---|---|
| Врста рада: | Докторска дисертација |
| Име и презиме аутора: | Милена Милошевић |
| Ментор (титула, име, презиме, звање, институција) | др Милан З. Бјелица, ванредни професор, Факултет техничких наука |
| Наслов рада: | Предлог архитектуре средњег слоја софтвера за рачунарски систем у возилима |
| Језик публикације (писмо): | Српски (ћирилица) |
| Физички опис рада: | Унети број:<br>Страница: 228<br>Поглавља: 7<br>Референци: 212<br>Табела: 44<br>Слика: 70<br>Графикона: 0<br>Прилога: 3 |
| Научна област: | Електротехничко и рачунарско инжењерство |
| Ужа научна област (научна дисциплина): | Рачунарска техника и рачунарске комуникације |
| Кључне речи / предметна одредница: | Архитектура софтвера, средњи слој софтвера, рачунарски систем у возилима, напредни системи за помоћ возачу, хетерогене платформе у возилу |
| Резиме на језику рада: | Ова докторска дисертација се бави истраживањем из области софтверских платформи у модерним возилима. Са појавом савремених технологија, број и сложеност функција у возилима расте, док произвођачима аутомобила постаје све теже да одржавају такве разноврсне системе због чега конвергирају уједињавању функција, тј. коришћењу што мањег броја савремених чипова на којима би се реализовао што већи број функција. Циљ истраживања у оквиру ове докторске дисертације је да се на основу истраживања стања у области предложи архитектура средњег слоја софтвера за рачунарски систем у возилима, која ће представљати корак напред у поменутој тежњи произвођача аутомобила. Предложено решење треба да омогући и бржи и једноставнији развој апликација у хетерогеном окружењу возила. Решење је реализовано на више платформи са циљем провере функционалности, перформанси решења као и евалуације архитектурних особина које утичу на једноставан развој апликација. Основни допринос се огледа у предложеној спрези која омогућава бржи развој апликација. |
| Датум прихватања теме од стране надлежног већа: | 27.10.2022. |
| Датум одбране: (Попуњава одговарајућа служба) | |
| Чланови комисије: (титула, име, презиме, звање, институција) | Председник: др Никола Теслић, редовни професор, ФТН Нови Сад<br>Члан: др Мирослав Поповић, редовни професор, ФТН Нови Сад<br>Члан: др Иван Каштелан, ванредни професор, ФТН Нови Сад<br>Члан: др Марио Врањеш, ванредни професор, ФЕРИТ Осијек, Хрватска<br>Члан/ментор: др Милан Бјелица, ванредни професор, ФТН Нови Сад |
| Напомена: | |

[1] Аутор докторске дисертације потписао је и приложио следеће Обрасце:
- 5б – Изјава о ауторству;
- 5в – Изјава о истоветности штампане и електронске верзије и о личним подацима;
- 5г – Изјава о коришћењу.

Ове Изјаве се чувају на факултету у штампаном и електронском облику и не кориче се са тезом.

**UNIVERSITY OF NOVI SAD**
**FACULTY OF TECHNICAL SCIENCES**

## KEY WORD DOCUMENTATION[2]

| | |
|---|---|
| Document type: | Doctoral dissertation |
| Author: | Milena Milošević |
| Supervisor (title, first name, last name, position, institution) | PhD Milan Z. Bjelica, associate professor, Faculty of technical sciences |
| Thesis title: | One proposal of software middleware for heterogenous in-vehicle environments |
| Language of text (script): | Serbian language (cyrillic script) |
| Physical description: | Number of:<br>Pages: 228<br>Chapters: 7<br>References: 212<br>Tables: 44<br>Illustrations: 70<br>Graphs: 0<br>Appendices: 3 |
| Scientific field: | Electrical and computer engineering |
| Scientific subfield (scientific discipline): | Computer engineering and computer communications |
| Subject, Key words: | Automotive, middleware, software architecture, advanced driver assistance systems, heterogenous in-vehicle platforms |
| Abstract in English language: | This PhD thesis addressed the problem of the software platforms in the field of heterogeneous in-vehicle environments. With modern technologies, the number and complexity of functions in the vehicle is constantly growing. It becomes harder for OEMs (*Original Equipment Manufacturer*) to maintain such different systems, and as a result there is a tendency to use as few modern chips as possible in order to realize as many functions. The goal of the research within this PhD thesis is to propose, based on the research, software middleware architecture for modern vehicle systems, which will be a step forward in the mentioned aspiration of OEMs. The proposed solution should enable faster and easier development of the applications in such environment. The solution is implemented on the multiple hardware platforms in order to check functionality, performance and to evaluate architectural features that affect ease application development. The main contribution of the thesis is the proposed interface that allows faster and easier application development. |
| Accepted on Scientific Board on: | 27.10.2022. |
| Defended:<br>(Filled by the faculty service) | |
| Thesis Defend Board:<br>(title, first name, last name, position, institution) | President: PhD Nikola Teslić, full professor, Faculty of technical sciences Novi Sad<br>Member: PhD Miroslav Popović, full professor, Faculty of technical sciences Novi Sad<br>Member: PhD Ivan Kaštelan, associate professor, Faculty of technical sciences Novi Sad<br>Member: PhD Mario Vranješ, associate professor, FERIT Osijek, Croatia<br>Member/mentor: PhD Milan Z. Bjelica, associate professor, Faculty of technical sciences Novi Sad |
| Note: | |

- [2] The author of doctoral dissertation has signed the following Statements:
- 5б – Statement on the authority,
- 5в – Statement that the printed and e-version of doctoral dissertation are identical and about personal data,
- 5г – Statement on copyright licenses.

The paper and e-versions of Statements are held at he faculty and are not included into the printed thesis.

*Change is the end result of all true learning.*

*Leo Buscaglia*

*Желела бих да се најискреније захвалим свима који су на било који начин допринели у узради овог рада - хвала на помоћи, саветима, подршци, стрпљењу и мотивацији.*

*Овај рад посвећујем својој породици, која је у сваком тренутку била уз мене и својом безграничном љубављу и подршком учинила да истрајем.*

.

# САДРЖАЈ

# СПИСАК СЛИКА

# СПИСАК ТАБЕЛА

# СКРАЋЕНИЦЕ И ПОЈМОВИ

| | |
|---|---|
| ADAS | **A**dvanced **D**river **A**ssistance **S**ystems – *Напредни системи за помоћ возачу* |
| AGL | **A**utomotive **G**rade **L**inux **–** *Оперативни систем заснован на Линуксу проширен спрегама за возило* |
| AI | **A**rtificial **I**ntelligence – *Вештачка интелигенција* |
| API | **A**pplication **P**rogramming **I**nterface – *Апликативна програмска спрега* |
| ASIC | **A**pplication **S**pecific **I**ntegrated **C**ircuit – *Интегрисано коло специфичне намене* |
| ASIL | **A**utomotive **S**afety **I**ntegrity **L**evel – *Аутомобилски безбедносни ниво интегритета* |
| ASPICE | **A**utomotive **S**oftware **P**rocess **I**mprovement and **C**apability d**E**termination – *Побољшање софтверског процеса и одређивање способности у аутомобилској индустрији* |
| AUTOSAR | **AUT**omotive **O**pen **S**ystem **AR**chitecture – *Аутомобилска архитектура отвореног система* |
| CAN | **C**ontroller **A**rea **N**etwork – *Магистрала у возилу.* |
| CBO | **C**oupling **B**etween **O**bject classes **–** *Повезаност објеката* |
| CC | **C**yclomatic **C**omplexity **–** *Циклична сложеност* |
| CD | **C**ode **D**ensity **–** *Густина коментара* |
| CPU | **C**entral **P**rocessing **U**nit – *Централна процесорска јединица* |
| D2B | **D**omestic **D**igital **B**us – *Магистрала у возилу* |
| DDR | **D**ouble Data **R**ate Memory **–** *Меморија двосуструке брзине преноса података* |
| DIT | **D**epth of **I**nheritance **T**ree **–** *Дубина стабла наслеђивања* |
| DL | **D**eep **L**earning – *Дубоко учење* |
| DSP | **D**igital **S**ignal **P**rocessor – *Дигитални процесор сигнала* |

| | |
|---|---|
| HUD | **H**ead-**U**p **D**isplay – *Провидни заслон који приказује податке без потребе да корисници скрећу поглед са свог уобичајеног гледишта* |
| E/E | **E**lectrics/**E**lectronic – *Електрика/Електроника* |
| ECU | **E**lectronic **C**ontrol **U**nit – *Електронска контролна јединица* |
| ESP | **E**lectronic **S**tability **P**rogram **–** *Електронска контрола стабилности мотора* |
| EVE | **E**mbedded **V**ision **E**ngine **–** *Наменско векторско језгро за обраду слике* |
| FPGA | **F**ield-**P**rogrammable **G**ate **A**rray – *Интегрисано коло пројектовано на тај начин да може да се конфигурише од стране крајњег корисника* |
| GPS | **G**lobal **P**ositioning **S**ystem – *Глобални позициони систем* |
| GPU | **G**raphics **P**rocessing **U**nit – *Графичка процесорска јединица* |
| HIL | **H**ardware-**I**n-the-**L**oop – *Симулација са хардвером у петљи* |
| HIS | **H**ersteller **I**nitiative **S**oftware – *Метрике изворног кода* |
| HPL | **H**alstead **P**rogram **L**ength **–** *Халстедова дужина програма* |
| HPV | **H**alstead **P**rogram **V**ocabulary **–** *Халстедов програмски речник* |
| HVOL | **H**alstead **Vol**ume **–** *Халстедов обим програма* |
| I2C | **I**nter **I**ntegrated **C**ircuit – *Синхрона серијска комуникациона магистрала* |
| IEBus | **I**nter **E**quipment **B**us – *Комуникациона магистрала у возилу* |
| IEC | **I**nternational **E**lectrotechnical **C**ommission **–** *Међународна електротехничка комисија* |
| IPC | **I**nter **P**rocess **C**ommunication – *Међупроцесна комуникација* |
| ISP | **I**mage **S**ignal **P**rocessor **–** *Специјализовани DSP за обраду слике* |
| IVI | **I**n-**V**ehicle **I**nfotainment – *Инфо-забавни системи у возилу* |
| I/O | **I**nput/**O**utput – *улазно/излазни* |
| ISO | **I**nternational **O**rganization for **S**tandardization – *Међународна организација за стандардизацију* |
| LCOM | **L**ack of **Co**hesion in **M**ethods **–** *Недостатак кохезије у методама* |
| LIN | **L**ocal **I**nterconnect **N**etwork – *Серијски мрежни протокол* |
| LOC | **L**ines **O**f **C**ode **–** *Број линија кода* |
| LTE | **L**ong-**T**erm **E**volution – *Стандард за бежичну широкопојасну комуникацију* |
| McCC | **Mc**C**abe**’s **C**yclomatic **C**omplexity **–** *Циклична сложеност* |

| | |
|---|---|
| MI | **M**aintainability **I**ndex **–** *Степен сложености одржавања софтвера* |
| MICD | **M**aintainability **I**ndex: **C**ode **D**ensity **–** *Степен сложености одржавања софтвера: верзија са густином коментара* |
| MIMS | **M**aintainability **I**ndex: **M**icro**s**oft version – *Степен сложености одржавања софтвера: Microsoft верзија* |
| MOST | **M**edia **O**riented **S**ystems **T**ransport – *Мултимедијална магистрала у возилу* |
| NCL | **N**umber of **Cl**asses **–** *Број класа* |
| NEN | **N**umber of **En**ums **–** *Број енумерација* |
| NOC | **N**umber **o**f **C**hildren **–** *Број подкласа* |
| NOI | **N**umber of **O**utgoing **I**nvocations **–** *Број одлазних позива* |
| OEM | **O**riginal **E**quipment **M**anufacturer – *Произвођач аутомобила* |
| OS | **O**perating **S**ystem – *Оперативни систем* |
| PCIe | **P**eripheral **C**omponent **I**nterconnect **E**xpress – *Стандард брзе серијске магистрале за повезивање периферних уређаја* |
| POSIX | **P**ortable **O**perating **S**ystem **I**nterface **–** *Програмска спрега оперативног система* |
| PU | **P**rocessing **U**nit – *Процесорска јединица* |
| QM | **Q**uality **M**anagement – *Управљање квалитетом* |
| RFC | **R**esponse set **F**or **C**lass – *Број одговора класе* |
| RTE | **R**un**T**ime **E**nvironment – *Окружење извршавања* |
| SAE | **S**ociety of **A**utomotive **E**ngineers – *Удружење инжењера аутомобилске индустрије* |
| SDK | **S**oftware **D**evelopment **K**it – *Софтверски развојни алат* |
| SoC | **S**ystem **o**n **C**hip – *Систем на интегрисаном колу* |
| SOME/IP | **S**calable service-**O**riented **M**iddlewar**E** over **IP** - *Комуникациони протокол у уграђеним системима који подржава удаљене позиве* |
| SOTIF | **S**afety **O**f **T**he **I**ntended **F**unctionality **–** *Стандард за безбедност предвиђене функционалности* |
| SPI | **S**erial **P**eripheral **I**nterface – *Серијска периферна спрега* |
| TARA | **T**hreat **A**nalysis and **R**isk **A**ssesment – *Анализа сигурносних претњи и процена ризика* |
| UART | **U**niversal **A**synchronous **R**eceiver/**T**ransmitter – *Универзални серијски асинхрони протокол за пренос података* |
| USB | **U**niversal **S**erial **B**us – *Универзална серијска магистрала* |

| | |
|---|---|
| V2X | **V**ehicle-**to**-Everything – *Комуникација возила са свима (инфраструктуром, мрежом, другим возилом, пешаком, уређајем)* |
| VAN | **V**ehicle **A**rea **N**etwork – *Магистрала у возилу* |
| VM | **V**irtual **M**achine – *Виртуална машина* |
| VMM | **V**irtual **M**achine **M**onitor – *Контролор виртуалне машине, тј. хипервизор* |
| VPU | **V**ision **P**rocessing **U**nit – *Процесорска јединица за обраду слике/видеа* |
| WMC | **W**eighted **M**ethods per **C**lass **–** *Сложеност пондерисаних метода* |
| YUV | **YUV** простор боја – *Y: лума/осветљеност, U: плава пројекција, V: црвена пројекција* |

# САЖЕТАК

У оквиру дисертације дат је предлог архитектуре средњег слоја софтвера за рачунарски систем у модерним возилима. Мотивација за овај рад лежи у чињеници да је тренутно у аутомобилској индустрији присутан тренд преласка на једну или више централних платформи.

Типична електронска управљачка јединица (енгл. *Electronic Control Unit*, у даљем тексту *ECU*) у возилу обавља једну функцију. Са порастом броја функција током година расте и број *ECU*-ова. Сем пораста броја *ECU*-ова, хардвер и софтвер постају све сложенији захваљујући савременим технологијама. Приметан је недавни раст у развоју софтвера за инфо-забавне системе, системе напредне помоћи возачу и кластер системе. Такође је приметан раст поновног коришћења већ постојећих технологија потрошачке електронике и њихова примена у возилима.

Због разноврсне природе система, произвођачима аутомобила постаје све теже да рукују *ECU*-овима, због чега се јавља растући захтев за консолидацијом *ECU*-ова. Уз доступност софистицираног хардвера, моћни системи на чипу (енгл. *SoC*) у спрези са брзим магистралама и напредним сензорима, се могу користити за реализацију више функција у возилу. У аутомобилској индустрији у току је процес преласка на мањи број *ECU*-ова. Функције се иницијално групишу у склопу домена, а онда теже ка интеграцији на једну или више централних платформи. У склопу овог преласка, потребно је прво консолидовати софтверски слој, а затим софтвер пренети на исту хардверску платформу.

У оквиру дисертације дат је преглед стања у области централних интеграционих платформи, кључни технолошки моменти који су довели до појаве централних интеграционих платформи, као и софтверске платформе које

представљају први корак ка преласку на централне платформе. Посебна пажња је посвећена истраживању софтверских платформи које омогућавају лакши развој апликација у системима за помоћ возачу. Овим платформама се постиже раздвајање софтвера од хардвера коришћењем средњег слоја који апстрахује хардверске могућности и пружа их доступним преко функција и сервиса користећи стандардизовану програмску спрегу.

Циљ ове дисертације је да на основу истраживања дефинише предлог архитектуре средњег слоја софтвера у возилима у једном таквом хетерогеном окружењу, што представља корак напред у консолидацији софтвера кроз пружање спрега за развој апликација на разним хардверским платформама.

На основу предложене архитектуре, средњи слој је реализован на више платформи у сврхе испитивања предложене архитектуре. У оквиру испитивања врши се провера функционалности, перформанси решења, као и могућности бржег и лакшег развоја апликација у хетерогеном окружењу у возилу кроз атрибуте квалитета софтвера. Такође је дат приказ реализованих система апликација, показујући могућности рада у реалном времену.

У оквиру дисертације је приказана могућност примене предложене архитектуре са основним доприносом пружања спреге за бржи и једноставнији развој и интеграцију апликација.

# ABSTRACT

This PhD thesis proposes software middleware architecture for modern heterogeneous in-vehicle environments. The motivation for this work is based on the currenty rising trend in the automotive industry - to move to one or more central platforms.

Typical ECU (*Electronic Control Unit*) usually performs single function in vehicle. With the rising number of functions through years, number of ECUs in vehicles is also constantly rising. Modern technologies lead to more and more sophisticated hardware, while in parallel software becomes more and more complex. There is an evident recent software growth in the infotainment, cluster and advanced driver assistance systems, as well as higher reusability of existing consumer electronics technologies in vehicles.

With the rising system complexity, it becomes harder for OEMs (*Original Equipment Manufacturer*) to manage all these ECUs. As a consequence, there is a request for ECUs consolidation. Availability of the sophisticated hardware and powerfull system on chips, together with the fast buses and advanced sensors, can be used for the implementation of multiple vehicle functions. Accordingly, there is ongoing transition period in automotive to reduce the overall number of ECUs in the vehicle. Functions are initially grouped based on domains to which they belong, with a trend toward integration on single or more central platforms. During this transition, there is a need to first consolidate software on multiple hardware platforms, and then to transit to the same hardware platform.

This thesis provides an overview of the field of central integration platforms, key tehchnological moments that led to appearance of the central integration platforms and software platforms that represent a first step towards the transition to the central platforms. Special attention is given to the software platforms that enable easier

development of application in the area of driver assistance systems. These platforms achieve the separation of software from hardware, using a middleware that abstracts hardware capabilities and makes them available through unified interfaces (functions and services).

The goal of this thesis is to propose software middleware architecture in heterogeneous vehicles. This presents one step forward in software consolidation, giving interfaces for aplication development on various hardware platforms.

Based on the proposed architecture, middleware is implemented on multiple hardware platforms in order to evaluate proposed architecture. The validation tests evaluate the functionality, performance, as well as the possibility of faster and easier development of applications in a heterogeneous environment in the vehicle through the attributes of software quality. An overview of the implemented complete realistic systems is also given, showing the real time solution capabilities.

The dissertation presents the possibility of usage of the proposed architecture, with the basic contribution of providing interfaces for faster and easier application development and integration.

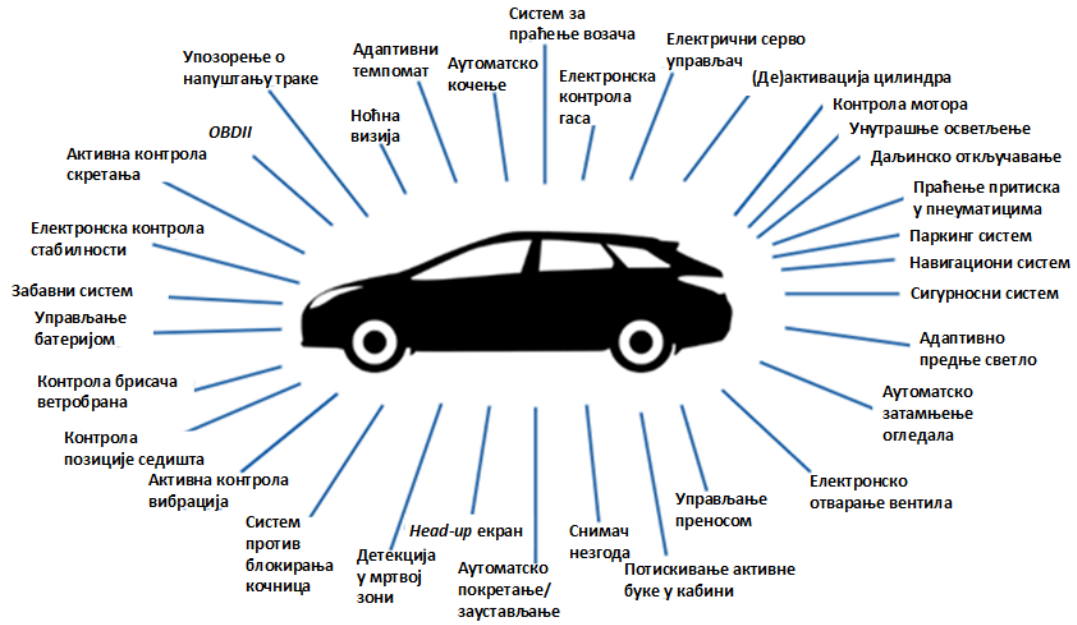
# 1. УВОД

Технолошка и инжењерска окосница возила двадесетог века је био мотор. Седамдесетих година у развоју возила преовлађивали су механички системи, док је осамдесетих година електроника била присутна углавном у доменима шасије и контроле мотора. Појавом система за информисање и забаву, деведесетих година, уноси се нова димензија развоја софтвера у аутомобилској индустрији. Развој софтвера добија све више на важности, тако да данас софтвер контролише многе функције. Улогу технолошке и инжењерске окоснице сем софтвера, преузимају и велика процесна моћ рачунарских система и напредни сензори, који заједно омогућавају најсавременије иновације: од умрежености до аутономне вожње, електрификације и нових мобилних решења.

Развој напредних система за помоћ возачу – *ADAS* (енгл. *Advanced Driver Assistance Systems*) и корак напред ка даљој аутономији вожње повећавају количину обраде и протока података у возилу. Поред тога, системи за информисање и забаву у возилу постају много сложенији и богатији функцијама, док информациони системи возача, као што су модерни дигитални кластер инструменти, *HUD* системи (енгл. *Head-up Display*) и ретровизори са камерама, захтевају екране који ће значајно променити дизајн унутрашњости возила. Електрификација возила такође доноси додатне захтеве за обраду, укључујући контролу мотора и могућност надзора за управљање енергијом и складиштењем батерија у возилу. Многе функције све више међусобно врше интеракцију између разних домена, доприносећи све већем расту сложености рачунарских система у возилу.

Модерна возила се састоје од великог броја електронских контролних јединица (енгл. *Electronic Control Unit - ECU*, у даљем тексту *ECU*) које реализују мноштво функција. *ECU* се може посматрати као мали ‘рачунар’ који води рачуна о одређеном аспекту возила, као што су клима, седишта, или мотор. *ECU*-ови су међусобно повезани разним магистралама (као што су *CAN* [1], *FlexRay* [2], *LIN* [3]) и самим тим чине дистрибуирани систем у возилу. Типично возило на путу данас већ има десетине рачунарски контролисаних функција (Слика 1.1), док возила

премијум класе на пример имају око 70 *ECU*-ова, повезаних на пет системских магистрала, реализујући преко 800 функција у возилу [4].




*Слика 1.1 Типичне ECU функције у данашњим возилима*

Потреба за већим бројем функција и повећан број иновација доводе до све већег садржаја софтвера у возилима. Самим тим софтвер добија све већу важност у аутомобилској индустрији. Према [4] до 40% цене возила је одређено електроником и софтвером, док је 50%-70% цене развоја *ECU*-а везано за софтвер.

Сви поменути аспекти утичу на електронику у возилу што доводи до све већег броја захтева по питању рачунарског система у возилу, даље доводи до промене архитектуре возила, као и наглашавања сигурности и безбедности. Данашњи изазов представља управљање растућим бројем функционалних захтева унутар једног и између више домена. Самим тим E/E (енгл. *Electrics/Electronics*) архитектуре које представљају међусобно повезане *ECU*-ове (хардвер и софтвер) морају да се развијају. Произвођачи аутомобила су почели да представљају идеје свеобухватних архитектура од којих је једна и централна интеграциона платформа која полако замењује дистрибуирану архитектуру у којој су традиционална решења прављена углавном испочетка за одређеног произвођача аутомобила (енгл. *Original Equipment Manufacturer,* у даљем тексту OEM). Централна интеграциона платформа се састоји од једног или више рачунара опште намене, као и рачунарских јединица високих перформанси за обраду података које комуницирају

преко интерних брзих магистрала. Прелазак на централну интеграциону платформу захтева изналажење динамичке софтверске платформе тј. прво прелазак на софтверском нивоу, а затим прелазак на хардверском нивоу.

## 1.1. ПРЕДМЕТ ИСТРАЖИВАЊА

У оквиру дисертације биће истражена област централних интеграционих платформи, кључни технолошки моменти који су довели до појаве централних интеграционих платформи, као и софтверске платформе које представљају први корак ка преласку на централне платформе. Посебна пажња биће посвећена истраживању софтверских платформи које омогућавају лакши развој апликација у системима за помоћ возачу. Овим платформама се постиже раздвајање софтвера од хардвера коришћењем средњег слоја који апстрахује хардверске могућности и пружа их доступним преко функција и сервиса користећи стандардизовану програмску спрегу.

## 1.2. ЦИЉ ИСТРАЖИВАЊА

Истраживање се врши са циљем да се обезбеди средњи слој софтвера као први корак у процесу преласка на једну или више централних платформи, што доноси бржи и једноставнији развој апликација у окружењима са присуством хетерогених платформи. Предложено решење треба да обрати пажњу на изазове једне такве централне рачунарске платформе у возилима.

Хипотеза на којој се темељи истраживање је да је могуће предложити архитектуру која систематском анализом стања у области и доступних релевантних информација доноси организацију и раслојавање новог софтверског слоја за примене у извршавању апликација за нова возила у складу са описаним проблемима.

Квалитет решења ће се оценити одговарајућим метрикама преко оцене квалитета реализованог средњег слоја: ХИС метрике (енгл. *Hersteller Initiative Software*), објектно-оријентисане метрике и метрике квалитета атрибута софтвера (степен сложености одржавања софтвера, портабилност, и проширивост). Верификација ће такође бити обављена кроз низ експерименталних сценарија и

кроз примере примене решења у реалним корисничким случајевима у реалном времену.

## 1.3. НАУЧНИ ДОПРИНОС

Научни допринос дисертације се односи на област централних интеграционих платформи у модерним окружењима возила. Резултати дисертације доприносе идентификацији кључних аспеката за потребе развоја апликација у централној интеграционој платформи.

Очекивани резултат истраживања у оквиру докторске дисертације је изналажење решења које пружа једноставнији начин развоја апликација у хетерогеном окружењу, где онај који пише апликацију не мора да води рачуна о томе где се који део апликације извршава и како се подаци допремају између процесорских јединица, а са друге стране има начин да на једноставан начин интегрише и покрене своје решење на једној или више циљних платформи.

Дефинисани су основни софтверски блокови, као и њихове спреге који могу да послуже као основа за будућа истраживања и унапређења.

Додатно, реализација софтверске платформе је спроведена на одређеном броју хардверских платформи, које могу да послуже као основа за брзи развој и интеграцију прототипа апликација и система.

Резултати вишегодишњег истраживања који су део ове дисертације објављени су у истакнутом међународном часопису [5] и техничким решењима [6], [7].

## 1.4. ОРГАНИЗАЦИЈА ДИСЕРТАЦИЈЕ

Дисертација је организована у седам поглавља и три додатка.

Уводно поглавље представља предмет и циљ истраживања, научни допринос и приказује организацију дисертације.

У другом поглављу је дато тренутно стање у области, преглед и еволуција E/E архитектура за рачунарске системе у возилима са посебним освртом на прелазак на централне рачунарске системе. Такође је дат опис релевантних информација за израду ове тезе. Приказане су теоријске основе централних

рачунарских система са елементима које носе са собом, са посебним осврtом на постојећа решења средњег слоја софтвера у рачунарским системима у возилу. Представљена је и анализа нефункционалних аспеката: безбедносних, сигурносних и архитектурних аспеката.

Треће поглавље представља анализу захтева и предлог архитектуре средњег слоја софтвера за рачунарски систем у возилима. Такође су размотрена ограничења у погледу сложености, квалитета и могућности предложеног решења.

Четврто поглавље даје предлог мера за оцену квалитета решења укључујући експерименталне тестове, мере за перформансе, софтверске мере, и мере атрибута квалитета софтвера.

Референтни пример реализације предложеног средњег слоја је дат у петом поглављу, са приказом одговарајућег окружења (платформи) на којима је решење реализовано.

У шестом поглављу су дати резултати мерења. Експериментална провера је извршена на неколико различитих платформи, да би се испитала платформска независност предложеног решења. Такође су измерене перформансе решења. У оквиру провере решења дати су резултати оцена квалитета решења у односу на мере које су предложене, као и поређење предложеног решења са другим решењима.

У седмом поглављу, дат је закључак истраживања са осврtом на правце даљег развоја.

Даље су приложени додаци који садрже преглед постојећих софтверских решења и иницијатива, преглед референтних коришћених тестних корисничких случајева, као и преглед софтверских метрика.

На крају ове дисертације се налази списак литературе коришћене при њеној изради.

# 2. ПРЕГЛЕД ОБЛАСТИ И ПОСТАВКА ЦИЉЕВА ИСТРАЖИВАЊА

У овом поглављу је дат преглед развоја области истраживања и поставка циљева истраживања. Преглед области дат је кроз еволуцију Е/Е архитектура, укључујући и визију како ће се даље архитектуре развијати. Посебан акценат је стављен на тренутне Е/Е архитектуре, њихову трансформацију, као и трендове који су присутни и изазивају ту исту трансформацију. Представљена су и кључна технолошка достигнућа која омогућавају овај степен еволуције Е/Е архитектура.

Дат је приказ области централних интеграционих платформи као једних од кључних момената у овој еволуцији, укључујући теоретске основе разних аспеката, од окружења: хардверске платформе, сензори и актуатори, магистрале у возилу, преко типова софтверских платформи и хипервизора, до нефункционалних аспеката као што су безбедносни, сигурносни и архитектурни.

Затим је дат преглед постојећих решења средњих слојева у овој области, укључујући решења из индустрије и академије. На крају је дата поставка циљева истраживања.

## 2.1. Е/Е АРХИТЕКТУРЕ

### 2.1.1. Изазови за модерне Е/Е архитектуре

Основни покретачи развоја аутомобилске индустрије који убрзано мењају захтеве дизајна возила и архитектуре, а самим тим утичу и на софтверске захтеве, укључују захтеве и очекивања тржишта, технолошки напредак и стратегије произвођача аутомобила и њихових добављача.

Очекивања тржишта снажно утичу на функције возила, а самим тим и на њихову архитектуру. Потражња за технологијама погонских система не укључује само традиционални мотор са унутрашњим сагоревањем, већ и електрични, хибридни и погонски систем горивих ћелија. Тржиште такође очекује високо

интегрисане функције као што је аутономна вожња [8]. Да би се то омогућило потребне су значајне промене у E/E архитектурама како би се задовољили захтеви везани за процесну моћ и функционалну безбедност. Захтеви не иду само у погледу процесне моћи *ECU*-ова него и ка побољшању технологија сензора и актуатора, при томе одржавајући функционалну безбедност. Скалабилне E/E архитектуре које пружају флексибилност при интеграцији софтвера од разних добављача ће ефикасно омогућити интеграцију иновација, омогућавајући значајну стратешку предност за произвођаче.

Тржиште такође захтева већу персонализацију и могућност прилагођавања у технологијама удобности и сигурности. Ове технологије се значајно разлику по својим особинама и захтевима, што захтева већу флексибилност E/E архитектура како би се подржала варијабилност. За идеалне будуће E/E архитектуре, ово би требало да представља минимални напор, и самим тим доводи до смањења времена за увођење нових иновација на тржиште.

Повећан је и број захтева за комуникацију са удаљеним уређајима и/или облаком. Напредни системи помоћи возачу, системи за информисање и забаву, и телематика могу захтевати повезаност са другим системима ван возила, што може довести до стварања више од једне везе између возила и облака. Такође, ажурирање софтвера на *ECU*-овима се помера из сервисних центара ка ономе на шта смо навикли на мобилним уређајима. Вишеструке везе повећавају изложеност возила нападима, чиме је смањена сигурност возила. Обезбеђивање сигурности у мрежама унутар возила и ван возила доноси нове изазове по питању сложености и скалабилности.

Након испоруке возила, појављују се захтеви за дугорочном подршком. Купци очекују да се софтвер ажурира и одржава као и на другим уређајима које поседују. Могућност прилагођавања постојећих функција или развоја нових функција доводи до постављања нових дигиталних пословних модела. Јавља се потреба за сервисно-оријентисаном архитектуром, у односу на досадашњу архитектуру која је усмерена на функције. Она омогућава како прилагодљивост, тако и осигурање постојања ствари које се разликују од других сервиса.

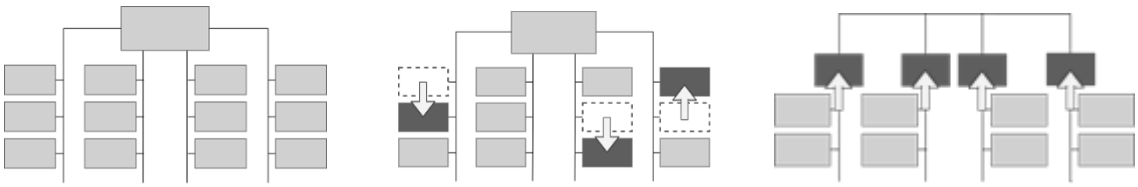
У последње време све је више актуелна прича о вештачкој интелигенцији (енгл. *Artificial Intelligence*) у возилима. Отвара се велики потенцијал за читање и

анализирање података са возила. Предвиђа се да ће возила будућности бити део система који узајамно делује унутар свеобухватног система система. Са становишта дизајна, кључни аспекат овога ће бити робустност система и могућност система да се избори са променама у свом окружењу.

### 2.1.2. Еволуција E/E архитектура

Изазови које доноси тржиште, технолошки напредак и стратегије произвођача доводе до развоја E/E архитектура. E/E архитектуре представљају међусобно повезане *ECU*-ове (хардвер и софтвер) у возилима. Еволуција E/E архитектура креће од дистрибуираних и тренутно тежи ка преласку на централизоване архитектуре.

Првобитна E/E архитектура била је модуларна где је један *ECU* руковао једном системском функцијом (Слика 2.1а.). Развојем контролних јединица веће процесне моћи и повећањем броја функција, кренуло се ка интеграцији неколико сличних функција у један *ECU*, са главним фокусом на смањење броја *ECU*-ова и самим тим и смањење каблирања у возилу, што је знатно утицало на цену (Слика 2.1б.). Сваки *ECU* је реализовао независне или слабо спрегнуте функције, које су биле повезане магистралама размењујући контролне поруке. Оваква децентрализована архитектура је широко прихваћена и омогућава раздвајање одговорности, велику флексибилност због слабо спрегнутих хардверских компоненти, као и једноставну верификацију услова за интеграцију у систем.



*а)* *б)* *в)*

*Слика 2.1 E/E архитектура: а) модуларна, б) интегрисана, в) доменска*

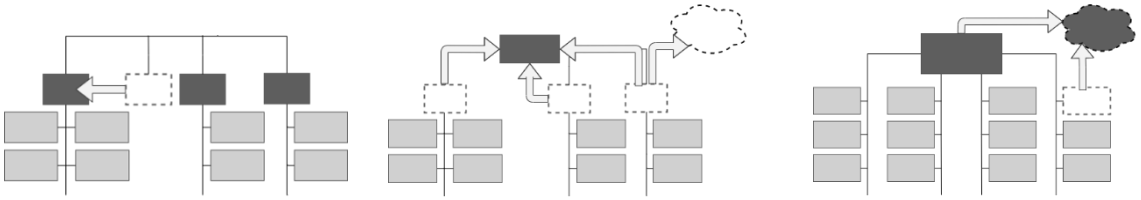
Додавањем нових функција расте и број електронских контролних јединица у возилу, као и број веза међу њима. Све већа сложеност функција (функције постаје тешко реализовати на једном *ECU*-у и захтевају повећану спрегнутост са другим функцијама) доводи до следећег већег степена интеграције у облику

централних доменских управљачких јединица које контролишу одређени домен у возилу (Слика 2.1в.). Доменске централне управљачке јединице могу бити посебне контролне јединице са већом процесном моћи, или већ постојећи *ECU* у возилу. Домени омогућавају одговарајуће груписање/раздвајање *ECU*-ова према њиховој функционалности и олакшавају приступ контролним јединицама које се односе на сличне функције. У оквиру истог домена присутна је већа интеракција између електронских контролних јединица. Типични домени су на пример систем за информисање и забаву, шасија, систем за погон, напредни системи за помоћ возачу - АДАС [9].

Ако за доменски пример узмемо АДАС, у возилу постоје разни АДАС сензори који прво обрађују податке и о резултатима комуницирају са *ECU*-овима. Ово је ефикасно за делимично аутоматизоване функције, као што је адаптивни темпомат. За већу аутоматизацију вожње потребан је већи број сензора и могућност изградње модела околине возила користећи податке са тих сензора. Неки од ових сензора ће радити заједно и обрађивати податке, пружајући информације другим сензорима. Ово води у делимично централизовани домен где неколико скупова сензора обезбеђују окружење једни другима. За омогућавање високе аутоматизације функција вожње, постоји потреба за централизованим системом фузије сензора који обједињује податке које прима са сензора и врши симултану локализацију и мапирање функција. Такав централизовани *ECU* који ради као глава АДАС домена представља добар пример доменског *ECU*-а. У индустрији већ постоје одређене реализације оваквог доменског *ECU*-а [10].

Даљи развој и интеграција, воде ка интеграцији уско повезаних домена, стварајући више-доменске *ECU*-ове (Слика 2.2а.). Интеграција доменских *ECU*-ова омогућава транспарентнији рад међу-доменских функционалности. На пример, за потребе високо аутоматизоване вожње користи се више домена који се сматрају различитим доменима: шасија, погонски систем, АДАС домен. Јавља се потреба за чвршћом везом између домена шасије и домена погона и њихових одговарајућих сензорских и актуаторских могућности. Такође, домени за погонски систем и за помоћ возачу траже блиску сарадњу. Електронска контролна јединица *ADAS* домена ће бити веза између сензора и уграђених функција у домену шасије. Након тога, погонски домен и домен шасије ће се спајати како би оформили домен

контроле кретања возила. Овај домен ће сарађивати са АДАС доменом и системом управљања енергијом како би контролисао возило.



*а)* *б)* *в)*

*Слика 2.2 E/E архитектура: а) више-доменска, б) са централним рачунаром, в) са обрадом у облаку*

У [11] се предвиђа да ће се овај тренд наставити ка стварању централног рачунара у возилу који ће да преузме главну улогу (Слика 2.2б.). Даље, са бољом умреженошћу возила порашће могућност коришћења ресурса у облаку за обраду захтевних функција у возилу (Слика 2.2в.).

Централизоване функције возила тј. централни рачунар подразумевају један извор информација који се може обрадити интерно на *ECU*-у користећи напредне прорачуне. Смањени број *ECU*-ова подразумева смањење сложености мрежа у возилу, а самим тим и уштеду на трошковима за ожичење каблова што је један од важнијих фактора. Интеграцијом софтвера на исти *ECU*, временска ограничења у систему се лакше постижу (у времену интеграције не мора се проћи успоравајући транспортни слој). У зависности од могућности *ECU*-а и архитектуре система, опажени догађаји као што су сигнали видео камера, могу се евалуирати много брже, ако се обрада улазног сигнала, одлучивања и излазног сигнала налази у једном систему. Ипак физичка веза са већином актуатора и сензора мора постојати.

Централизовање софтверске одговорности и обраде на појединачном *ECU*-у повезано је и са одређеним недостацима. Очигледан проблем је потреба за способним хардверским контролером високих перформанси. То подразумева скупе хардверске платформе. У случају оштећења, вредна компонента се мора заменити. Ово отежава руковање превеликим системима по питању флексибилности и скалабилности. Гледано из безбедносне перспективе, ово значи губитак редундантности, што се мора решити у складу са одговарајућим стандардима за возила.

Даље ћемо се фокусирати на тренутно актуелно стање, а то су Е/Е архитектуре које иду ка томе да се састоје од високо интегрисаних доменских контролера и/или једног или више централних рачунарских система, који су међусобно повезани брзим магистралама. У наставку су излистани недостаци тренутних архитектура и кључни технолошки моменти који омогућавају пут ка даљој централизацији.

### 2.1.3. Недостаци данашњих Е/Е архитектура

Реализација функционалности возила нове генерације у обрасце данашње дистрибуиране Е/Е архитектуре се суочава са неколико недостатака:

- *Пропусни опсег комуникационих канала* - Пропустност магистрала унутар домена и између домена није довољна за транспорт свих података који ће требати у будућности. Комуникационе магистрале у возилу су примарно засноване на *CAN*-у [1], уз постојање *FlexRay* [2], *LIN* [3] и *MOST* [12] магистрала. Те магистрале нису довољно ефикасне у погледу захтева за аутономну вожњу и умреженост: већи пропусни опсег, мање кашњење и већа сигурност.
- *Екстерна комуникација (умреженост)* - Умреженост доводи до протока све веће количине података и повећава сигурносне захтеве, посебно оне који се тичу комуникације возила са спољним светом.
- *Повећана варијабилност* - Нове функционалности упарене са разним погонским системима захтеваће Е/Е архитектуре које подржавају руковање разним варијантама које је потребно подржати. Овим ће произвођачи имати широк спектар понуде и смањити трошкове руковања варијантама разним архитектурама.
- *Могућности рачунарске платформе (процесна моћ)* - Нове функционалности захтевају *ECU*-ове са већим могућностима. Даља интеграција функција у возилима ће захтевати да *ECU*-ови имају софтверске механизме који омогућавају безбедно одвајање функција преко механизама хардверских виртуализација, нпр. хипервизора [13].
- *Проширивост у прихватљивом времену* - Будући Е/Е системи треба да подрже лаку интеграцију нових технологија у постојећу архитектуру. Типичан развојни циклус од пет до седам година неће бити одржив за

увођење иновација у аутомобилске E/E системе. E/E архитектуре је потребно дизајнирати имајући у виду њихову могућност проширења у прихватљивом времену. Овим се обезбеђује стратешка предност за нове технологије у погледу изласка првог на тржиште.

### 2.1.4. Кључни технолошки моменти

Као решење горе побројаних захтева (аутономна вожња, персонализација, умреженост са облаком, удаљено ажурирање софтвера, сигурност, безбедност, разни погонски системи) и недостатака (пропусни опсег комуникационих канала, варијабилност, умреженост, процесна моћ, проширивост) намећу се централне платформе (једна или више које међусобно комуницирају преко брзих магистрала). Неколико кључних технологија омогућава централне платформе и развој нових функционалности у возилима:

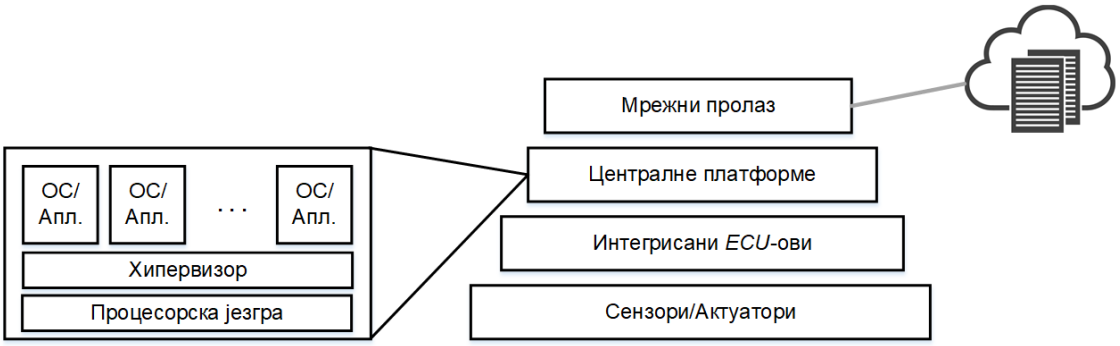
- *Нове магистрале у возилима* - Магистрале у возилима са новим стандардима као што су аутомобилски етернет испуњавају захтеве за већим пропусним опсегом и пружају боље сигурносне и безбедносне мере. Ове особине доприносе омогућавању високо аутоматизоване вожње и високо квалитетних система за информисање и забаву у возилима. Такође, ту је и *PCIe* магистрала која исто тако испуњава захтеве за пропусним опсегом унутар компоненти возила.
- *Напредни мрежни пролаз (енгл. gateway)* - Са поменутим магистралама у возилу, напредни мрежни пролази са новим механизмима усмеравања ће обезбедити мало кашњење и високу пропустност користећи хардверско убрзање. Механизам хардверског убрзања ће бити потребан да би се смањило оптерећење централног процесора увођењем аутомобилског етернета.
- *ECU-ови велике процесне моћи* – Вишејезгарни *ECU*-ови високих перформанси омогућавају интеграцију апликативног софтвера на мањи број *ECU*-ова. То се може постићи користећи методе виртуализације хардвера, као што су хипервизори. Виртуализација хардвера омогућава даљу интеграцију *ECU*-ова.

- *Умрежена/повезана возила* - Са растућим бројем сценарија за повезивање са спољним светом, као што је ажурирање софтвера, акценат ће бити на искоришћавању пуног потенцијала умрежености очувавајући возило сигурним и безбедним од злонамерних напада користећи разне безбедносне механизме.
- *Интеграциона платформа / нови софтверски приступи* - Доменима као што су инфо-забавни системи, системи шасије, и погонски системи ће бити управљано додатним доменским контролером или интеграционом платформом, која пружа велику процесну моћ и делује као мастер домена контролишући и извршавајући доменске обраде вишег нивоа. Обзиром да ће доменски *ECU*-ови обављати главнину рачунања на високом нивоу, тренутне блоковске компоненте ће руководити само основним обрадама везаним за ту компоненту, омогућавајући стандардизоване компоненте. Ниво интеграције доменских контролера зависи од односа између волумена возила и трошкова везаних за развој. Нови софтверски приступи укључују и интеграцију технологија и функција који не потичу нужно из традиционалне аутомобилске индустрије, као што су на пример технологије потрошачке електронике: машинска визија (енгл. *machine vision*) и фузија сензора (енгл. *sensor fusion*).

## 2.2. ЦЕНТРАЛНА ИНТЕГРАЦИОНА ПЛАТФОРМА

Произвођачи аутомобила су кренули да представљају идеје свеобухватних архитектура (укључујући централне платформе) које повезује електронику у возилу са *OEM*-овим услугама у облаку. Општи концепт архитектуре који је присутан када је у питању свеобухватна архитектура је приказан испод (Слика 2.3).

Мрежни пролаз одваја домен корисничког интерфејса (инфо-забавни систем / веза са паметним телефонима) од домена погона (погон, систем кочења, управљање батеријама) и повезује возило са *OEM*-овим системом у облаку користећи тзв. паметну антену. Главни задатак паметне антене и мрежног пролаза је да реализују различите сигурносне слојеве као што су мрежна баријера (енгл. *firewall*) и детекција упада.




*Слика 2.3 Свеобухватна архитектура у возилу*

На врху архитектуре се налазе један или више централних рачунара који комуницирају преко мреже високог пропусног опсега и задужени су за домене возила (као што су инфо-забавни системи, аутономна вожња), као и за комуникацију која повезује централну платформу са интегрисаним *ECU*-овима, сензорима и актуаторима. Централне платформе обављају обимне обраде и рачунице. У оквиру централних платформи реализује се логика и оне представљају потенцијал за диференцијацију између *OEM*-ова.

Под интегрисаним *ECU*-овима се подразумевају *ECU*-ови који се неће интегрисати у централну платформу. Две групе интегрисаних *ECU*-ова се издвајају:

- *ECU*-ови у којима су интегрисане функције специфичне за *OEM*-а, попут контроле мотора или *ESP*-а (енгл. *Electronic Stability Program*).
- *ECU*-ови који контролишу специфични део који се испоручује са скупом стандардних функционалности, попут подизања прозора и сл. Ови *ECU*-ови су ослобођени било каквих *OEM* специфичних функција и софтвера.

На дну архитектуре се налазе сензори и актуатори без којих возило не би функционисало.

Повезивање са системом у облаку омогућава развој нових функција у возилу као што је достављање података о окружењу возилима (услови на путу, бесплатна паркинг места или најновије понуде произвођача возила). Ове услуге, као и опција за омогућавање функција (нпр. системи за помоћ возачима) дају произвођачима возила могућност да остваре приход и након продаје возила, када је возило у употреби, што до сада није био случај. Стална веза са возилом омогућава *OEM*-у да прикупља корисничке податке и на тај начин добије више информација о

поузданости и трошењу појединих компоненти. Извори грешака у хардверу и софтверу и припадајућим подацима околине могу да се открију преко дијагностичке спреге, софтвер се може побољшати код произвођача и возило се може одмах ажурирати - слично ажурирањима апликација за паметне телефоне на које су корисници навикли током година. У овом моделу, функције самог возила све више ће зависити од комуникације у возилу и ка облаку.

Централна интеграциона платформа полако замењује дистрибуирану архитектуру. Састоји од једног или више рачунара опште намене, као и рачунарских јединица високих перформанси за обраду података које комуницирају преко интерних брзих магистрала. За безбедносно критичне функције на чип се интегрише додатно безбедносно језгро или други безбедносни процесор на саму плочу. Како би се опслужили захтеви како премијум возила тако и возила широке производње, скалабилност представља један од захтева за овакву рачунарску архитектуру.

Процес преласка на централну платформу захтева време и не може се десити преко ноћи. Прелазак мора да се деси прво на софтверском нивоу, тј. прво треба да се изврши софтверско-хардверско раздвајање. Након тога следи и транзиција на хардверском нивоу. Први део преласка захтева динамичку софтверску платформу. Стабилна софтверска платформа доводи до значајног смањења количине софтвера који се мора развити и валидирати. Основни софтвер платформе се може поново користити знајући да ће апликације које су већ тестиране радити и на следећој хардверској платформи, чак и ако долази од другог добављача. Наравно, коначни тестови се морају спровести у складу са процедурама у аутомобилској индустрији. Софтверско-хардверско раздвајање се већ дешава у домену инфо-забавних система. Неки инфо-забавни уређаји су у стању да прихвате софтвер од независних добављача (нпр. навигацију у виду апликације).

Што се архитектуре тиче - раздвајање софтвера од хардвера се може постићи коришћењем средњег слоја који апстрахује хардверске могућности и пружа их доступним преко функција и сервиса користећи стандардизовану програмску спрегу.

У наставку овог потпоглавља су дате теоријске основе одређених аспеката који се морају узети у обзир код софтверске централне интеграционе платформе:

хардверске платформе, сензори и актуатори, магистрале у возилу, типови софтверских платформи, хипервизори који омогућавају безбедно раздвајање функција, као и нефункционални аспекти (безбедносни, сигурносни, и архитектурни). Изложени су основни принципи и дат је преглед елемената који утичу на архитектуре средњег слоја софтвера за централне рачунарске системе у возилима.

### 2.2.1. Хардверске платформе

Са развојем технологије доступно је мноштво хардверских платформи како из света аутомобилске индустрије, тако и из других индустрија - првенствено потрошачке електронике. Појављују се процесори високих перформанси који су способни да подрже извршавање великог броја функција и самим тим омогућавају концепт централне платформе у возилу. Хардверске платформе можемо поделити у неколико логичких целина: безбедносна сертификована микроконтролерска решења, хетерогени чипови високих перформанси, доменски контролери и развојне централне платформе.

| Име | Особине |
|---|---|
| *Aurix* [14] | 32-битни заснован на *Tricore* архитектури |
| *Hercules TMS570* [15] | 32-битни заснован на *ARM Cortex-R4F* |
| *Qorivva MPC5643L* [16] | 32-битни заснован на *Power* архитектури |
| *RH 850-P1x* [17] | 32-битни заснован на *RH850* |
| *SPC5 G/ P* [18] | 32-битни заснован на *Power PC* архитектури |

*Табела 2.1 ASIL-D сертификовани микроконтролери*

За безбедносно критичне функције као што су провера веродостојности, праћење и валидација резултата, на чип се интегришу додатна безбедносна језгра или се на плочу интегрише други процесор. Постоји неколико микроконтролерских решења који су сертификовани за *ASIL-D* ниво (Табела 2.1).

Технолошки развој доводи до појаве разноврсних хетерогених, вишејезгарних чипова који омогућавају концепт централне платформе. Ови системи обухватају разне врсте процесорских јединица од графичких, опште наменских, до процесора за дигиталну обраду сигнала, наменских процесора

специфичних за обраду слике, векторског процесирања, па до оних заснованих на *FPGA*.

Чипови засновани на *FPGA* већином су спрегнути са другим, углавном опште-наменским процесорима. Чип *Cyclone* [19] поред *FPGA* садржи и двојезгарни *ARM Cortex A9*, и карактерише га мала потрошња и повећана пропусна моћ. *Zynq UltraScale* [20] поред програмабилне логике садржи више врста *ARM* процесора као и графички процесор.

Чипови засновани на графичким процесорима (високих перформанси) такође су углавном спрегнути са процесорима опште намене као и са додатним акцелераторима. *NVIDIA* је представила целу линију оваквих чипова у *Tegra* фамилији [21]: од *Tegra* 1,2,3,4, *Tegra K1*, *Tegra X1*, *X2*, до *Xavier* и *Orin* чипа. *MobileEye* представља *EyeQ* фамилију чипова [22] од *EyeQ1* до *EyeQ5*, која подржава комплексну и захтевну обраду слике одржавајући малу потрошњу. Чипови се састоје од процесора опште намене и наменских језгара за обраду слике и видеа. Са *EyeQ5* чипом циљају централну платформу за фузију сензора у оквиру аутономне вожње.

*Qualcomm* је представио чип који садржи *CPU*, *GPU* и *DSP* процесор и намењен је за аутомобилску индустрију - *s820a* [23], док *Samsung* преставља *Exynos* [24] који се састоји од *CPU*-а, *GPU*-а и *LTE* модема. *Texas Instruments* представља *TDAx* [25] фамилију хетерогених чипова од којих је најновији *TDA4x* и засновани су на *DSP* процесорима. Поред *DSP*-ова садрже општенаменске процесоре, графичка језгра, као и наменска језгра и хардверске блокове за обраду слике.

На тржишту су присутни и чипови који садрже наменске процесоре који се фокусирају на рачунарску визију и дубоко учење као што су *CV2 4K* чип [26] и *Ceva-XM6* [27]. *CV2 4K* поред *CVflow* архитектуре и стерео-визије такође садржи и *DSP* екстензију.

Чипови засновани на општенаменским процесорима представљају вишејезгарне и многојезгарне процесоре. *Intel* представља *Xeon* и *Atom* вишејезгарне процесоре са високим перформансама. Вишејезгарни *ARM Cortex*-и присутни су у више чипова: *I.MX8* фамилија [28] која поред процесора опште намене садржи опционо *DSP* и *GPU* језгро, затим *R-Car H3* [29] који поред *ARM Cortex*-а садржи и *3D* графички процесор.

Постојећа решења за доменске контролере се фокусирају на домен и у складу са тим је и избор чипова који ће се наћи на тим платформама. На пример, платформе за инфо-забавне системе се углавном састоје од чипова високих перформанси способних за графичку обраду. *MIB2* [30] модуларна платформа за инфо-забавне системе у себи садржи *Tegra K1* чип.

Платформе за помоћ возачу сем једног или више чипова високих перформанси способних да врше захтевне обраде у реалном времену у себи садржи углавном и безбедносни процесор. Пример је *zFas* [10] централни контролер за помоћ возачу који садржи *Tegra K1*, *EyeQ3, Cyclone* и *Aurix.*

Разни произвођачи су већ почели да рекламирају централне развојне хардверске платформе у аутомобилској индустрији. У табели испод су излистане неке од централних платформи као и којој области су намењене и које чипове садрже. Приметно је да се већина састоји од једног или више хетерогених, вишејезгарних процесора повезаних брзим етернет магистралама. За безбедносно критичне функције (као што су провера веродостојности, праћење и валидација резултата) присутан је и један безбедносно сертификовани микроконтролер.

| Име | Особине |
|---|---|
| *GO* [31] | Развојна платформа за аутономну вожњу<br>Чип(ови): 2 x *Xeon* или *Atom*, *ASIC/Arria*, *Aurix* |
| *ProAI Gen 3* [32] | Платформа за аутономну вожњу<br>Чип(ови): 3 x *NVIDIA Xavier/Xilinx*, безбедносни микропроцесор |
| *Razor Motion* [33] | Платформа за аутономну вожњу<br>Чип(ови): 2 x *R-Car H3, RH850/P1H-C* |
| *TTA Drive* [34] | Развојна платформа за системе за помоћ возачу<br>Чип(ови): *Aurix*, скалабилни процесори високих перформанси |

*Табела 2.2 Централне развојне хардверске платформе*

### 2.2.2. Сензори и актуатори

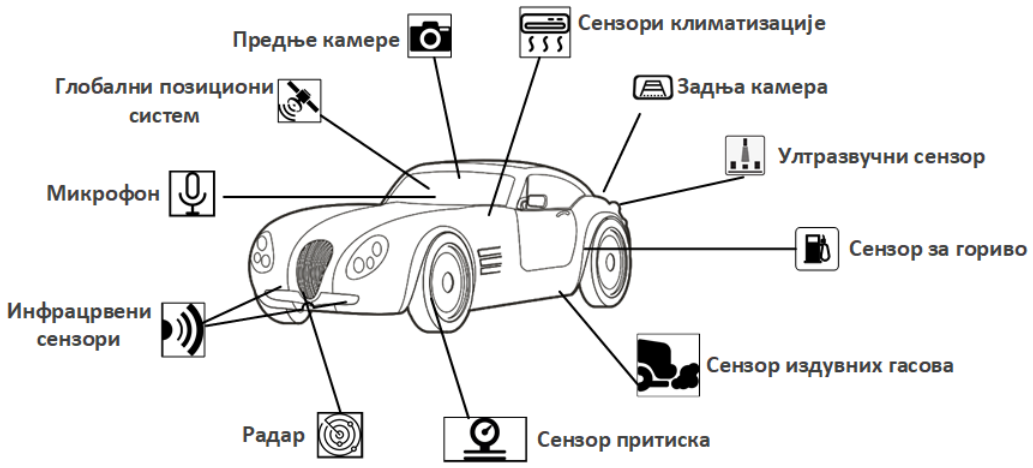
Сензори и актуатори су уско интегрисани са *ECU*-овима и тиме помажу побољшању перформанси возила, његовој поузданости и трајности. Сензор представља уређај задужен за откривање и претварање физичких и хемијских величина у електронске сигнале или импулсе. Може се пронаћи стотине аутомобилских сензора који мере разне врсте величина, као што су температура, доток ваздуха у мотор, удаљеност од препреке, итд. Актуатор је генерални назив за

управљачки механизам. Широк избор актуатора се налази у аутомобилима. Неки од њих су: соленоиди, истосмерни електро-мотори, корачни мотори, итд.

Критеријуми селекције сензора су разни у односу на захтеве и крајњу сврху сензора. Карактеристике рада сензора се могу сврстати у четири групе:

- *Статичке особине* - описују перформансе у односу на споре промене (статичко окружење),
- *Динамичке особине* - описују перформансе узевши у обзир варијације у времену и у количини која се мора мерити,
- *Особине везане за окружење* - перформансе сензора током и након излагања одређеним спољним условима,
- *Особине поузданости* - описивање очекиваног животног века сензора.

Поред описа перформанси важне су и карактеристике грешке сензора, као што су осетљивост, нелинеарност и хистерезис, резолуција, опсег, тачност, помак и нулти помак, шум, време одзива и фреквенцијски одзив. Такође, ту су и друге особине које се морају задовољити: цена, величина, комуникација, напајање.




*Слика 2.4 Разни типови сензора у возилу*

Појавом захтева за аутономном вожњом, возило мора да буде свесно своје околине. Присуство сензора и актуатора је кључ аутономне вожње како би се спречило да људска интеракција буде потребна током вожње. Десетине сензора (Слика 2.4) су потребне како би се све потребне информације од окружења прикупиле. Сада се не говори само о детекцији једне препреке него о свесности возила целог окружења. За то се користе сензори као што су камере, радари, лидари,

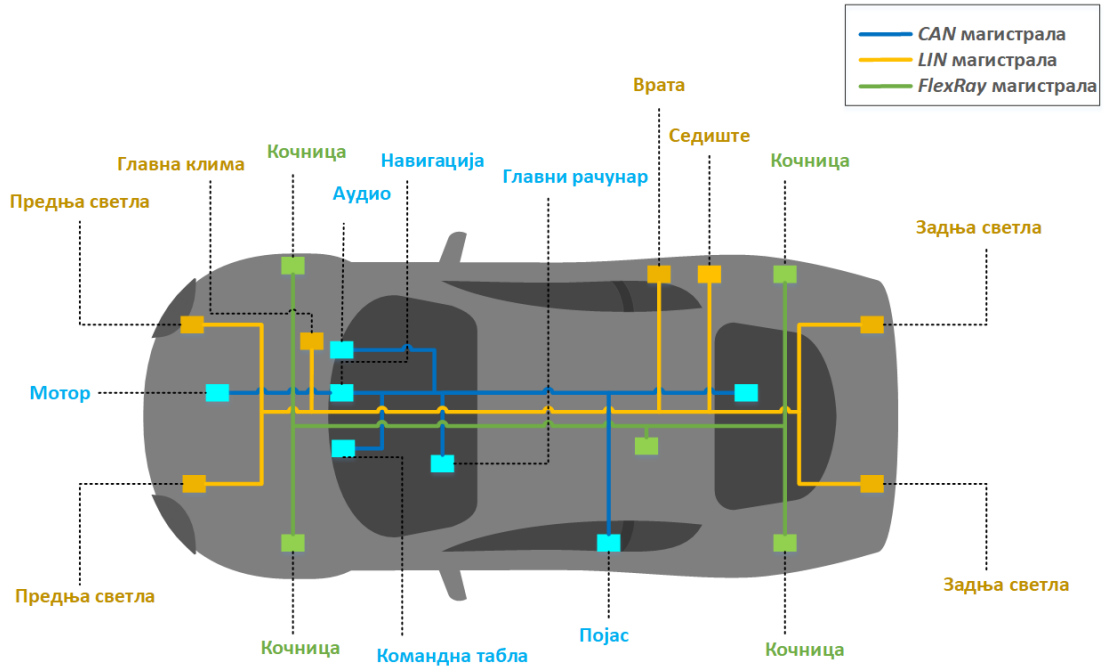
ултразвучни сензори, сензори система глобалног позиционирања који могу да генеришу мноштво података (Табела 2.3) и које је потребно обрадити на једном месту како би се донео поуздан закључак на основу не једног, него више сензора. Долази до потребе за централизовањем обраде података са сензора и присуства централне рачунарске платформе. Постојећи дистрибуирани системи били су довољни за руковање сензорима који нису генерисали мноштво података и док те податке нису захтевали многи, него одређени *ECU* уређаји.

| Тип сензора | Број сензора | Количина података по сензору |
|---|---|---|
| *Радар* | 4-6 | 0.1-15 мегабита по секунди |
| *Лидар* | 1-5 | 20-100 мегабита по секунди |
| *Камера* | 6-12 | 500-3500 мегабита по секунди |
| *Ултразвучни* | 8-16 | <0.01 мегабит по секунди |
| *Позиционирање, покрети* | - | <0.01 мегабит по секунди |
| ***Аутономно возило*** | - | 3-40 гигабита по секунди |

*Табела 2.3 Процена количине генерисаних података из аутономног возила [35]*

### 2.2.3. Магистрале у возилу

Магистрале служе као комуникациони систем за пренос података између компоненти или компоненте и улазно-излазног уређаја у возилу (добављање података са сензора). Произвођачи аутомобила су утрошили време, енергију и новац како би развили разне серијске магистрале. Обзиром да не постоји универзална магистрала (као што је *USB* на рачунарима опште намене), у возилу се користи више магистрала сходно потребама и из историјских разлога. Слика 2.5 приказује један пример магистрала. Избор магистрале коју ће нека компонента користити диктирају захтеви као што су цена, кашњење, брзина, детерминизам (гаранција достављања поруке), механизам приоритета итд.




*Слика 2.5 Пример магистрала и компоненти у возилу*

У типичном возилу примена серијске комуникације се односи на три велике апликативне области: размена података између компоненти, дијагностика и тестирање, и спуштање садржаја на трајну меморију (енгл. *flash memory*). Због специфичних захтева, поред *UART*, *I2C* и *SPI* опште познатих магистрала, у возилу се користе и мање познате магистрале као што су:

- *CAN/CAN-FD* (енгл. *Controller Area Network / Flexible Data rate*) [1] - Омогућава прикључивање већег броја *ECU*-ова на магистралу. Омогућава средње брзине преноса података као што је нпр. у дијагностици и приликом спуштања садржаја на меморију.
- *VAN* (енгл. *Vehicle Area Network*) [36] - Серијска комуникација мале брзине (125*Kbit/s*).
- *MOST* (енгл. *Media Oriented Systems Transport*) [12] - Протокол високе брзине и специјалних комуникационих механизама је уведен како би олакшао интеграцију инфо-забавних система у возило. Зове се још и мултимедијална спрега јер се користи за пренос аудио и видео сигнала као и звука и података.
- *LIN* (енгл. *Local Interconnect Network*) [3] - Серијски протокол за потребе ниског протока (до 20*Kbit/s*) који омогућава јефтину интеграцију сензора и

актуатора у возило. Примењује се код ретровизора, подизача прозора, климе, седишта, брисача, светла, контролне табле итд.

- *K-Line* [37] - представља један од првих начина приступа *ECU*-у у комплетном возилу. Због тога је прописана у *OBD* стандарду и ту налази највећу примену. Данас, *CAN* магистрала је полагано замењује.
- *IEBus* (енгл. *Inter Equipment Bus*) [38] - Комуникациона магистрала ‘између опреме унутар возила или шасије. Углавном се користи за аудио и навигациону опрему и представља де факто стандард у Јапану.
- *FlexRay* [2] - Општенаменски протокол велике брзине (до 10*Mbit/s*) са безбедносним карактеристикама. Омогућава детерминизам и дизајниран је да буде редундантан. Користи се за безбедносно-критичне потребе: вожња, скретање.
- *D2B* (енгл. *Domestic Digital Bus*) [39] – Фибер оптичка технологија заснована на прстену са опсегом до 12*Mbit/s,* коју је Конзорцијум оптичких чипова (енгл. *Optical Chip Consortium*) одредио за коришћење у апликацијама у возилу.

Најчешћа магистрала која се користи у сврху размене порука између компоненти је *CAN* која подржава различите брзине преноса података. Пренос података са сензора се углавном врши преко *LIN* магистрале. У пракси ове две магистрале су и најзаступљеније у возилу.

У последње време због нових захтева укључујући и потребу за преносом веће количине података све више се разматра етернет технологија:

- Етернет - Представља поуздану дуговечну технологију која је уопштено направљена, оптимизације нису намењене за аутомобилски индустрију и присутан је недостатак детерминизма. Добра примена ове магистрале је код сервера за складиштење података при вожњи, а лоша код управљања самом вожњом.
- Детерминистички етернет (енгл. *deterministic ethernet*) - Технологија која је ближа захтевима аутомобилске индустрије, омогућава ниска кашњења и треперења, гаранцију квалитета сервиса и има уграђену отпорност на грешке.

- *BroadR-Reach* [40] - Алтернатива за протокол високог протока. Омогућава да више система у возилу истовремено приступа информацијама, што омогућава уградњу већег броја *ECU*-ова и уређаја, као што су напредни системи за помоћ возачу, инфо-забавни системи итд.

За потребе преноса веће количине података у реалном времену, као што је пренос слика са више камера до једне или више компоненти код напредних система за помоћ возачу *PCIe* магистрала је магистрала која се све више помиње у аутомобилској индустрији.

Сем жичаних, присутне су и бежичне аутомобилске комуникације. Постоје многи сценарији комуникације возила са другим возилом, са облаком, са инфраструктуром и са мобилним уређајима (енгл. *vehicle-to-everything*) [41]. Примена бежичних комуникација у возилу се може наћи и код инфо-забавних система (музика, игрице и сл.), док изван возила говоримо о контролним/статусним информацијама (нпр. клизав пут, време, стање на путевима, навигација (конвој, визија унапред), безбедност (најава инцидента), интернет (мултимедија, забава, информације)).

### 2.2.4. Преглед типова софтверских платформи

Аутомобилска индустрија тежи ка ограничавању типова софтверских платформи које се користе у спрези са хардвером. Као последица интеграције *ECU*-ова, очекује се нормализација одређеног броја софтверских платформи које ће омогућити раздвајање функција возила од хардвера *ECU*-а. Хардвер и наменски софтвер (укључујући оперативни систем) ће зависити од кључних нефункционалних захтева. У циљу омогућавања поменутог раздвајања и сервисно-оријентисане архитектуре, следеће четири врсте софтверских платформи се виде као доминантни типови платформи за возила будућности:

- *Платформа заснована на времену* - Подразумева контролер директно повезан са сензором или актуатором у системима који морају да подрже строге временске захтеве у реалном времену - мало кашњење. Расподела ресурса је заснована на времену. Платформа се односи на системе који треба да достигну највише *ASIL* нивое. Пример платформе је класични АУТОСАР.

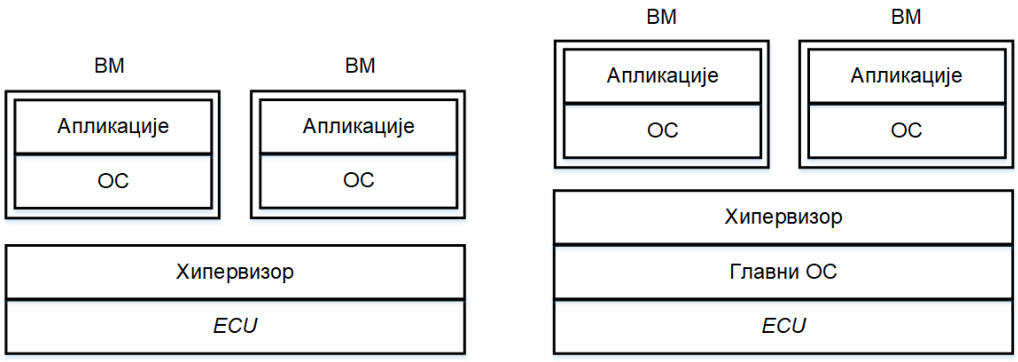
- *Платформа заснована на времену и на догађајима* - Хибридна платформа која комбинује безбедносне апликације високих перформанси (нпр. подржавајући *ADAS* и високо аутоматизовану вожњу). Оперативни систем раздваја апликације и периферне уређаје (сензоре, актуаторе), док се апликације распоређују на временској бази. Радно окружење осигурава да се безбедносно критичне апликације извршавају у изолованим контејнерима са јасним одвајањем од других апликација у возилу. Пример платформе је адаптивни АУТОСАР.
- *Платформа заснована на догађајима* - Односи се на системе који нису критични по питању безбедности (нпр. системи за информисање и забаву). Апликације су јасно одвојене од периферних уређаја, а ресурси се распоређују користећи распоређивање засновано на догађајима или најбоље доступно (енгл. *best effort)* распоређивање. Платформа садржи видљиве и високо кориштене функције које омогућавају интеракцију корисника са возилом. Примери платформе су *Android*, *AGL*, *GENIVI* и *QNX*.
- *Платформа заснована на облаку (ван возила)* - Завршна платформа покрива и координише спољни приступ подацима и функцијама у возилу. Платформа је одговорна за комуникацију, као и безбедносне и сигурносне провере апликације и успоставља дефинисану спрегу возила, укључујући удаљену дијагностику.

Добављачи у аутомобилској индустрији и технолошки играчи су већ почели да се специјализују за неке од ових платформи. На пример, у системима за информисање и забаву (платформа заснована на догађајима) компаније развијају комуникационе могућности као што су *3D* и проширена (енгл. *augmented*) навигација. Други пример је вештачка интелигенција и сензори за апликације високих перформанси, где се добављачи удружују са *OEM*-овима како би развили рачунарску платформу.

### 2.2.5. Хипервизори

Виртуализација је технологија која подржава стварање софтверске реализације рачунара (виртуалне машине), који се мапира на ресурсе доступне на постојећем рачунару, познатом као хост или главни рачунар. Свака виртуална машина може приступити делу ресурса главне машине (као што су процесорско време, *I/O* ресурси и сл.). За креирање виртуалних машина користе се хипервизори или контролори виртуалне машине (енгл. *Virtual Machine Monitor*). Хипервизори се деле у два главна типа (Слика 2.6):

- *Тип 1 или нативни хипервизор* - не захтева оперативни систем на главном рачунару. Извршава се директно на хардверу, пружајући све могућности које су потребне гостујућим оперативним системима,
- *Тип 2 или хостовани хипервизор* - представља врсту софтвера која се извршава над оперативним системом и може да користи све примитиве које оперативни систем нуди немајући директан приступ хардверу. Сходно томе, све хардверске операције које гостујући ОС захтева од виртуалне машине, морају да се преведу у операције приступа главног ОС-а хардверу, што доводи до губитка перформанси. Примери ових хипервизора су *VMware Workstation* [42], *VMware Player* [43] и *VirtualBox* [44].




*а)* *б)*

*Слика 2.6 Типови хипервизора: а) тип 1: нативни б) тип 2: хостовани*

За потребе наменских система хипервизори типа 1 су пожељнији јер имају минимални утицај на перформансе. У наставку је дат преглед овог типа хипервизора по два критеријума [13]: дизајн хипервизора и тип виртуализације.
Према дизајну хипервизори се могу поделити на:

- *потпуно монолитне хипервизоре* - један софтверски слој одговоран за приступ хардверу главног рачунара-а, *CPU* виртуализацију и *I/O* емулацију госта ( *Xvisor* [45] и *VMware ESXi Server* [46]),
- *делимично монолитне хипервизоре* - обично представљају проширење оперативног система опште намене. Подржавају приступ хардверу главног рачунара и *CPU* виртуализацију у ОС кернелу, а *I/O* емулацију госта из корисничког простора софтвера (*Linux KVM* [47] и *VMware Workstation* [42]),
- *хипервизоре са микро кернелима (енгл. micro-kernelized)* - лагани микрокернели који обезбеђују хардверски приступ и виртуализацију процесора у микрокернелу хипервизора. Зависе од руковаоца виртуалним машинама за пуни приступ хардверу, *I/O* емулацији госта и осталим услугама. Неки покрећу сваки руковалац (енгл. *driver*) главног рачунара као посебну виртуалну машину руковаоца, и немају заједничко управљање ( *Xen* [48], *Microsoft Hyper-V* [49], *OKL4 Microvisor* [50] и *INTEGRITY Multivisor* [51] ).

Према типу виртуализације, постоје два типа реализационо, који користе различите приступе за додељивање ресурса главне машине виртуалним машинама:

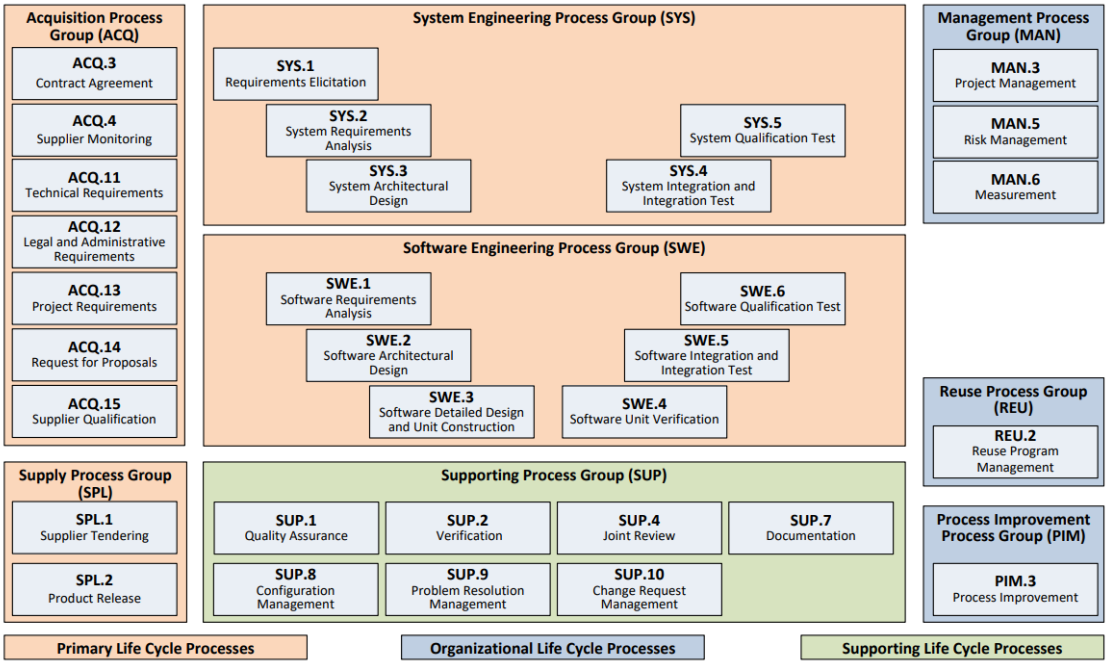
- *пуна виртуализација* - хипервизор директно користи хардверске ресурсе главне машине (процесор, меморију) и служи као виртуална хардверска платформа за гостујуће ОС-ове. Држи виртуалне машине независне једне од других, и свака виртуална машина извршава свој ОС. Обзиром да ови хипервизори емулирају виртуални хардвер, и имају потребе за својом обрадом, главна машина мора резервисати одређену процесну моћ и ресурсе да извршава хипервизорску апликацију. Ово може утицати на свеукупне перформансе и успорити апликације (*ARM-VE* [52], *Intel VT-x* [53] и *VT-d* [54]),
- *паравиртуализација* - гостујући ОС-ови су свесни постојања једни других. Хипервизор и даље одржава виртуални хардвер за сваку виртуалну машину, на којима се извршавају модификовани гостујући ОС-ови (свака инструкција која приступа дељеним ресурсима главне машине се мења са хиперпозивом, тј. позивом ка хипервизору) (*Xen* [48], *z/VM* [55], и *VLX* [56]).

Без обзира на технику виртуализације хипервизори такође омогућавају механизме за размену података између виртуалних машина.

Увођење хипервизора типа 1 у дизајн система у аутомобилској индустрији доноси неколико ствари: поновно коришћење софтвера за нове пројекте, осигуравање сигурности и безбедности (кроз изолацију различитих домена), омогућавање интеграције софтвера отвореног кода (*Android*/*Linux*) и АУТОСАР софтверских компоненти. Такође, овај тип хипервизора омогућава и брзо и сигурно покретање система (нпр. преко РТОС-а како би се омогућио рани приступ *CAN* магистрали). Коришћење хипервизора такође доприноси мањој потреби за простором, тежином, и напајањем (нпр. дигитални кластер и инфо-забавни систем на једној хардверској платформи).

### 2.2.6. Нефункционални захтеви

Поред стандардних функционалних захтева у аутомобилској индустрији акценат се ставља и на нефункционалне захтеве. Да би се задовољили високи стандарди квалитета развоја аутомобилског софтвера, процесни модел *Automotive SPICE* [57] (у даљем тексту *ASPICE*) је успостављен у целој индустрији и чини основу за сигурност и безбедност.




*Слика 2.7 ASPICE 3.1 процесне групе ([57])*

*ASPICE* стандард је настао као иницијатива немачких произвођача аутомобила. Стандард пружа смернице за побољшање процеса развоја софтвера и процене добављача. Изведен је из општег *SPICE* стандарда (*ISO/IEC 15504* [58]) који представља међународни ИСО стандард за спровођење процена пословних процеса, са посебним нагласком на процес развоја софтвера. Састоји се од више процесних група које су приказане на Слици 2.7.

Све процесне групе се могу разврстати у три врсте процеса развојног циклуса: главни процеси, организациони процеси и процеси подршке. Главни процеси развојног циклуса *ASPICE* стандарда се заснивају на *V* (енгл. *Verification and Validation*) развојном моделу – за сваки процес, од захтева до изворног кода, постоји одговарајући тест. Сам *ASPICE* стандард је општи и не специфицира одређене алате и методологије за његову реализацију током развојног процеса. Општи кораци су представљени следећом секвенцом процеса:

- Добављање захтева од наручиоца, јасно дефинисање и пречишћавање истих (енгл. *requirements elicitations*).
- Мапирање захтева наручиоца у системске захтеве (енгл. *system requirements analysis*). У овом кораку врши се реструктурирање захтева како би се могли тестирати, и такође дефинисање додатних системских захтева који потичу од других заинтересованих страна (нпр. других добављача)
- Системски архитекта рашчлањује захтеве на логичке целине (енгл. *system architectural design*). При томе доноси одлуке о дизајну, попут тога шта се ће се радити у хардверу, шта у софтверу, где ће се покретати и како ће комуницирати.
- За сваку софтверску целину, изводе се софтверски захтеви из системских захтева (енгл. *software requirements analysis*). Ове две врсте захтева се обично разликују према формулацији – нешто у вези са аутомобилом је системски захтев, док се софтверски захтеви често односе на улазне и излазне сигнале дефинисаних целина.
- Софтверски архитекта рашчлањује софтверске захтеве софтверских целина у софтверске јединице (енгл. *software architectural design*). Овде је фокус на одређеном уређају при чему се води рачуна о расположивим ресурсима (меморија, процесорско време итд.).

- Дизајнирање и реализација сваке софтверске јединице (енгл. *software detailed design and unit construction*). Дизајнирање апстрактних модела (машина стања) и реализација у облику изворног кода.

Са десне стране *V* модела се налазе одговарајући тестни процеси. Процеси од доле ка горе су:

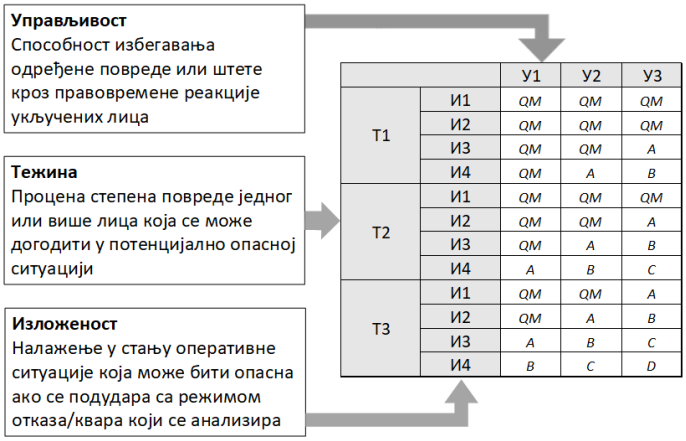
- Тестови софтверских јединица који потврђују дизајн (енгл. *unit tests*). Тестирају да ли изворни код одговара дизајну, да ли су не-функционални захтеви (као што су код се не сме срушити током извршавања) испуњени.
- Интеграциони тестови за софтверску архитектуру (енгл. *software integration and integration test*). Дају одговор на питање да ли композиција софтверских јединица и даље ради у саставу софтверске целине.
- Тестови за квалификацију софтвера који потврђују софтверске захтеве (енгл. *software qualification test*). Одговарају на питање да ли је софтверска целина у складу са софтверским захтевима.
- Системски интеграциони тестови за системску архитектуру (енгл. *system integration and integration test*). Представља састављање свих логичких целина (софтверских, хардверских) у један систем и проверу да ли исправно функционишу заједно.
- Тестови за квалификацију система који потврђују системске захтеве (енгл. *system qualification test*). Одговарају на питање да ли је цели систем у складу са системским захтевима.
- Тестови прихватљивости које извршава наручилац, како би потврдио да је развијено то што је тражено (енгл. *acceptance test*).

Један од појмова који се често помиње заједно са стандардом је двосмерна повезаност (енгл. *bidirectional tracebility*). Односи се на постојање веза између резултата рада (енгл. *work products*) чиме се подржавају процеси покривености, анализе утицаја, праћења статуса реализације захтева. Додатно, двосмерна повезаност се експлицитно дефинише између:

- тестних случајева и резултата тестова,
- захтева за промену и резултата рада на које ови захтеви за промену утичу.

**Безбедносни аспекти**

Функционална безбедност је изазов који се адресира још у процесу дизајна система. *OEM*-ови када развијају спецификације за *ECU* или за платформу која ће консолидовати више *ECU*-ова, такође специфицирају тражени безбедносни ниво за дате функције и тиме утичу на дизајн платформе или система. Неки *ECU*-ови имају безбедносне захтеве из којих се рађа потреба за ригорознијим процесом развоја софтвера како би се испоштовао ИСО26262 стандард [59]. Овај стандард је предвиђен за примену у безбедносним системима који укључују један или више Е/Е система и који су уграђени у серијска путничка возила са максималном бруто масом возила до 3500кг. Дефинише *ASIL* шему класификације ризика, уводећи четири нивоа од *A* до *D*. *ASIL D* диктира највише захтеве за интегритет производа, а *ASIL A* најниже. Функције које су идентификоване као *QM* не диктирају никакве безбедносне захтеве.




*Слика 2.8 Одређивање ASIL нивоа*

Одређивање *ASIL* нивоа је резултат анализе вероватноће могућих непожељних догађаја и процене параметара ризика (тежина (енгл. *severity*), изложеност (енгл. *exposure*) и управљивост (енгл. *controllability*)). Слика 2.8 приказује *ASIL* нивое у односу на ова три параметра. На пример, систем управљања (енгл. *Steering Control System*), представља висок ризик у случају квара у ситуацијама када је возило у покрету. Ако се догоди такав квар, постоји велики потенцијал да нанесе штету људима (укључујући возача, путнике, пешаке и друге у оближњим возилима). Због тога је класификован високим безбедносно-

критичним нивоом *ASIL D*. Насупрот њему, квар на радио уређају неће нанети штету возачу, па се класификује као *ASIL A* или *QM*.

Стандард ИСО26262 дефинише могући начин реализације функционалних безбедносних аспеката у развоју система и на нивоу процеса и на нивоу метода. За софтверске архитектуре, функционална безбедност је кључни фактор. Основни механизми интегритета као што су интегритет праћења система, партиционисање или праћење времена и процеса су доступни и већ се користе у серијским пројектима.

Стандард у развој софтвера уноси додатне безбедносне захтеве и препоручује методе за сваки од корака у развоју софтвера. Ниво препоручености методе зависи од самог *ASIL* нивоа. Пример метода које се препоручују у разним фазама се налази у табели испод.

| **Развојна фаза** | **Пример препоручене методе** |
|---|---|
| *Планирање* | • Коришћење утврђених принципа дизајна софтвера<br>• Коришћење утврђених алата, конвенција именовања |
| *Дизајн архитектуре софтвера* | • Нотација дизајна (полу-формална нотација, …)<br>• Принципи дизајна (ограничена употреба прекида, хијерархијска структура софтверских компоненти)<br>• Механизми откривања грешака (провера веродостојности, провера граница улазних и излазних података, спољна компонента за надзор)<br>• Механизми руковања грешкама (независна паралелна обрада, …)<br>• Методе верификације (анализа протока података, симулација динамичког дела дизајна, …) |
| *Дизајн софтверских јединица и реализација* | • Нотација дизајна (полу-формална нотација, …)<br>• Принципи дизајна (иницијализација променљивих, једна улазна и једна излазна тачка из функције, …) – већински покривени за *C* програмски језик са *MISRA-C* [81]<br>• Методе верификације дизајна (статичка анализа кода, анализа протока података, …) |
| *Тестирање софтверских јединица* | • Методе тестирања (тестирање спрега, тестирање убацивањем намерних грешака, …)<br>• Методе извођења тестних случајева (анализа захтева, анализа граничних вредности, …)<br>• Метрике покривености тестирања (покривеност грана, …) |

| Развојна фаза | Пример препоручене методе |
|---|---|
| *Интеграција и тестирање софтвера* | • Методе тестирања (тестирање спрега, тестирање убацивањем намерних грешака, …)<br>• Методе извођења тестних случајева (анализа захтева, анализа граничних вредности, …)<br>• Метрике покривености тестирања (покривеност функција, покривеност позива) |
| *Верификација безбедносних захтева* | • Тестна окружења (*HIL, ECU* мрежно окружење, возило) |

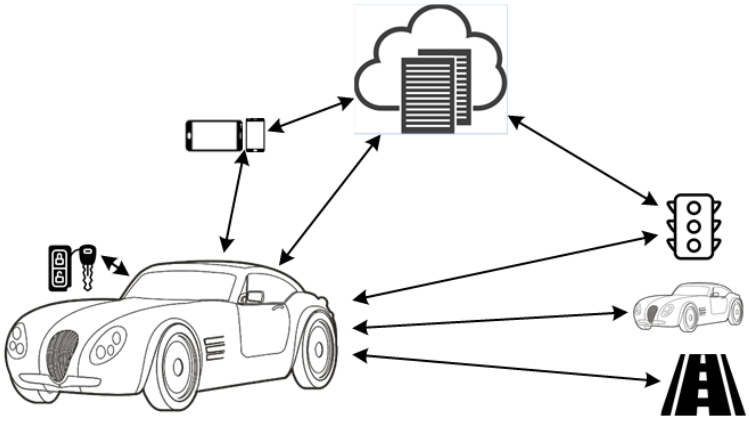
*Табела 2.4 Примери препоручених метода у односу на ASIL ниво*

Стандард не обухвата и не узима у обзир неке технологије које су се почеле у новије време користити у развоју аутономних возила (нпр. методе дубоког учења). Концепт рада са неуралним мрежама је другачији него што је то случај са традиционалним аутомобилским компонентама. Сам стандард није примењив на такве и сличне проблеме. Као 14. део овог стандарда је био предвиђен *SOTIF* (енгл. *Safety of the Intended Functionality*) - безбедност предвиђене функционалности. Због сложености проблематике, издвојен је у посебан стандард ИСО 21448 [61]. ИСО 21448 стандард обухвата аспекте ограничене могућности предвиђене фукционалности и злоупотребе система. Под предвиђеном фунционалношћу се подразумева употреба система у складу са упутствима произвођача. Први аспект обухвата фунционалне недостатке који доводе до ризика од опасности који могу бити унутрашње природе (ограничен домет детекције сензора, недовољна свест о ситуацији (нпр. на кривини систем детектује статични објекат испред возила који се налази ван пута и покреће кочење у случају опасности)) и спољне природе (временски услови који утичу на домет сензора, сметње између сензора на другим возилима). Други аспект предвиђа да се током анализе система узме у обзир очекивана злоупотреба од стране корисника и покуша да се елимише или ублажи. Пример злоупотребе је омогућавање функције помоћи на аутопуту током градске вожње. Стандард пропагира усклађеност са тренутном ситуацијом пре доношења одлука, ради безбеднијег функционисања возила.

**Сигурносни аспекти**

Сигурносни механизми су релевантни у развоју аутомобила већ дуже време. Системи као што су сигурни електронски кључеви или сигурно чување пређене километраже су често већ стандардне функције. Међутим, због све веће повезаности возила (кроз апликације као што су удаљена дијагностика, удаљени приступ, глобално позиционирање и сл.), индустрија се суочава са новим изазовима по питању сигурности и приватности. Према основном правилу информационе технологије, "*оно што је повезано биће нападнуто*", системски аспекти сигурности и приватности имају све већи значај и у аутомобилској индустрији.

Сигурносни захтеви код повезаног возила су присутни у комуникацији у самом возилу, као и у комуникацији возила са окружењем (друга возила, друмска инфраструктура, мобилни уређаји и друге апликације које захтевају приступ интернету).



*Слика 2.9 Сигурносни екоситем возила*

Слика 2.9 приказује сигурносни екосистем возила у коме се могу уочити три велике групе: системи у возилу, *V2X* комуникација и комуникација која захтева приступ интернету. Главни аспект сигурности система у возилу се тиче заштите хардвера и софтвера који се извршава на *ECU*-у. Потребно је обезбедити сигурно покретање (енгл. *secure boot*), извршавање и складиштење апликација. Разни хардверски сигурносни модули се предлажу као решење. Један од изазова представља и ажурирање софтвера преко мреже где је потребно верификовати аутентичност и интегритет софтверског ажурирања.

Мреже у возилима као што су *CAN* и *Flexray* првобитно нису дизајниране тако да задовољавају сигурносне захтеве [62], [63] [64]. Разне мере се предлажу да се то надомести [65] [66] [67]. АУТОСАР стандард дефинише *SecOC* модул (енгл. *Secure On Board Communication*) који врши аутентикацију порука и верификацију ажурности порука.

Један од сигурносних механизама у системима возила представља и одвајање домена. Са све већом умреженошћу возила, уређаји мрежног пролаза би требало да реализују мрежне баријере који ће дозвољавати само ауторизованим *ECU*-овима да пошаљу поруке безбедносно критичним деловима система и да регулишу размену информација између функционалних домена у возилу. Такође је потребно обезбедити извршавање више апликација на истом *ECU*-у тако да оне не утичу једна на другу, тј. да се извршавају у изолованом окружењу.

Бежичне сензорске мреже, као и уређаји повезани са возилом представљају потенцијалну тачку сигурносног напада. Бежичне сензорске мреже се користе на пример код удаљеног откључавања возила. Примери уређаја повезаних са возилом су *OBDII* и паметни уређаји (мобилни телефони, таблети), који имају дефинисане своје комуникационе интерфејсе као што су *USB*, Блутут и мобилна мрежа.

Умрежена возила комуницирају са другим возилима и са инфраструктуром, где се мрежа и окружење стално мењају. Домени попут телематике и инфо-забавних система имају апликације које захтевају приступ интернету. Све ово отвара многобројна питања и изазове везане за сигурносне ризике и механизме, од којих су неки већ решени у другим областима и решења се могу пресликати, док неки тек треба да се реше у аутомобилској индустрији. Разне организације су почеле да се баве сигурносним аспектима и предлажу механизме како би се сигурност повећала, међутим јединствени стандард још увек не постоји.

*IPA* (енгл. *Information Technology Promotion Agency*) је у Јапану објавила водич за заштиту информација возила 2013 [68] . Овај водич предлаже стратегије за разне сигурносне претње.

*SAE International* је објавио приручник за развој сигурних система почетком 2016. године [69] под називом „Водич за кибернетичку сигурност“. Овај приручник описује процесе и методе и следи ИСО26262 у погледу животног циклуса. Документ није стандард. Међутим, он сумира основне напоре као што су

истраживачки програми или постојећи стандарди и публикације. Као такав, он је вредан допринос и може послужити као полазна тачка за увођење сигурносних процеса и метода.

Сврха стандарда за кибернетичку сигурност ИСО/САЕ 21434 [70] је да осигура да произвођачи аутомобила, као и сви добављачи имају структуиране процесе који ће подржавати сигурносне аспекте у развоју аутомобила. Односи се на друмска возила укључујући све компоненте, везе и софтвер. Такође, развој сигурног хардвера не треба потценити. Слично као и ИСО 26262 и овај стандард се односи на цели развојни процес и развојни циклус возила. Прати *V* развојни модел, и наглашава како се у свакој фази сигурносни аспекти морају узети у обзир. Одређивање нивоа сигурносног ризика возила и његових компоненти је једна од главних активности која ће бити дефинисана у стандарду - *TARA* (енгл. *Threat Analysis and Risk Assesment*).

Из свега наведеног види се да се у аутомобилским системима мора правити разлика између разних домена у возилу (инфо-забавни, системи за помоћ возачу, телематски системи итд.) и на основу тога применити одговарајуће технике поузданости, безбедност и сигурности, како за софтверску инфраструктуру тако и за развојни процес.

**Архитектурни аспекти**

Квалитет архитектуре софтвера се може проценити кроз својства, недостатке и квалитете архитектурног стила. Као резултат процене архитектуре очекује се став о повољним околностима за дату архитектуру у односу на захтеве. Поред функционалних захтева које архитектура треба да испуни, ту су и други аспекти архитектуре средњег слоја, као што су проширивост, скалабилност, конфигурабилност, портабилност, модуларност, целовитост. Ови аспекти иако не утичу директно на саму функционалност имају важну улогу у развојном процесу и утичу на ефикасност развоја софтвера.

Постоје разни модели квалитета који се односе на структурирање не-функционалних захтева: *McCall* [71], *Boehm* [72], *Dromey* [73], *FURPS* [74], *IEEE* [75], *ISO/IEC 9126* [76], *ISO/IEC 25010* [77]. Сваки од њих анализира и има различите особине и факторе квалитета. Неке особине се провлаче кроз већину

модела квалитета. Разни аутори [78] [79] анализирају и пореде наведене моделе. Закључак је да се ИСО модел чини најприкладнијим за процену квалитета развоја софтвера. На слици испод се виде пример *ISO/IEC 25010* модела (особине и подособине којима се баве ).


*Слика 2.10 ISO/IEC 25010 модел квалитета ([80])*

## 2.3. ПРЕГЛЕД РЕШЕЊА СРЕДЊЕГ СЛОЈА

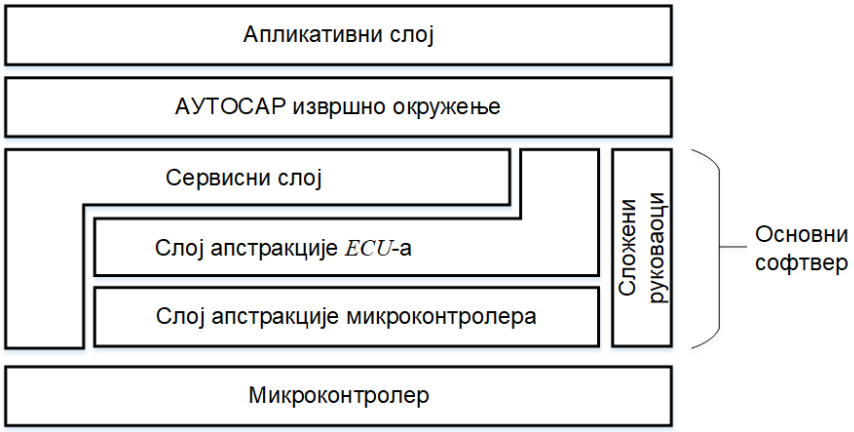
Постоји неколико категорија у које би се могла сврстати тренутна решења средњег слоја: стандарди и иницијативе, комерцијална решења, решења отвореног кода и решења у академији.

### 2.3.1. Стандарди и иницијативе

АУТОСАР (енгл. *Automotive Open System Architecture*) [81] је стандард који развија конзорцијум који се састоји од *OEM*-ова и добављача. Циљ конзорцијума је да дефинише заједничку софтверску архитектуру за наменски софтвер у возилима. Овај стандард дозвољава дефинисање софтверске компоненте - апликације независно од *ECU*-а, како би се између осталог повећала поновна искоришћеност софтверских компоненти. АУТОСАР представља референтну софтверску архитектуру са следећим особинама:

- стандардизација софтверске архитектуре *ECU*-а, са циљем компатибилности компоненти и једноставнијом интеграцијом у систем,
- односи се на више организација које учествују у аутомобилском тржишту (*OEM*-ови, добављачи, произвођачи алата, и нови учесници на тржишту). Глобални је стандард са преко 200 партнера до Септембра 2020.

Представља класичну софтверску референтну архитектуру која је дефинисана када су технологија, софтвер и алгоритми потребни за софтверску архитектуру аутомобилских апликација већ тестирани у пракси.




*Слика 2.11 Софтверски слојеви AUTOSAR платформе*

Слика 2.11 приказује слојевиту структуру AUTOSAR софтверске платформе. Уочавају се три главна слоја:

- *Апликативни слој* - састоји се од софтверских компоненти које су мапиране на *ECU*. AUTOSAR софтверске компоненте представљају атомске софтверске компоненте (апликативног типа или типа сензор/актуатор). Све интеракције између AUTOSAR софтверских компоненти се усмеравају кроз окружење извршавања. AUTOSAR спрега обезбеђује повезивање софтверских елемената који окружују извршно окружење.
- *Окружење извршавања* (енгл. *Runtime Environment* , у даљем тексту *RTE*) - обезбеђује апстракцију комуникације пружајући исте спреге и сервисе без обзира да ли се користе комуникациони канали између *ECU*-ова (*CAN*, *LIN*, *FlexRay* и *MOST*) или на самом *ECU*-у.
- *Основни софтвер* (енгл. *Basic Software*) - стандардизовани софтверски слој који пружа сервисе AUTOSAR софтверским компонентама и неопходан је за покретање функционалног дела софтвера. Не испуњава ниједан функционални посао и налази се испод *RTE*-а. На пример, одговоран је за руковање комуникацијом између различитих *ECU*-ова на магистралама и за време дијагностичких сервиса који се читају када се возило одведе у радионицу. На самом дну овог слоја се налазе слојеви везани за *ECU*

хардверске ресурсе чиме АУТОСАР нуди механизме за независност хардвера и софтвера.

АУТОСАР стандард (класични) се односи на статичку, унапред конфигурисану и самим тим ограничену платформу. Стандард се не односи на системе за информисање и забаву и друге нове технолошке захтеве као што су *ADAS*, *V2X*, екстерна комуникација.

Како би испунио нове технолошке захтеве исти конзорцијум дефинише нову платформу - Адаптивни АУТОСАР [82], чији је циљ да пружи окружење са већом процесном моћи и већим протоком података, динамички развој нових функционалности, интеракцију са не-АУТОСАР апликацијама, и ажурирање софтвера преко мреже. Адаптивна платформа се независно развија са циљем да обе платформе коегзистирају на истој мрежи, без угрожавања стабилности класичне АУТОСАР архитектуре која је доказана у пракси.

Карактеристике нове платформе су дефинисане како би опслужиле нове захтеве и укључују сервисно-оријентисани приступ архитектури, заснован на *SOME/IP* комуникационом протоколу, подржавање динамичког развоја нових функционалности и гарантују интеракцију са не-АУТОСАР апликацијама. Ово је подржано оперативним системом који подржава динамичко покретање и распоређивање апликација, динамичко заузимање ресурса и одржавање компатибилности између различитих оперативних система преко *POSIX* апликативне спреге. Адаптивна АУТОСАР платформа намерава да допуни специфичне функционалности возила и одржава основне карактеристике аутомобилског домена као што су: поузданост, доступност, одрживост и безбедност.

*JasPar* (енгл. *Japan Automotive Software Platform and Architecture*) [83] је индустријско партнерство са циљем промовисања софтверске технологије у возилима и смањења трошкова развоја подстичући јапанске компаније да заједнички развијају неконкурентне технологије.

### 2.3.2. Комерцијална решења

У понуди постоје разна комерцијална решења средњег слоја софтвера за рачунарски систем у возилима. Ова решења су углавном развијана за одређену намену, тј. за одређени домен и представљају развојне платформе. Поред извршивог средњег слоја, произвођачи углавном нуде и алате за рад са њима. Већина доступних решења бави се напредним системима за помоћ возачу или инфо-забавним системима. Неки произвођачи као што су *Mentor, QNX* и *Electrobit* нуде више од једне платформе, тј. по једну платформу за сваки домен.

Платформе за напредне системе помоћи возачу засноване на АУТОСАР стандарду углавном долазе са помоћним алатима за генерисање и коришћење. Ове платформе ако су засноване на класичном АУТОСАР-у испуњавају *ASIL* захтеве и то углавном највишег нивоа. У наставку је наведено неколико платформи.

*QNX* платформа за *ADAS* (енгл. *QNX Platform for ADAS*) [84] се састоји од *QNX* безбедносног оперативног система и средњег слоја који се односи на сензоре, обраде и приступ хардверским ресурсима, као и на мрежну комуникацију. *Volcano VSTAR*, преименован у *Capital VSTAR* [85] представља развојну платформу која подржава АУТОСАР 4. *EB Tresos* платформа [86] се састоји од *Autocore OS*-а и средњег слоја *Tresos Autocore*. *Vector* нуди *MICROSAR* [87] платформу и алате, која представља пуну реализацију АУТОСАР стека и испуњава захтеве за свим *ASIL* нивоима. *MotionWise* [88] је платформа средњег слоја која подржава већи број оперативних система поред АУТОСАР стандарда, као што су *Linux, QNX* и *VxWorks*. *EB Corbos* [89] представља *ASIL* сертификовану платформу која је компатибилна са адаптивним АУТОСАР-ом и поред средњег слоја (*Corbos AdaptiveCore*) и оперативног система (*Corbos Linux*) садржи и хипервизорски слој (*EB Corbos Hypervisor*).

Присутне су и платформе чији су фокус инфо-забавни системи. *QNX* платформа возила (енгл. *QNX Car Platform*) [90] се састоји од *QNX Neutrino* оперативног система у реалном времену, платформског дела, средњег слоја и слоја за интеракцију са човеком. Средњи слој се фокусира на мултимедију, навигацију, радио и блутут (енгл. *bluetooth*). *Automotive OPTstack Middleware* [91] је платформа намењена инфо-забавним системима, извршава се на *Linux* оперативном систему.

Састоји се од аудио, видео и графичких библиотека и компатибилна је са *GENIVI* платформом отвореног кода.

*EB Connected Vehicle* [92] платформа се фокусира на умреженост, сигурност, ажурирање преко мреже и удаљену дијагностику. *Connext Drive* [93] пружа средњи слој за умрежавање за апликације како унутар возила тако и за сценарије умрежених аутономних возила.

*Integrity* платформа [94] је обухватнија платформа од горе наведених и намењена је како инфо-забавним, тако и системима за помоћ возачу. Подржава одвајање функција преко хипервизорске технологије.

Присутне су и разне софтверске платформе које су специфичне за одређену хардверску платформу, прилагођене су и искоришћавају њен максимум по питању перформанси и омогућавају приступе разним хардверским блоковима. Да би се овакве платформе користиле потребно је знање о самој хардверској платформи. *NVIDIA DRIVE™ OS* [95] представља референтни оперативни систем са софтверском платформом дизајнираном за развој апликација аутономне вожње на хардверу заснованом на *DRIVE AGX*. Основни софтверски пакет обухвата хипервизор типа 1, *NVIDIA® CUDA®* библиотеке, *NVIDIA TensorRT™*, *NvMedia*, и друге компоненте које су оптимизоване да пруже директан приступ *DRIVE AGX* хардверским акцелераторима. *Processor SDK-Vision* [96] је вишејезгарна развојна платформа за фамилију *TDAx* чипова. Софтверски пакет омогућава корисницима креирање различитих *ADAS* апликација кроз прављење тока података који укључује прихват видеа са камера, претпроцесирање истог, обраду разним алгоритмима и приказ резултата. Ова софтверска платформа садржи основне руковаоце, као и кернеле за рачунање и обраду слике за *EVE* и *DSP* наменска језгра. *Qualcomm* је такође представио своју платформу - *Snapdragon Ride Autonomous Stack* [97].

Оно што се може приметити као заједничко код већине платформи је структура високог нивоа, тј. присуство хипервизорског слоја, слоја оперативног система и средњег слоја. Све ове платформе углавном су специјализоване за један домен и не могу се лако применити у другом домену. Тренутно не постоји платформа која ће бити примењива у сваком сценарију возила нове генерације. У зависности од критеријума поређења и самих захтева за шта је нека платформа потребна, платформе се могу поредити по томе да ли подржавају AUTOSAR

(класични и адаптивни), које хипервизоре, оперативне системе и хардверске платформе подржавају, да ли задовољавају неки од *ASIL* нивоа и да ли обезбеђују извршавање без сметњи у временском и просторном смислу. Платформе се разликују и по додатним алатима који олакшавају рад са њима и подржавају нпр. симулацију и/или извршавање на платформи.

Средњи слојеви ових платформи омогућавају разне функционалности, од аудио, видео, графичких функционалности код платформи оријентисаних на инфо-забавне системе до функционалности везаних за добијање података са сензора, њихову обраду, могућност приступа разним хардверским блоковима (кодовање, акцелерација и сл.).

### 2.3.3. Отворена решења

Иако је тржиште аутомобилске индустрије доста затворено тржиште у односу на потрошачку електронику, постоје промене које се већ догађају у развоју софтвера у инфо-забавним системима, где произвођач и добављачи усвајају софтвер отвореног кода. Линукс платформа се већ неко време користи у овим системима, иако то није видљиво на први поглед. Произвођачи аутомобила подржавају велики број Линукс дистрибуција, и полако теже ка преласку на *Automotive Grade Linux* [98], у даљем тексту АГЛ. АГЛ представља пројекат отвореног кода који прави отворени оперативни систем заснован на Линуксу, као и спреге за апликације у возилима. Прва спецификација се фокусира на софтверску платформу за инфо-забавне системе.

Такође, у последње време Андроид платформа има велики удео у системима за информисање и забаву. Коришћење Андроид платформе за инфо-забавне системе пружа све функције које нуде врхунски уграђени инфо-забавни системи, са додатком информативног садржаја који помаже возачу, мултимедијалног центра и навигационог система (*Google Maps*). У наставку је дат преглед неколико постојећих решења отворених платформи са нагласком на примену и функционалности које подржавају.

Некадашња *GENIVI* (енгл. *Geneva In-Vehicle Infotainment*) референтна платформа, сада под именом *COVESA* [99] се фокусира на инфо-забавне апликације и сценарије умрежених возила. Састоји се од Линукс централних сервиса, средњег слоја и спреге апликативног слоја.

*Open Robinos* [100] представља отворену референтну платформу за аутономну вожњу која дефинише архитектуру, софтверске компоненте, спреге, контролне и комуникационе механизме, узимајући у обзир и безбедносне аспекте. Платформа циља разне хардверске платформе са различитим оперативним системима, као што су *QNX* или *Unix* оперативни системи.

*ROS2* (енгл. *Robot Operating System*) [101] средњи слој (скуп алата, библиотека и конвенција) потиче из окружења роботских платформи са циљем олакшавања стварања сложеног и робустног понашања на овим платформама. Такође се све више помиње у контексту аутомобилске индустрије, и служи као база за прављење софтверских платформи углавном за напредне системе помоћи возачу. Једна од платформи заснованих на РОС средњем слоју је *AutoWare* [102]. Овај софтверски пројекат отвореног кода пружа скуп модула за самостално управљање возилом у условима урбане вожње.

Присутне су и разне отворене платформе као резултати истраживачких пројеката који имају за циљ развој референтне спецификације и реализацију исте. Један од таквих пројеката је и *OVERSEE* (енгл. *Open Vehicular Secure Platform*) [103] пројекат који се фокусира на сигурносне проблеме које доносе отворене платформе и сигурносну интеграцију широког спектра комуникационих канала.

### 2.3.4. Решења у академији

Број решења у академији приметно је мањи у односу на комерцијална решења. Такође, решења у научним радовима се углавном односе на специфичне корисничке случајеве или на специфичан сегмент. Мали је број решења који се односе на централне интеграционе платформе као и на њихове средње слојеве софтвера. Већина решења се налази у раном развоју, тј. представљају концептуална решења и/или решења која се потврђују симулацијом на персоналним рачунарима, или у комбинацији персоналних рачунара и наменске платформе.

У пар радова су представљене централне платформе дефинишући и хардверске елементе [104], [105], [106], [107], [108]. Аутори се углавном концептуално осврћу на саму платформу, њене могућности и анализу како задовољава потребе централне интеграционе платформе. Као и код комерцијалних решења и овде је видљив сличан шаблон, а то је безбедносни микроконтролер са

хетерогеним обрадним јединицама високих перформанси. Средњи слој софтвера и апликативне спреге нису детаљно приказане, углавном се спомиње коришћење постојећих технологија: *ROS* у [104], Аутосар дизајн методологија и *SOME/IP* у [105], Аутосар (класични и адаптивни) у [106].

*RACE* пројекат [108] представља модуларну архитектуру за будућа електрична возила. Хардверска архитектура се састоји од централне платформе са вишеструким обрадним јединицама које су повезане на сензоре и актуаторе преко етернет везе. Апликативни софтвер је раздвојен од хардвера користећи *RACE RTE* [109] који је заснован на *Chromosome* средњем слоју [110]. *RACE RTE* пружа сервисе за распоређивање и за пренос података. Сервиси се фокусирају на податке и засновани су на механизму објаве и претплате.

*Chaaban* у својој дисертацији [111] представља дистрибуирану архитектуру за напредна возила, под називом *SCOOT-R*. Софтверско решење представља скуп основних сервиса, у склопу средњег слоја софтвера, изнад кернела у реалном времену. Омогућава дистрибуцију компонената на вишепроцесорске јединице, заједно са комуникационим и синхронизационим сервисима. Заснива се на клијент/сервер архитектури. Има за циљ да смањи трошкове и време развоја дистрибуираних апликација у реалном времену. Фокусира се на технике распоређивања. Хардверске платформе над којима врши симулацију су умрежени микроконтролери и персонални рачунари.

*UNICARagil* пројекат [112] нуди модуларну архитектуру за агилна, аутоматизована концептуална возила. У оквиру пројекта развијена је сервисно оријентисана софтверска архитектура [113] која је примењива на централне платформе у возилима (микроконтролер и *POSIX* јединице за обраду). Комуникациони механизми су засновани искључиво на етернет вези (укључујући и актуаторе). За комуникацију се користи наменски развијен софтвер заснован на РТПС протоколу. Рад приказује симулацију са персоналним рачунарима као POSIX јединицама за обраду не користећи хетерогену наменску платформу.

*Fusion* платформа [114] се декларише као софтверска платформа која подржава развој система за аутономну вожњу, адресирајући комплексност и перформансе. Ова софтверска платформа се може инстанцирати на више различитих хардверских платформи. Омогућава независност апликативног

софтвера у смислу оперативног система и у смислу преноса података. Користи механизам микрокернела. Поред апликација, пружа и неке сервисе са унапред дефинисаним спрегама: логовање, руковање извршавањем, складиштење података. Развијен је иницијални концепт, аутори не дају информације о конкретним хардверским платформама над којима је *Fusion* платформа реализована.

Одређени број радова бави се архитектурама заснованим на контејнерима и на микросервисима за аутономна возила [115], Конкретне платформе нису наведене. Аутори истичу да у будућности планирају да адресирају извршавање у реалном времену, укључујући и проток података.

Број радова који се односе на средње слојеве за специфичан домен или оперативни систем је далеко већи. На пример, у инфо-забавном домену постоје средњи слојеви за развој апликација засновани на Андроид оперативном систему [116]. У [117] је представљен флексибилни и модуларни средњи слој који омогућава лакши пренос мултимедијалних апликација са персоналног рачунара или из области потрошачке електронике у возило. Безбедносни аспекти платформи се разматрају у [118] - достизање детерминизма у Адаптивној Аутосар платформи и [119] - предлог платформе за надгледање безбедносно критичних апликација у централном рачунару возила. *Iorio* [120] предлаже унапређење сигурносног аспекта за средње слојеве који користе *SOME/IP,* док се аутори [121] баве средњим слојем који ће бити независтан од хардвера а односи се на домен вештачке интелигенције.

## 2.4. ПОСТАВКА ЦИЉЕВА ИСТРАЖИВАЊА

Циљ истраживања је предложити архитектуру која систематском анализом доноси организацију и раслојавање новог софтверског стека за примене у извршавању апликација за нова возила у складу са описаним проблемима. Такође, архитектура треба да уважи активности у индустрији и академији (компатибилност са постојећим широко примењеним стандардима).

Динамичка софтверска платформа се сусреће са изазовима, индустрија активно ради на томе, али су и даље нерешени неки од општих проблема. **Основни циљ истраживања** је предложити архитектуру средњег слоја софтвера у хетерогеном окружењу у возилу који ће омогућити бржи и једноставнији развој и интеграцију апликација у једном оваквом окружењу где је присутна транзиција ка

централној интеграционој платформи, тј. потребно је првенствено консолидовати софтвер, како би се омогућио прелазак на јединствену платформу. Средњи слој представља корак напред у овој транзицији.

Да би се омогућио једноставнији развој одређене проблеми/изазови се требају адресирати. Они уједно представљају подциљеве овог истраживања. Табела 2.5 приказује изазове/проблеме, тренутни начин суочавања са изазовима и потенцијална решења/циљеве.

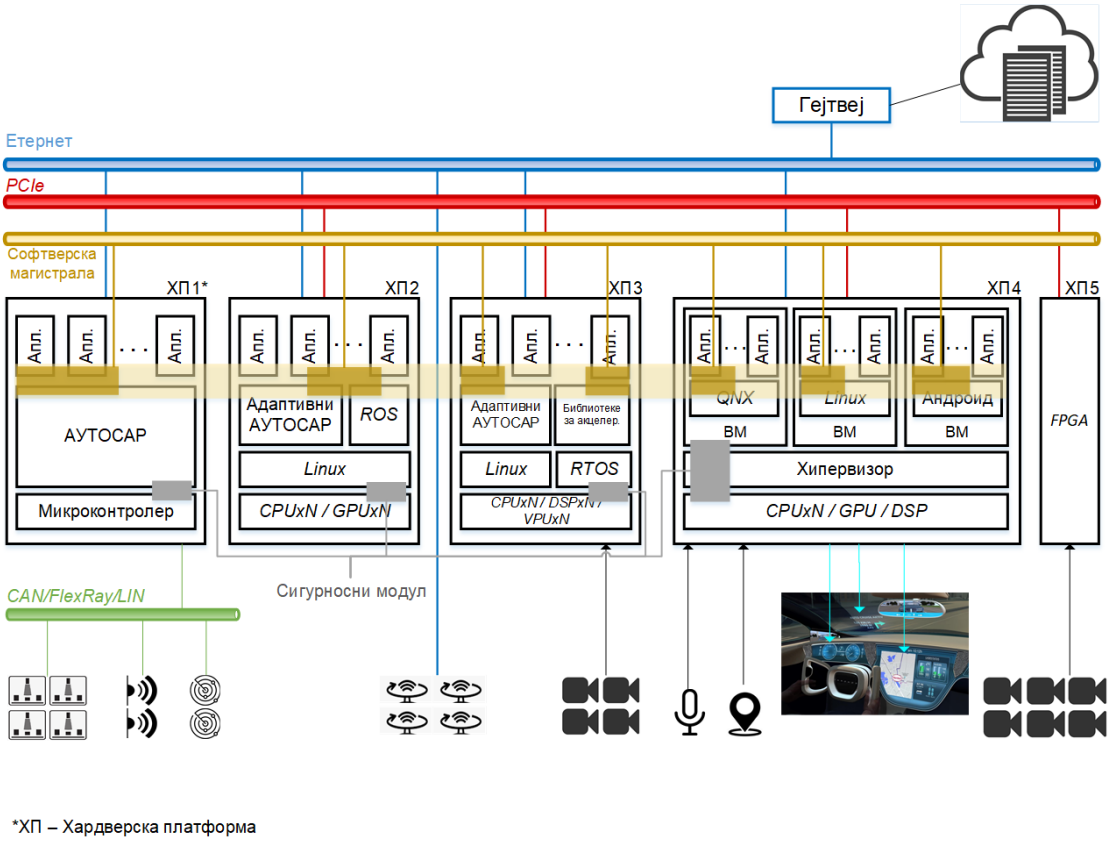
| Изазов | Тренутно стање/Циљ |
|---|---|
| *Једноставније коришћење хетерогене природе чипова* | *Тренутно стање:* Произвођачи чипова пружају спреге специфичне за чипове за приступ језгрима и наменским ресурсима чипа. Са друге стране, постоје спреге које се односе на један специфичан аспект, а доступне су на више врста чипова (нпр. *OpenGLes*).<br>*Циљ*: Апликација треба да буде у стању да униформно користи разне процесорске јединице на једном или више истих и/или разних чипова |
| *Већа портабилност апликација* | *Тренутно стање*: Делови апликација (у мањем или већем облику) су и даље везани за спреге специфичне за одређену хардверску или софтверску платформу (нпр. добављање слика са камера).<br>*Циљ*: Апликација треба да буде потпуно портабилна, тј. да средњи слој уведе потпуно раздвајање апликативног софтвера од хардвера. |
| *Интероперабилност (лакше повезивање и размена информација)* | *Тренутно стање:* Систем се састоји од разних компоненти које користе разне сервисе, који чето долазе од стране независних тимова, користећи различите технологије и алате. Повезивање свих компоненти у систем представља изазов.<br>*Циљ:* Средњи слој треба да постави темељ за међусобну сарадњу, тако да компоненте могу без проблема да комуницирају. |
| *Проширивост и конфигурабилност* | *Тренутно стање:* Системи су проширриви и делимично конфигурабилни, али не на једноставан начин. Убацивање нових функција захтева додатне ресурсе и траје знатно времена.<br>*Циљ:* Проширива и конфигурабилна софтверска платформа (новим функцијама, апликацијама, као и хардверским блоковима). |
| *Коришћење стандардизованих библиотека и спрега* | *Тренутно стање:* Стандардизоване библиотеке и спреге су широко распрострањене, наглашавају специфичне аспекте, али саме по себи нису довољне за развој АДАС апликације/система.<br>*Циљ:* Средњи слој треба да пружи апликацијама могућност коришћења стандардизованих библиотека и спрега отвореног кода, као и могућност повезивања са технологијама потрошачке индустрије. |

| Изазов | Тренутно стање/Циљ |
|---|---|
| *Рани развој и интеграција апликација на циљној платформи* | *Тренутно стање:* Уобичајени животни циклус развоја АДАС апликација је развој алгоритма на персоналном рачунару, затим пренос истог на циљну платформу, при чему се тек на крају увиде евентуални недостаци и/или некомпатибилности са циљном платформом.<br>*Циљ:* Омогућити рани развој апликација или делова апликација на једној и/или више циљних платформи. |

*Табела 2.5 Изазови централних платформи, тренутно стање и циљеви*

# 3. ПРЕДЛОГ АРХИТЕКТУРЕ СРЕДЊЕГ СЛОЈА СОФТВЕРА

Први корак при преласку на централну платформу је раздвајање софтвера од хардвера. То се може постићи коришћењем средњег слоја који апстрахује хардверске могућности и пружа их доступним преко функција и сервиса користећи стандардизовану програмску спрегу. Ово поглавље даје приказ и анализу окружења и захтева за средњи слој, као и предлог архитектуре средњег слоја софтвера за централне хардверске архитектуре у односу на дато окружење и захтеве. На Слици 3.1 је приказан концепт софтверске платформе у окружењу где постоји више доменских хардверских платформи.




*Слика 3.1 Динамичка софтверска платформа на примеру више доменских хардверских платформи*

Динамичка платформа користи софтверску магистралу и пружа апликацијама могућност коришћења платформе и њених сервиса на униформан начин (нпр. добављање информација са сензора без обзира на коју хардверску платформу је сензор прикључен). Разни аутори анализирају изазове и захтеве које софтверска платформа треба да испуни. На основу њихових резултата и анализом датом у Поглављу 2. идентификоване су главне области које је потребно адресирати при дефинисању архитектуре средњег слоја софтвера:

- Апстраховање хетерогености
    - хардверске хетерогене платформе – комплексност из више аспеката, различити ресурси, руковање ресурсима, програмирање хетерогене платформе, апстракција хетерогености платформе [122], [123], [114], [108], [124], [125] и [126].
    - разни сензори и актуатори - које је потребно апстраховати тако да апликације могу да приступе истима, без обзира где се извршавају [122].
    - разне магистрале у возилу – апликације треба да користе сервисе без обзира где се тај сервис извршава и без знања којим магистралама су повезани, док магистрале треба да омогуће довољан пропусни опсег за потребе апликација [122], [123], [108], [125].
- Руковање животним циклусом апликација и интеграција апликација у реалном времену
    - Рад у реалном времену – апликације треба да имају могућност извршавања на циљној платформи у реалном времену [114], [124], [119], [127].
    - Постојећа решења средњег слоја је потребно уважити – компатибилност са стандардима, могућност коришћења широко прихваћених библиотека и сл.,
- Помоћне функције за развој и тестирање
    - Функције логовања података, статистички подаци током извршавања, повратне информације из система на радном месту где се развија апликација [122],

- Нефункционални аспекти
    - Безбедносни аспекти – захтеви које диктира ИСО26262 стандард и анализа безбедносних аспеката технологија које стандарди још не обухватају (нпр. методе дубоког учења) [114], [122], [108], [124], [125].
    - Сигурносни аспекти – у области повезаности, ажурирања софтвера преко мреже, дијагностике итд., [114], [108], [124], [125].
    - Архитектурни аспекти - релевантни аспекти архитектуре средњег слоја софтвера као што су проширивост, скалабилност [123], [122], [127], конфигурабилност, портабилност [122], [108], [128], [127], сложеност одржавања софтвера [127], целовитост, тестабилност [122].

## 3.1. АНАЛИЗА ЗАХТЕВА

Потребно је дизајнирати средњи слој софтвера за централне хардверске архитектуре које захтевају динамичку софтверску платформу. Предложено решење треба да обрати пажњу на изазове једне такве централне рачунарске платформе у возилима. Неки од кључних идентификованих проблема који воде до споријег развоја апликација у аутомобилској индустрији су недостатак могућности поновног искоришћења софтвера као и разноликост платформи. Средњи интеграциони слој треба да омогући и олакша развој апликација (првенствено АДАС апликација, али не нужно само апликација из АДАС домена), тако што ће апстраховати хетерогеност и тиме омогућити веће поновно искоришћење софтвера.

Кључни изазов у једном оваквом хетерогеном окружењу је успостављање интероперабилности (тј. способности различитих делова система да се лако повежу и да размењују међусобно информације). Систем се састоји од раздвојених делова, тј. компоненти које користе разноврсне сервисе, често развијене од стране независних тимова, користећи различите технологије и алате. Средњи слој треба да постави заједнички темељ за међусобну сарадњу између ових хетерогених компоненти тако да могу без проблема да комуницирају међусобно.

Захтеви се генерално могу поделити на функционалне и нефункционалне захтеве. У наставку је дат преглед листе основних захтева. Захтеви су груписани по логичким групама, и на крају је дат приказ нефункционалних захтева.

**Апстраховање хетерогености платформе** представља један од основних захтева за средњи слој, тј. пружање подршке за извршавање апликације на једној или више различитих хардверских платформи (узимајући у обзир сензоре, актуаторе, магистрале).

*Захтев 1.* Апстракција хетерогености платформе – униформна спрега. Средњи слој треба да обезбеди апстракције саме хетерогене платформе. Апликација не треба да води рачуна о детаљима приступа и реализацији функционалности које се односе на саме хетерогене платформе (нпр. комуникациона магистрала између два чипа, начин преноса података између две процесорске јединице). Средњи слој се конфигурише на нивоу система како би имао у виду специфичности самог система (нпр. чип А и чип Б су повезани *PCIe* магистралом).

*Захтев 2.* Подршка за различите типове процесорских јединица. Потребно је обезбедити механизам који ће омогућити да се делови апликације или апликација извршавају на одређеној процесорској јединици и пружити подршку за више различитих процесорских јединица (централна процесорска јединица, процесорска јединица за обраду слике, графичка процесорска јединица итд.)

*Захтев 3.* Подршка за вишејезгарне процесорске јединице. Потребно је обезбедити механизам који ће омогућити да се делови апликације или апликација извршавају на вишејезгарним процесорским јединицама (нпр. двојезгарна ARM централна процесорска јединица), са могућношћу избора језгра, тамо где је то могуће.

*Захтев 4.* Подршка за више врста чипова. Потребно је омогућити извршавање средњег слоја на више врста чипова. Реализацијом на више чипова се показује општост и независност од самог чипа. Тиме се такође омогућава бржа процена и евалуирање чипова у циљу нпр. избора чипа на коме апликација има боље перформансе.

*Захтев 5.* Подршка за више чипова. Потребно је омогућити извршавање апликације над једним или више чипова, коришћењем средњег слоја (нпр. један чип некада није довољан за обезбеђивање задовољавајућих перформанси, па се за делове обраде користе и процесорске јединице са другог чипа).

*Захтев 6.* Подршка за више врста сензора и актуатора. Сензори и актуатори су повезани на платформу на разне начине (користе се разне спреге, често и руковаоци који су специфични за самог произвођача). У оквиру средњег слоја, потребно је раздвојити апликацију од одређеног сензора, пружити апстракцију сензора и актуатора, тј. јединствену спрегу за приступ.

*Захтев 7.* Подршка за више врста магистрала. Магистрале се вишеструко користе у хетерогеним системима, за комуникацију свих нивоа софтвера (од руковаоца до апликација) са другим системима, сензорима, итд. Потребно је обезбедити могућност подршке за више врста магистрала, које у овом случају користи и сам средњи слој, а апликације имају могућност коришћења истог.

*Захтев 8.* Подршка за комуникацију између процесорских јединица. Хетерогени чипови пружају више од једне врсте процесорских јединица. Апликације које се развијају, да би постигле одговарајуће перформансе, користе често поред централне процесорске јединице и друге доступне процесорске јединице. Средњи слој треба да омогући и пружи апстракције за комуникацију између разних процесорских јединица на чипу.

*Захтев 9.* Подршка за комуникацију између чипова. У хетерогеном систему, понекад се налази више чипова и за реализацију неке апликације потребно је користити оба чипа (нпр. на једном чипу су повезане камере, а на другом треба да се врши обрада). Средњи слој треба да омогући комуникационе канале који ће омогућити комуникацију између чипова, тако да апликација не мора да води рачуна о доступним магистралама и самом начину преноса података између та два чипа.

*Захтев 10.* Подршка за пренос већих количина података. Неке од постојећих библиотека за комуникацију се профилишу за пренос одређене количине података, и тврде оптималне перформансе над мањом количином података

(нпр. *vsomeip* [129] је спрега за контролне податке, док при преносу већих количина података значајније утиче на перформансе система). Узевши у обзир већи број камера у систему, потребно је обезбедити и подршку за дистрибуирање тих података.

*Захтев 11.* Подршка за извршавање на различитим оперативним системима. Због сложености циљних хетерогених платформи, коришћење проверених оперативних система је неизбежно. Са становишта обраде у реалном времену, пожељан је оперативни систем за рад у реалном времену. Међутим, за одређене платформе, произвођачи подржавају ‘општије’ оперативне системе (нпр. *Linux* оперативни систем на *NVIDIA* хардверу – за коришћење графичких могућности захтева се *CUDA* библиотека која није доступна за оперативне системе за рад у реалном времену). Средњи слој треба да подржи извршавање на обе врсте оперативних система.

*Захтев 12.* Подршка за апстракцију руковања меморијом. Организација и врста меморије се може разликовати од једног до другог чипа. За приступ меморији некад се користе библиотеке које су широко распрострањене, а некада су у питању библиотеке које су специфичне за самог произвођача чипа. Апликација не треба да зна детаље приступа меморији. Средњи слој треба да омогући апстракције за руковање меморијом.

**Руковање апликацијама** треба да буде омогућено преко средњег слоја, као и интеграција апликација у систем, притом одржавајући могућност извршења у реалном времену.

*Захтев 13.* Подршка за извршавање апликација у реалном времену. АДАС апликације треба да обрађују податке и информишу систем о резултатима истог у реалном времену (нпр. детектован објекат близак аутомобилу приликом паркирања). Иако извршавање апликације у реалном времену у великој мери зависи од саме природе и реализације апликације и слојева софтвера на које се ослања апликација, предложени средњи слој не треба да онемогући извршавање апликације у реалном времену.

*Захтев 14.* Подршка за руковање апликацијама. Средњи слој треба да омогући подршку за руковање животним циклусом апликације, подршку за

параметризовање апликације при покретању, као и током извршавања апликације.

*Захтев 15.* Подршка за спрегнутост са другим доменима. Модерна окружења у возилу су таква да границе између домена нису јасно разграничене и модерне апликације све више се простиру и ван свог домена. Средњи слој треба да пружи могућност спреге апликација са инфо-забавним и/или другим доменима у возилу. На пример, преглед задње камере и додатних информација на екрану инфо-забавног система.

*Захтев 16.* Подршка за коришћење постојећих хетерогених библиотека и спрега. Средњи слој треба да подржи тј. очува могућност коришћења широко распрострањених и прихваћених библиотека и спрега (нпр. апликација треба да има могућност коришћења *OpenGL* спреге).

**Додатне функционалности за развој и тестирање апликација** потребно је да обезбедити преко средњег слоја. Ове додатне функције се не тичу директно функције коју апликација реализује, него представљају помоћне функционалности средњег слоја, који олакшавају и убрзавају сам развој и тестирање.

*Захтев 17.* Подршка за извршавање на персоналном рачунару у сврху развоја апликација. Процес развоја апликација на персоналном рачунару је бржи у односу на развој на наменским платформама. АДАС алгоритми или њихови делови се иницијално развијају и тестирају на персоналним рачунарима где се и потврђује њихова исправност, а накнадно се преносе на наменске платформе где се додатно оптимизују. Средњи слој треба да подржи извршавање апликације (целе или делимично) и на персоналном рачунару као једној од платформи.

*Захтев 18.* Подршка за разне симулаторе. Средњи слој треба да подржи могућност симулирања одређених података (нпр. брзина возила, слика са камере) током верификације и развоја апликације.

*Захтев 19.* Подршка за добављање статистичких информација. Средњи слој треба да подржи могућност добављања статистичких информација о самом извршавању апликације (нпр. број слика добијених са камере и просечно време обраде једне слике).

*Захтев 20.* Подршка за добављање додатних корисничких информација. Средњи слој треба да подржи могућност дефинисања и добављања додатних релевантних информација о самом извршавању апликације (нпр. проценат времена у одређеном периоду када је детектовано да је возач поспан).

**Нефункционални захтеви** се не односе на саме функције које средњи слој треба да подржи, него утичу на квалитет софтвера у смислу проширивости, портабилности, поновне искористивости и сл.

*Захтев 21.* Подршка за проширење средњег слоја. Средњи слој треба да омогући механизам за проширење новим функционалностима (нови алгоритам, нови сензор, нови симулатор, нова платформа, ново процесорско језгро), као и да тај механизам буде једноставан за коришћење.

*Захтев 22.* Средњи слој треба да буде модуларан. Модуларност тј. декомпозиција софтвера на мање делове са стандардизованим спрегама смањује комплексност система и доприноси поједностављивању тестирања и интеракције.

*Захтев 23.* Средњи слој треба да буде одговарајућег степена сложености ради лакшег одржавања софтвера током времена. Степен сложености одржавања софтвера представља степен ефикасности и ефектности до кога се може мењати софтвер како би се побољшао, поправио и прилагодио променама у захтевима и/или окружењу.

*Захтев 24.* Средњи слој треба да буде портабилан. Портабилност подразумева способност адаптације софтвера за извршавање на различитим платформама.

*Захтев 25.* Средњи слој треба да буде тестабилан. Обзиром да је безбедност рада од велике важности у аутомобилској индустрији, тестирање је битан захтев. Компоненте треба да се могу тестирати изоловано једна од друге као и док међусобно комуницирају.

*Захтев 26.* Средњи слој треба да буде поновно искористив (енгл. *reusabilty*). Потребне је да буде дизајниран тако да омогућује поновно коришћење компоненти у неком облику у оквиру процеса развоја производа.

*Захтев 27.* Средњи слој треба да буде компатибилан са стандардима, тј. не треба да буде дизајниран тако да не може да буде компатибилан са постојећим широко распрострањеним стандардима (нпр. адаптивни Аутосар).

*Захтев 28.* Средњи слој треба да буде компатибилан са развојним праксама у аутомобилској индустрији. Добре развојне праксе које су резултат вишегодишњег искуства из аутомобилске индустрије треба да се прате (нпр. ХИС метрике).

*Захтев 29.* Средњи слој треба да буде обухваћен концептом интеграције са безбедносним системима на нивоу система као и било који други софтвер. Безбедност представља кључни захтев у аутомобилској индустрији. Безбедносно-критичне функције треба да раде у случајевима отказа појединих компоненти. Да би се ово успоставило, одређене мере треба да се донесу кроз цели процес развоја, рада и одржавања функција. Различити делови система треба да буду у складу са различитим нивоима безбедности (*ASIL* нивои).

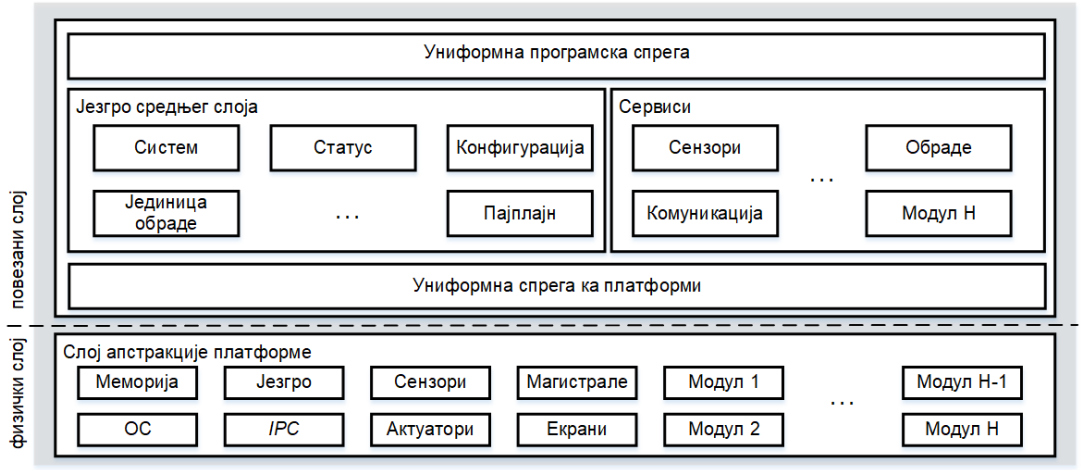
*Захтев 30.* Средњи слој треба да буде обухваћен концептом интеграције са сигурносним системима на нивоу система као и било који други софтвер. У оквиру возила, претпоставља се да сервиси могу веровати једни другима, док иста претпоставка не важи на отвореном – у складу са најновијим трендовима повезаних возила, сервиса, инфраструктуре.

## 3.2. ПРЕДЛОГ АРХИТЕКТУРЕ

Софтверска платформа има за циљ да пружи интеграциони (средњи слој) слој који омогућава и олакшава развој апликација у аутомобилској индустрији (првенствено АДАС апликација). Концепт архитектуре средњег слоја који се извршава на свакој од хардверских платформи са Слика 3.1 је приказан на Слика 3.2.

Архитектура средњег слоја се састоји од два дела: доњег који представља слој апстракције платформе (где се под платформом не подразумева само хардверска платформа него и софтверски делови који су карактеристични за саму платформу: приступ ресурсима, оперативни систем, хипервизор, дељена меморија, језгра,

платформски развојни алат итд.) и горњи део који је заједнички за све платформе и који реализује саму логику средњег слоја, његове сервисе и пружа униформну програмску спрегу ка апликацијама. Поредећи са архитектуром двојне структуре (енгл. *dual structure architecture*), горњи део (језгро, сервиси и спрега) одговара повезаном слоју (енгл. *connected layer*) који је сличан ако не и исти за све платформе, док слој апстракције платформе одговара физичком слоју и карактеристичан је за платформу.



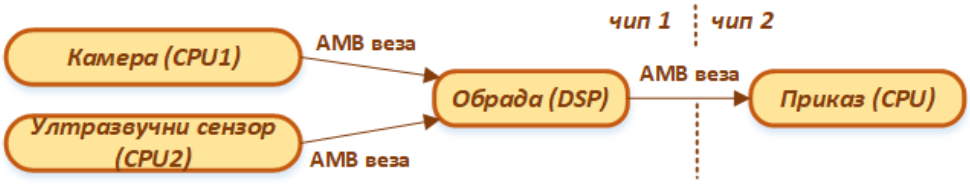

*Слика 3.2 Концепт архитектуре средњег слоја динамичке софтверске платформе*

У наставку је дат преглед термина који ће се користити у овом раду:

- AMB средњи слој – представља назив средњег слоја софтвера
- Апликација – представља функцију у возилу, нпр. праћење будности возача, преглед окружења возила, и користи спреге средњег слоја софтвера
- AMB пајплајн – механизам којим се развија апликација. Представља граф задатака, где сваки задатак прима податке, обрађује податке и проследи даље податке наредним задацима. AMB пајплајн представља ток података и операције обраде над тим подацима.
- AMB јединица обраде (енгл. *processing unit*) је појединачна фаза у пајплајну. Извршава се на одређеном језгру.
- AMB веза међу јединицама обраде (енгл. *inter processing unit connection*) – представља везу између AMB јединица обраде које се извршавају на истом или различитом језгру истог или различитог чипа.

AMB средњи слој је предвиђен да се првенствено користи кроз две групе корисничких случајева: прављење апликације дефинисањем AMB пајплајна, тј. дефинисањем обрадних јединица и веза међу њима, и проширење сервиса, тј. дефинисање и портовање разних обрада користећи AMB спреге, независно од чипа на коме се извршава.

Апликација се развија на принципу AMB пајплајна. При дизајнирању пајплајна изабран је приступ заснован на графовима због саме природе алгоритама за обраду слика – засновани су на подацима. AMB пајплајн се састоји од AMB јединица обраде као главних блокова обраде и веза међу њима. Пример једноставног AMB пајплајна је приказан на Слика 3.3. Пајплајн се састоји од четири јединице обраде, три на првом чипу и једна на другом чипу. У загради је приказан тип језгра на ком се извршава јединица обраде.



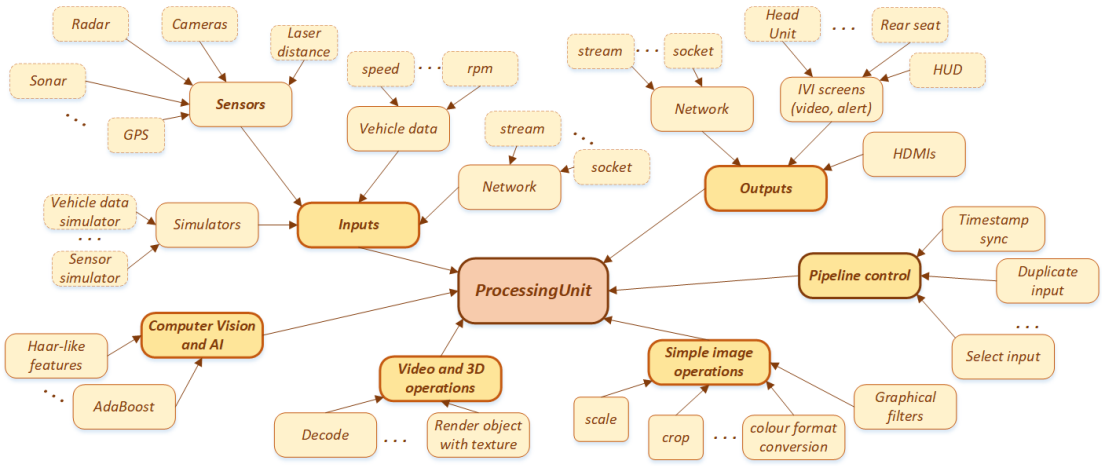

*Слика 3.3 Пример једноставног AMB пајплајна*

AMB пајплајн се конструише коришћењем AMB униформне програмске спреге како би се дефинисале AMB јединице обраде и размена података између њих. Језгро средњег слоја реализује основну логику елемената пајплајна као и везу између јединица обраде, као и системске функције, док сервиси представљају реализацију конкретних јединица обраде. Основни скуп јединица обраде, тј. сервиса долази у склопу AMB средњег слоја, а корисници имају могућност да реализују своје јединице обраде које ће касније користити у склопу пајплајна. Слика 3.4 приказује основне веће логичке групе сервиса/јединица обраде:

- Улазне јединице обраде (енгл. *Input PUs*)– налазе се на почетку пајплајна. Представљају скуп улаза који су захтевани у апликацији. Ова група укључује сензоре, разне податке из возила, податке добављене преко етернет мреже, податке из симулатора, итд.
- Излазне јединице обраде (енгл. *Output PUs*) – налазе се на крају пајплајна. Представљају начин како да се прикаже резултат апликације. Укључују

ХДМИ излазе, објављивање података преко етернет мреже, инфо-забавне екране.

- Јединице обраде за контролу пајплајна (енгл. *Pipeline control PUs*) – представљају усмеривач тока пајплајна за избор и комбиновање улаза и излаза. На пример, одабир једног од више улаза.
- Јединице обраде слике (енгл. *Simple image PUs*) – операције над индивидуалним пикселима или низом пиксела слике, као што су промена формата боје, исецање, промена величине и сл.
- Јединице обраде видеа и 3Д графике (енгл. *Video and 3D graphics PUs*) – коришћење видео и аудио декодера или 3Д графичких библиотека за обраду. Примери су декодовање, исцртавање објекта са одређеном текстуром итд.
- Рачунарска визија и вештачка интелигенција (енгл. *Computer vision and AI (Artificial Intelligence) PUs*) – операције из области рачунарске визије, машинског учења и друге операције вештачке интелигенције.




*Слика 3.4 Пример поделе група сервиса – јединице обраде*

АДАС апликације се обично развијају на персоналним рачунарима и након тога се пребацују на циљне платформе. AMB средњи слој пружа разне апстракције улаза и излаза кроз улазне и излазне јединице обраде. Са већ готовим блоковима за улазе и излазе на циљној платформи, фокус при развоју апликације се може пребацити на оптимизацију алгоритма и само мапирање на платформу, уместо на адаптацију на ново окружење циљне платформе.

Апстракције које су битне за развој апликације су везане за интеграцију на платформу, нпр. сензори (камере, лидари, радари, ултразвучни сензори) и подаци из возила. AMB такође подржава апстракције догађаја из других апликација, као и симулацију сензора и података из возила у случају да развојна платформа не пружа још увек подршку за неки од потребних делова за алгоритам. Као део излазних апстракција, подржани су разни екрани (ХДМИ излаз на екран, излаз на инфо-забавне јединице). Апстракције излаза на инфо-забавну јединицу у возилу подразумевају приказ видеа на централном екрану, на екрану на задњем седишту и/или разна визуална и звучна обавештења.

Хетерогена архитектура платформе је сакривена од апликација. Средњи слој пружа униформну програмску спрегу за јединице обраде које се извршавају на различитим језгрима. Свака од AMB јединица обраде се извршава на језгру које је дефинисано у моменту креирања јединице обраде. За неке од јединица обраде могућ је избор језгра на ком ће се извршавати, док су неке јединице обраде додељене само одређеном језгру. Језгра на којима је подржана јединица обраде се дефинишу одвојено за сваку јединицу обраде, у зависности од реализације јединице обраде. Реализација извршавања јединице обраде на одређеном језгру је сакривена од апликације.

Поред дефинисања језгра на ком се извршава јединица обраде, потребно је дефинисати и како су јединице обраде повезане (излаз једне јединице обраде је улаз у другу јединицу обраде). Ове везе се дефинишу на униформан начин за све јединице обраде. Реализација AMB везе води рачуна о правилној конфигурацији веза између језгара (нпр. дељена меморија у оквиру истог чипа, *PCIe* и/или етернет магистрала између чипова). Без обзира на језгро за које је јединица обраде намењена, исте спреге се користе ка претходној јединици обраде (добављање улазних података) и ка наредној јединици обраде (достављање резултата).

AMB повезани слој дефинише и спрегу ка платформи на униформан начин, у сврху брже и једноставније преносивости и реализације на више платформи.

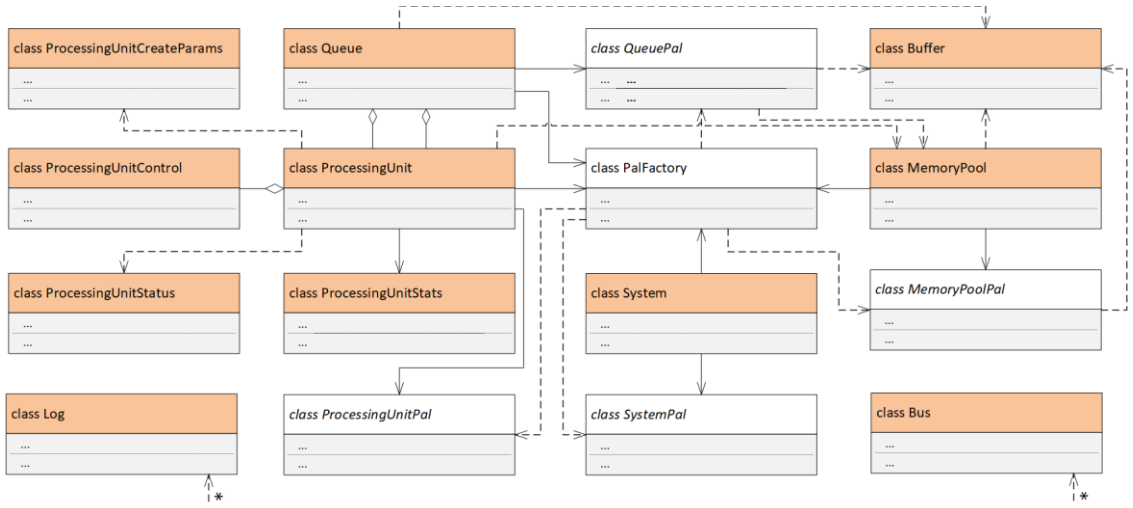
AMB слој апстракције платформе повезује горње AMB слојеве са самом платформом. Омогућава преносивост AMB језгра и апликација на различите платформе, омогућавајући поновно искоришћавање софтверских слојева изнад слоја апстракције платформе. За сваку платформу слој апстракције платформе

треба једном реализовати. Овај слој се директно ослања на развојни пакет хетерогене платформе, који обично долази од произвођача чипа. Слој апстракције платформе обухвата реализацију извршавања јединице обраде на одређеном језгру и пренос података између језгара. Такође, обухвата реализације разних апстракција везаних за меморију, систем (нити, семафори), руковаоце (сензори, екрани), методе оптимизоване за одређену платформу и слично.

У наставку је дат детаљан приказ архитектуре AMB средњег слоја софтвера.

### 3.2.1. AMB повезани слој

AMB повезани слој реализује логику средњег слоја, његове сервисе/јединице обраде и пружа униформну програмску спрегу ка апликацијама. AMB језгро омогућава механизам пајплајна реализујући апстрактне јединице обраде и везе међу њима. Слика 3.5 приказује основне класе језгра и спреге ка слоју апстракције платформе.




*Слика 3.5 Језгро средњег слоја са основним спрегама ка слоју апстракције платформе*

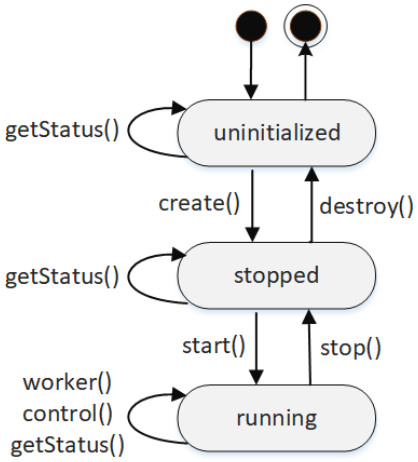
Јединица обраде представља један део пајплајна у оквиру кога се обрађује део апликације. Реализована је преко класе *ProcessingUnit*. Представља асинхрону нит, која се извршава на одређеном језгру (*CPU*, *GPU*, *DSP*, *VPU* или друго језгро) и комуницира са суседним јединицама обраде. Веза између ових јединица обраде је реализована преко класа *Queue* и *Buffer*. У оквиру једног циклуса обраде, јединица

обраде прихвати улазне податке преко AMB везе, обради податке и достави резултате излазној AMB вези. Овај циклус се понавља све док се јединица обраде експлицитно не заустави. Класе *ProcessingUnitCreateParams*, *ProcessingUnitControl*, *ProcessingUnitStatus* реализују параметризацију, руковање и статус јединицама обраде, док су системске функционалности (логовање, руковање меморијом, магистралама и сл.) смештене у класама *System*, *Log*, *Bus* и *MemoryPool*.

**AMB јединица обраде**

Слика 3.6 приказује дијаграм животног циклуса јединица обраде (стања и основне доступне функције за свако стање). Три су основна стања сваке јединице обраде:

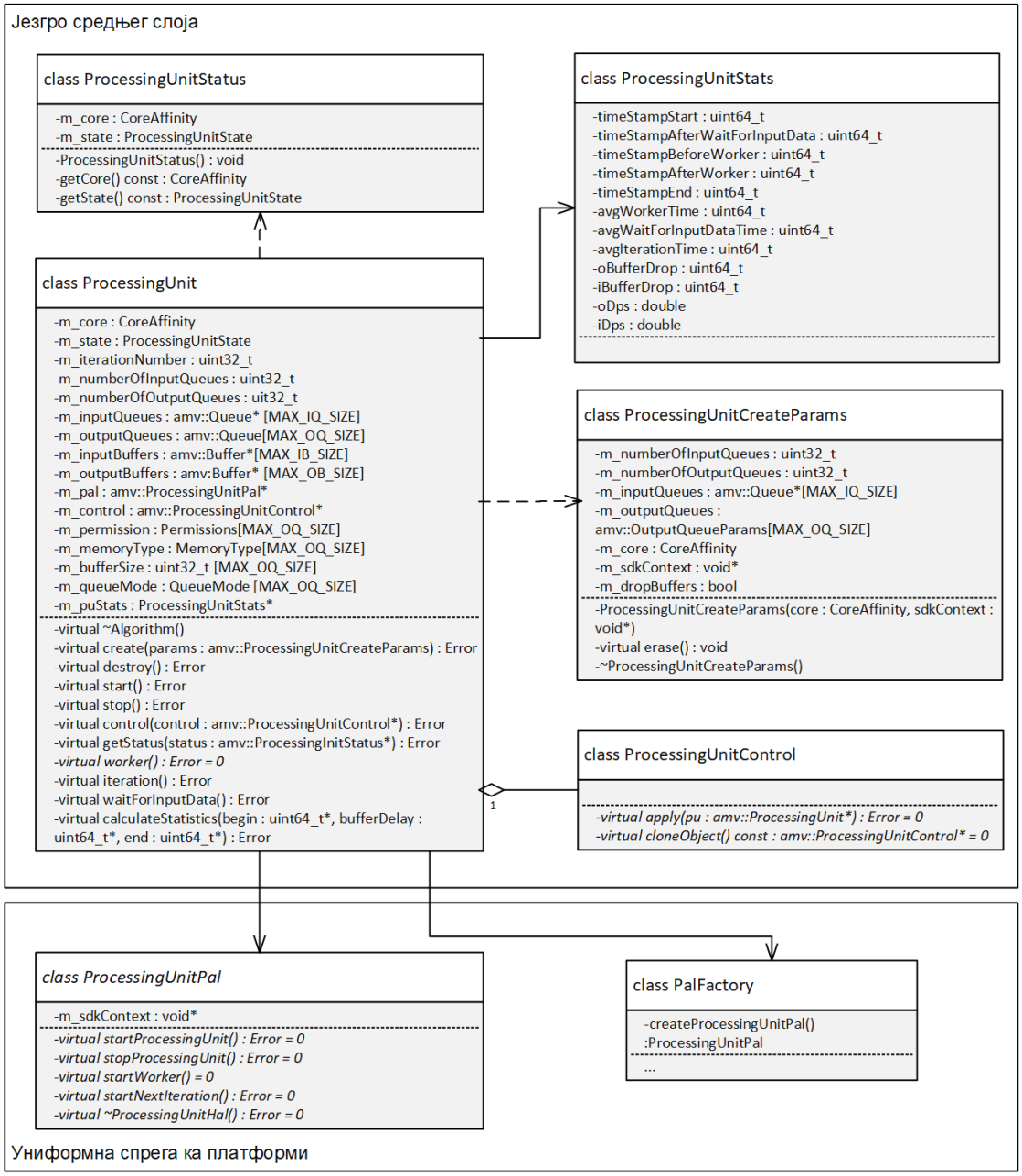
- неиницијализовано стање (енгл. *uninitialized*) – представља почетно и крајње стање; јединица обраде се налази у овом стању након позива конструктора, а пре позива методе *create*,
- стопирано стање (енгл. *stopped*) – представља стање у ком је јединица обраде спремна за рад (све потребне иницијализације и ресурси су прибављени), али није покренута,
- стање рада (енгл. *running*) – означава да је процес обраде у току, тј. да се обрада покреће итеративно.




*Слика 3.6 Животни циклус AMB јединице обраде*

Функционалност јединице обраде реализује више класа које су приказане на Слика 3.7. На овом и осталим дијаграмима у наставку нису приказани сви атрибути

и методе датих класа – одређене методе су изостављене ради боље прегледности (нпр. методе за добављање и постављање вредности атрибута).




*Слика 3.7 Дијаграм основних елемената јединице обраде*

Класа *ProcessingUnit* представља класу у којој је апстрахована реализација извршавања на одређеном језгру. Основне методе класе су приказане на слици, а односе се на животни век јединице обраде, на њено тренутно стање и управљање јединицом обраде и њеним итерација као и додатне помоћне методе. Метода *worker* није реализована у самом језгру, за разлику од осталих које имају основну

реализацију у језгру. Потребно је да изведене класе реализују ову методу. Остале методе се могу, и не морају наследити и проширити. Изведене класе обично реализују и методе везане за покретање и заустављање јединице обраде, проширујући их са (де)иницијализацијама које су специфичне за саму јединицу обраде. Следеће методе су од највећег интереса за изведене класе, тј. за корисника средњег слоја:

- *worker* – главна метода сваке јединице обраде, где се дефинишу операције над подацима. Ова метода се извршава на циљном језгру.
- *create/destroy* – стварање и уништавање пајплајна, излазних редова и прављење веза између јединица обраде, на основу параметара као што су жељено језгро извршавања и улазни редови.
- *start/stop* – покретање и заустављање нити јединица обраде, такође се користи за иницијализацију/деиницијализацију након што је пајплајн створен или пре уништавања.
- *control* – конфигурација јединице обраде, дозвољава измене одређених параметара током извршавања. Јединица обраде се обавештава о изменама у наредном *worker* циклусу.
- *getStatus* – добављање тренутног статуса јединице обраде (*ProcessingUnitStatus* класа). Додатни статуси се могу добавити наслеђивањем и проширењем класе *ProcessingUnitStatus*.

За креирање класе *ProcessingUnit* користе се информације садржане у *ProcessingUnitCreateParams* класи:

- параметри улазних веза (број, редови и њихови креирани објекти),
- параметри излазних веза (број, редови који треба да се креирају и објекти који треба да се сместе),
- језгро на ком се извршава јединица обраде,
- додатни системски (контекст, одбацивање бафера код препуњености редова) и кориснички параметри (специфични за саму природу обраде) који утичу на конфигурацију јединице обраде.

Класа *ProcessingUnit* се ослања на спрегу *ProcessingUnitPal* за извршавање операција специфичних за платформу. *ProcessingUnitPal* класа треба да се изведе

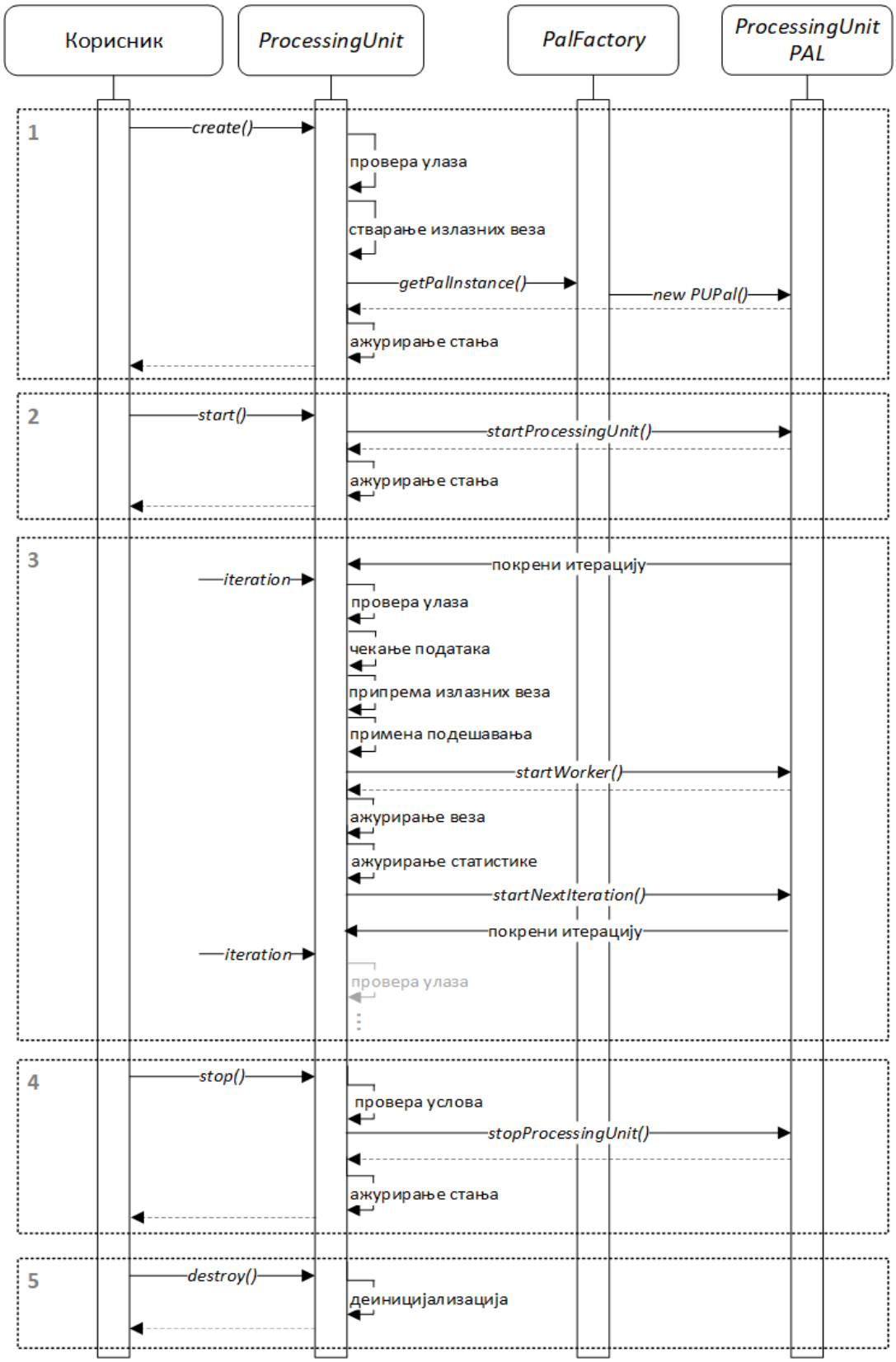
за сваку подржану платформу и инстанцира у току извршавања користећи *PalFactory* класу.

*ProcessingUnitControl* је интерфејс који користи *ProcessingUnit::control()* метода. Конкретна изведба јединице обраде ће дефинисати параметре којима се може управљати. Позивом методе *apply()* нови параметри се примењују.

Класа *ProcessingUnitStatus* пружа основне информације о статусу јединице обраде. За детаљнији статус о конкретној јединици обраде, препоручује се наслеђивање и проширивање ове класе са статусима у складу са конкретном обрадом.

Детаљи реализације јединице обраде у језгру средњег слоја су дати на Слика 3.8, у односу на интерфејсе ка платформи и животни циклус јединице обраде:

1. *create()* метода – стварање јединице обраде се врши у неколико корака:
    i. провера улазних параметара и иницијализација јединице обраде,
    ii. стварање излазних веза и прављење веза између јединица обраде, на основу параметара као што су жељено језгро извршавања и улазни редови,
    iii. стварање објекта јединице обраде који је карактеристичан за платформу - стварање се врши обраћајући се дефинисаним интерфејсима платформе,
    iv. прелазак у стање спремности за обраду.
2. *start()* метода – покретање јединице обраде:
    i. провера да ли су сви услови испуњени за покретање јединице обраде, обраћање конкретној реализацији јединице обраде на платформи да припреми и покрене процес обраде на одређеном језгру – итерацију по итерацију, и ажурирање стања јединице обраде.
3. извршавање итерација у току рада јединице обраде (енгл. *iteration*). У свакој итерацији сем обраде која се извршава у *worker()* функцији, врше се и додатне операције у оквиру *iteration()* методе, од чега су основне следеће:
    i. ако је статистика укључена, мерење почетног времена итерације,
    ii. провера улазних редова и података: чекање на податке улазних редова, тј. чекање да сви услови за покретање методе *worker*() буду испуњени,




*Слика 3.8 Јединица обраде*

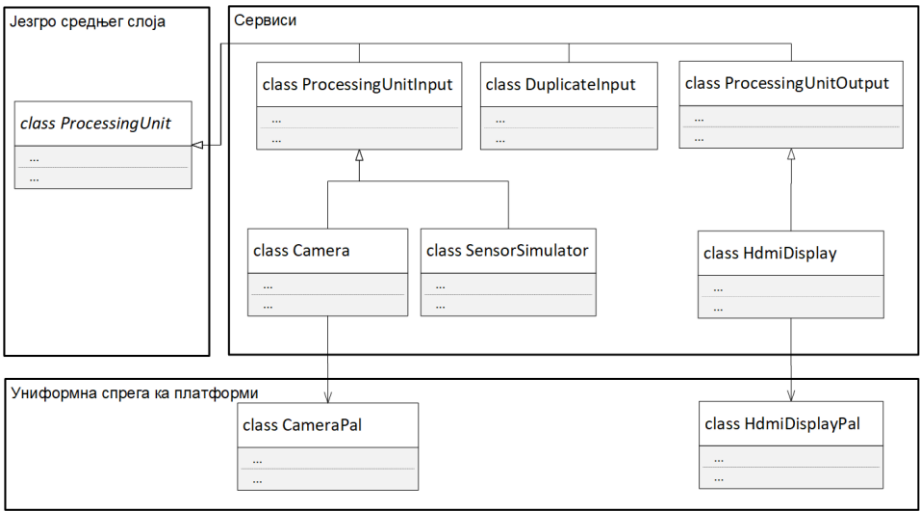
iii. ако је статистика укључена, мерење почетног времена итерације,
iv. провера улазних редова и података: чекање на податке улазних редова, тј. чекање да сви услови за покретање методе *worker*() буду испуњени,
v. провера излазних редова и података: провера параметара редова и ослобађање ресурса који нису потребни, добављање свих излазних бафера података,
vi. ако је статистика укључена, мерење тренутног времена: време кашњења узроковано радом са редовима и баферима,
vii. провера да ли је *control()* метода позвана у међувремену: ако јесте, примена нових подешавања позивом *apply()* методе.
viii. извршавање *worker()* методе која чита улазне податке, врши обраду и попуњава излазне податке,
ix. ажурирање статуса излазног реда са резултатом обраде, обавештавање следеће јединице обраде у пајплајну да постоје нови подаци у њеном улазном реду,
x. означавање бафера у улазном реду као искоришћених како би власник, тј. претходна јединица обраде, могао да их очисти,
xi. ако је статистика укључена, мерење крајњег времена итерације и израчунавање додатних статистичких података као што су средње време извршавања итерације, средње време обраде бафера и редова итд.

4. *stop()* метода - заустављање јединице обраде.
   i. ако постоји покренута итерација, чека се док се тренутна итерација не заврши и након тога се приступа заустављању и ажурирању стања јединице обраде,
5. *destroy()* метода – деиницијализација и ослобађање ресурса које је јединица обраде користила.

### AMB сервиси

Конкретне реализоване јединице обраде представљају сервисе. Корисници могу да их комбинују у сврху креирања пајплајна. Основна идеја средњег слоја је да постоји скуп сервиса који долази реализован са средњим слојем, али исто тако корисницима се пружа начин да сами реализују своје сервисе на једноставан начин у складу са потребама.

Реализовани сервиси у оквиру AMB средњег слоја се углавном односе на управљачке и улазно/излазне сервисе. На пример, улазни сервиси укључују апстракцију стварних сензора (*Camera* и *UltrasonicSensor* сервиси), симулаторе података у возилу као што су брзина, број обртаја, информације са *CAN* магистрале (*VehicleDataSimulator* сервис) и помоћне сервисе за читање из датотека, са мреже и слично. Излазни сервиси укључују апстракцију екрана (*HDMI*, инфо-забавних екрана у возилу (енгл. *HeadUnit*)) и помоћне сервисе за упис у датотеке и слање садржаја преко етернет мреже. Сви ови сервиси су изведене класе *InputPU/OutputPU* класа које наслеђују *ProcessingUnit* класу.

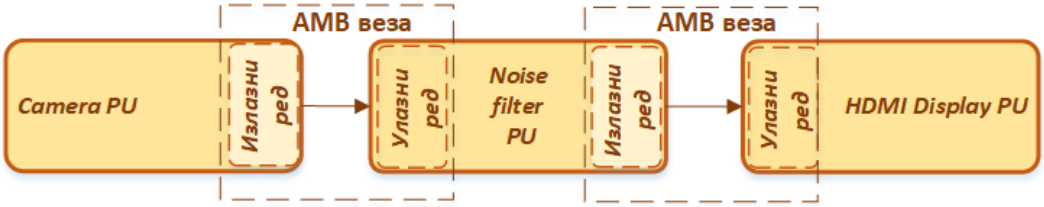



*Слика 3.9 Дијаграм класа примера сервиса*

Слика 3.9 приказује дијаграм класа примера сервиса у склопу AMB средњег слоја. Додавање новог сервиса се врши наслеђивањем једног од постојећих сервиса или класе из језгра и његовим проширивањем и реализацијом за дате потребе. У оквиру сервиса за његову реализацију могу се користити доступне библиотеке отвореног кода. Уколико је сервис зависан од ресурса саме платформе (нпр. камере), онда је потребно проширити спрегу ка платформи.

### AMB веза између јединица обраде

AMB веза се простире на две јединице обраде, и преноси податке у једном смеру. Изворна јединица обраде посматра AMB везу као излазни ред (енгл. *output queue*), док циљна јединица обраде посматра исту везу као улазни ред података (енгл. *input queue*) (Слика 3.10).




*Слика 3.10 AMB веза на примеру најплајна*

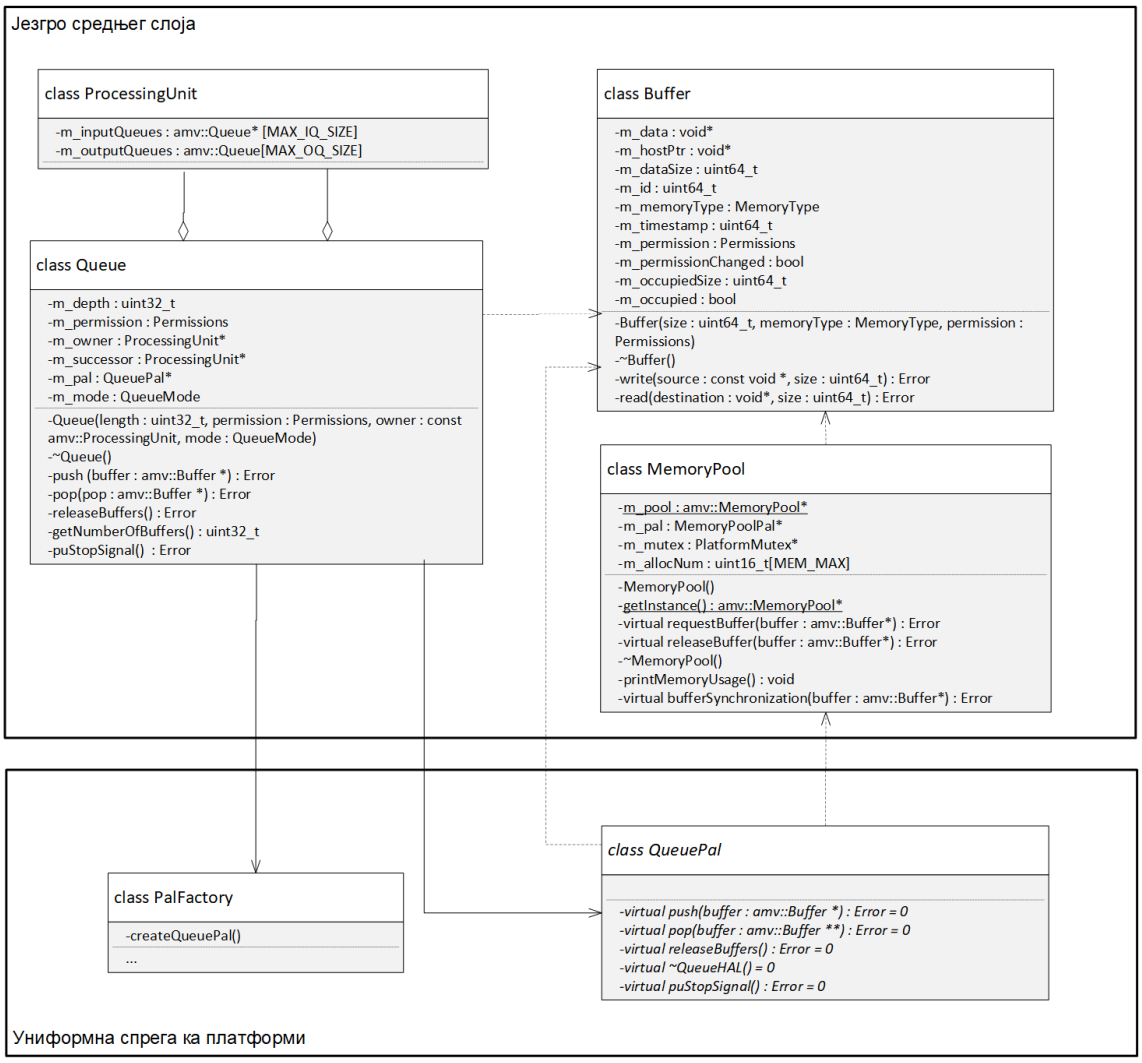
Између јединица обраде такође могу постојати вишеструке везе, чиме се повезује нпр. једна јединица обраде са више других јединица обраде. Свака јединица обраде може имати нула или више улазних и излазних редова. Јединица обраде поседује свој излазни ред и рукује њиме. Ред се састоји од секвенце података. Редови се реализују користећи *Queue* класу, а подаци се реализују преко *Buffer* класе.

Подаци су представљени меморијским регионом у меморијском базену (енгл. *MemoryPool*). У зависности од тога на ком језгру се налазе јединице обраде, меморијски регион је лоциран у делу базена коме могу приступити обе јединице обраде. Показивач на податке се дели између јединица обраде. Показивачи на податке могу да се пропагирају из улазног у излазни ред у оквиру јединице обраде (тј. исти меморијски регион који је примљен као улазни податак у јединицу обраде се прослеђује следећој јединици обраде). На овај начин, избегава се потреба са дубоким копирањем података из једног у други меморијски регион.

Из перспективе приступа меморији, подацима се може приступати у два режима: режим за читање (непроменљиви) и режим за читање и писање (променљиви). Када подаци достигну непроменљиви режим, не могу се више мењати у текућој или у наредним јединицама обраде. Овај принцип је уведен зато што правилна примена може да доведе до значајне оптимизације управљања меморијом. Слој апстракције платформе и хетерогено платформско окружење воде

рачуна о меморијским регионима и одређивању (у току процеса компајлирања или у току извршавања) ко може да приступи подацима дате везе како би се оптимизовала алокација и руковање меморијама и како би се смањило копирање података између различитих меморијских складишта.

Како је *Buffer* само меморијски регион, наслеђивањем ове класе и дефинисањем нових метода уводе се додатни контексти и интерпретације меморије (нпр. *ImageBuffer* класа са методама *getWidth*, *getHeight*, *getBytesPerPixel*, итд.).




*Слика 3.11 Дијаграм класа AMB везе*

Слика 3.11 представља дијаграм основних класа које учествују у реализацији AMB везе између јединица обраде. Класа *Queue* представља ФИФО (енгл. *First In Frist Out - FIFO*) листу чији су елементи *Buffer* класа или класе изведене од *Buffer* класе. Главне методе *Queue* класе су методе за додавање елемената (*push()*) и за

узимање елемената са почетка реда (*pop()*). Објекат јединице обраде (*ProcessingUnit*) креира *Queue* објекат, који постаје један од излазних редова. Као параметар конструктору се прослеђује највећи број могућих редова (*length*), а тренутни број редова се чува у атрибуту *m_length*.

Сваки ред је у исто време излазни ред једне јединице обраде и улазни ред у следећу јединицу обраде. Према томе, ред има дефинисане дозволе приступа када се посматра као излазни ред, као и потенцијално другачије дозволе приступа када се посматра као улазни ред. Дозволе приступа над улазним редом морају да буду исте или рестриктивније него ли над излазним редом.

Класа *Queue* не поседује листу елемената као чланове класе, него се ослања на класу *QueuePAL* да приступи стварним елементима који зависе од реализације на конкретној платформи. Елемент реда је реализован као *Buffer*, који се креира и уништава користећи *MemoryPool* класу. Поред самих алоцираних података (*m_data*, *m_dataSize*), класа *Buffer* садржи следеће групе метода:

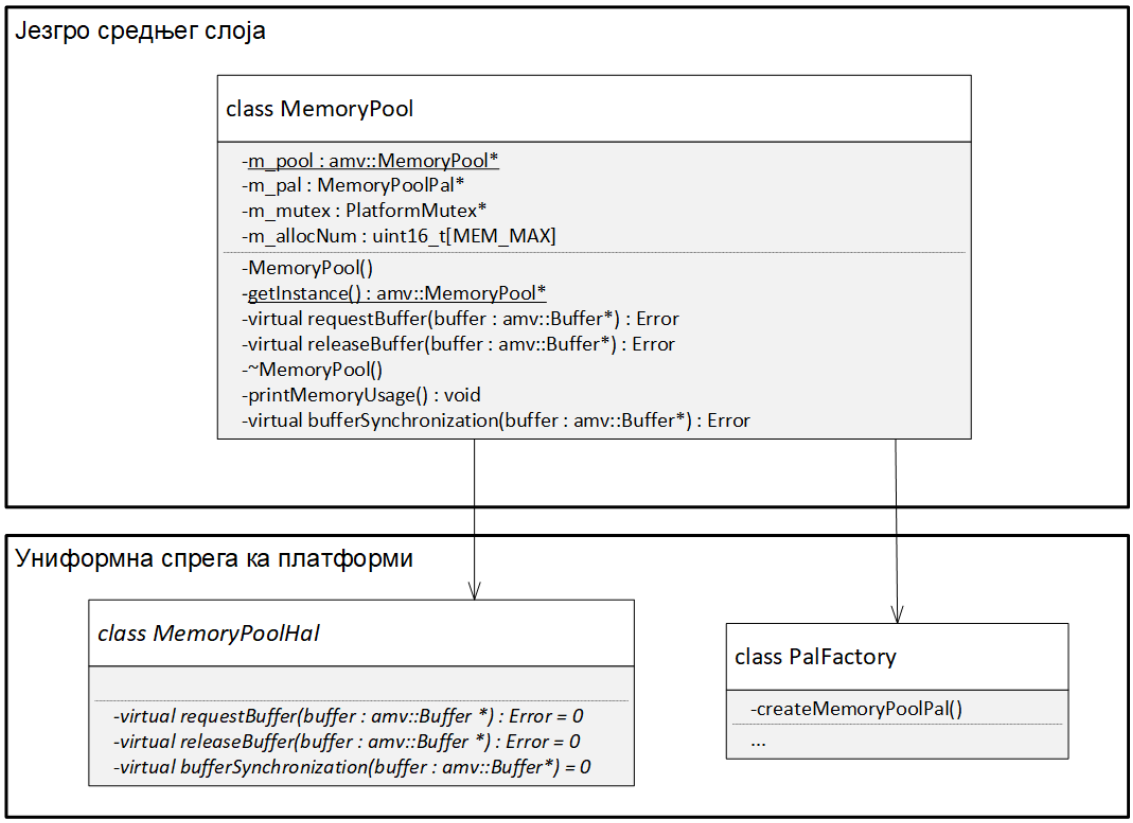
- тумачење података на које показује показивач (*getTimestamp()*, *setTimestamp*()),
- руковање подацима (*writeToBuffer()*, *readFromBuffer()*),
- руковање животним циклусом података (*markBufferDrained()* – означава објекат као искоришћен и спреман за брисање, јединица обраде која је власник објеката треба да га поново искористи или да ослободи ове податке).

Препоручује се наслеђивање *Buffer* класе, и дефинисање додатних метода како би се олакшао приступ информацијама које су специфичне за саме податке који се смештају.

*MemoryPool* класа рукује свим захтевима за меморију. Ослања се на платформску реализацију која поседује знање о доступној меморији на датој платформи. Јединица меморије са којом ова класа рукује је *Buffer*. *MemoryPool* класа је реализована користећи уникатни образац (енгл. *singleton pattern*). Креира се током *PalFactory* креирања и приступа јој се користећи *MemoryPool::getInstance* методу.

Слика 3.12 приказује дијаграм класа за реализацију меморијског базена. *MemoryPool* подржава операције захтева за заузимањем меморије и ослобађања

меморије. *Buffer* се може заузети у једном од неколико постојећих меморијских складишта, тј. типова меморије. Складиште се селектује за време креирања бафера.




*Слика 3.12 Дијаграм класа MemoryPool*

Пример општих меморијских типова дат је у наставку. Постојање одређеног меморијског типа зависи од подршке и реализације на конкретној платформи:

- *DDR_NON_CACHED_SR* - *Heap* меморија у *DDR*, некеширана, дељена између јединица обраде. Препоручује се за коришћење малих структура података које се деле између јединица обраде.
- DDR_CACHED_SR - *Heap* меморија у *DDR*, кеширана, дељена између јединица обраде. Препоручује се за коришћење великих структура података које се деле између јединица обраде.
- FAST_SR - *Heap* меморија у *DDR, L2* (или другде), кеширана или некеширана, зависно од чипа и конфигурације, дељена између јединица обраде. Препоручује се за коришћење код потреба за брзим приступом привременим подацима.

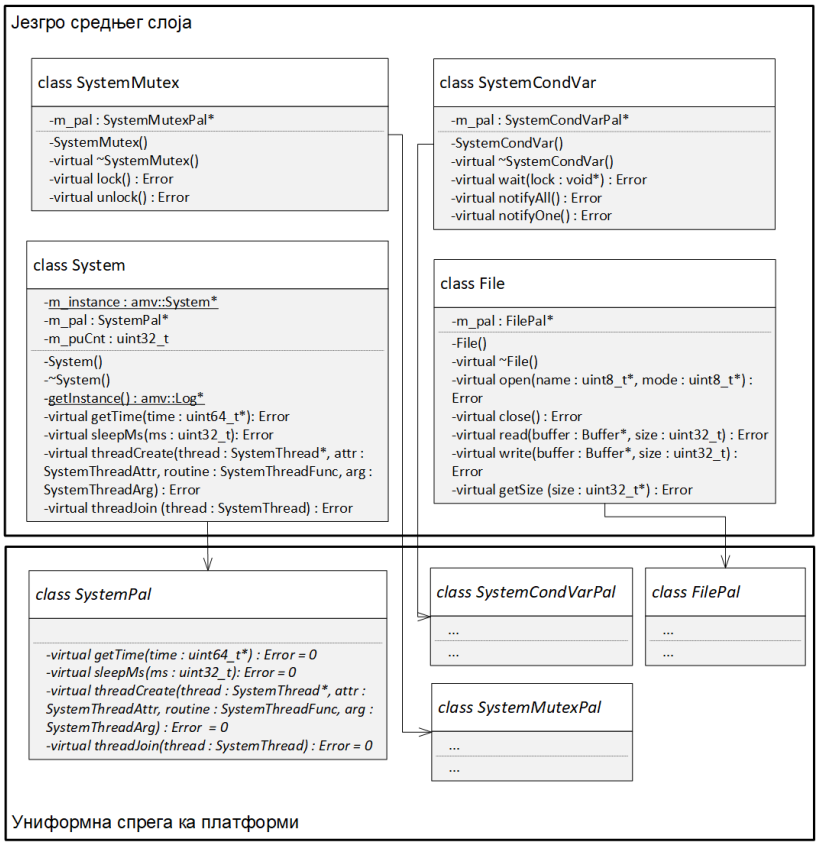
- *DDR_CACHED_LOCAL* - *Heap* меморија у *DDR*, кеширана, видљива једино локалној јединици обраде. Препоручује се за коришћење малих структура података које су потребне локалној јединици обраде.

### AMB системске функционалности

Системске функционалности обухватају руковање оперативним системом, исписима, типовима података (од основних типова : 8, 16, 32 и 64-битни означени и неозначени типови података), начину смештања података (енгл. *endianess*), механизмима синхронизације, датотекама и слично. У наставку је дат преглед.

*Класе System, File, SystemMutex и SystemCondVar*

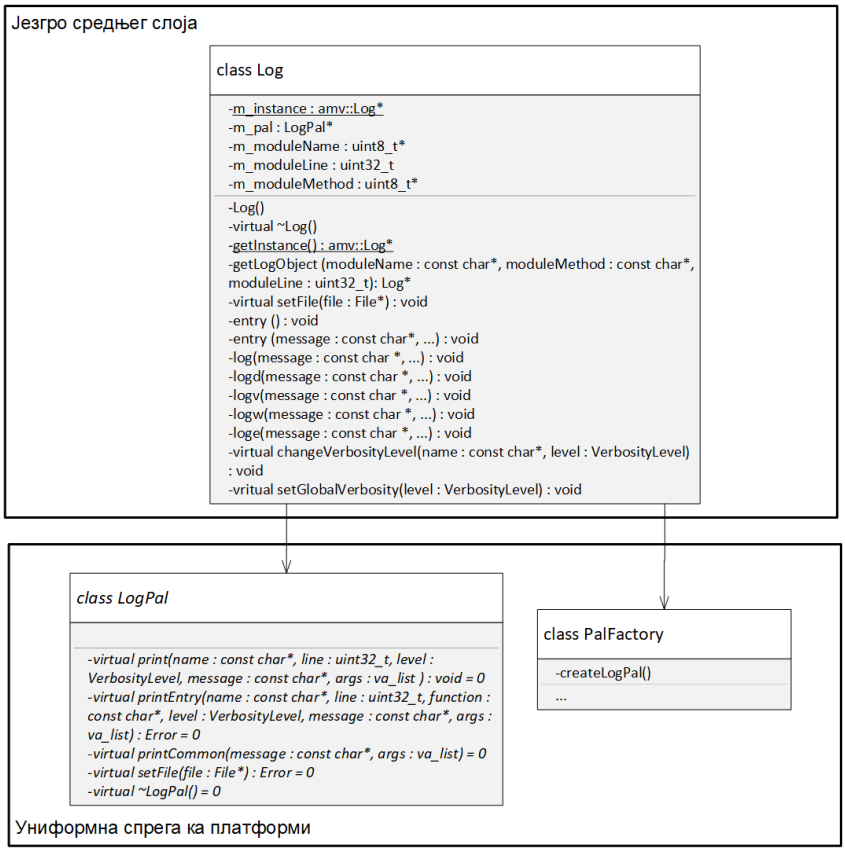
Класа *System* пружа спреге за комуникацију са оперативним системом и платформом уопште. Ослања се на *SystemPAL* класу у којој је имплементација везана за циљну платформу. Методе које ова класа апстрахује су везане за рад са временом и нитима. Додатне класе апстрахују рад са датотекама, и синхронизационим механизмима који су специфичне за саму платформу.




*Слика 3.13 Дијаграм системских класа*

*Класа Log*

Класа *Log* подржава неколико нивоа исписа (*verbose*, *debug*, *error*, *warning*) са могућношћу промене нивоа током извршавања за сваки модул. Такође, постоји могућност пружања статистичких података за сваку јединицу обраде у смислу количине протока података, просечног времена обраде, времена чекања на податке, броја итерација и слично.




*Слика 3.14 Дијаграм класе Log*

**AMB спрега ка платформи**

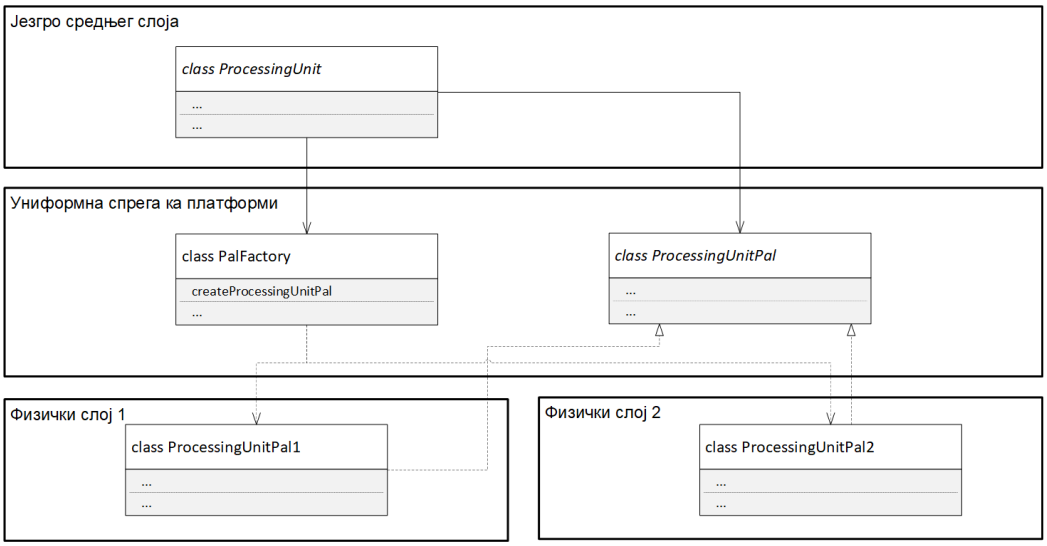
Спрега ка платформи је дефинисана на униформан начин, у сврху брже и једноставније преносивости и реализације на више платформи. Овај слој се састоји од скупа спрега (енгл. *interface*) који апстрахују платформу. Горњи делови повезаног слоја користе ове спреге, док се сама реализација налази у оквиру физичког слоја и специфична је за сваку платформу.

При реализацији се користи *C++* фабрички образац метода (енгл. *factory method pattern*) – који омогућава да класа делегира одговорност за креирање одговарајућег објекта. Фабрички образац метода инстанцира одговарајућу

подкласу на основу информација које му даје клијент или на основу информација које извлачи из тренутних стања. *PALFactory* класа се реализује користећи уникатни образац (омогућава да класа буде инстанцирана једном и да поседује глобалну тачку приступа) и објекту се приступа преко статичке методе *PALFactory::getInstance()*. Касније се користи да креира објекте слоја апстракције платформе.

Слика 3.15 приказује фабрички образац метода на примеру AMB јединице обраде. Исти принцип се користи и за остале класе. Јединица обраде (енгл. *ProcessingUnit*) за инстанцирање објекта *ProcessingUnitPal* класе позива методу *createProcessingUnitPal* класе *PalFactory*. У овој методи је садржана логика која реализација физичког слоја ће се креирати. *ProcessingUnitPal* представља спрегу за јединицу обраде. *ProcessingUnitPal1* и *ProcessingUnitPal2* представљају класе које реализују ову спрегу. *PalFactory* представља фабрику која пружа методе за креирање објеката, у овом случају *createProcessingUnit* методу.

На овај начин логика за одлучивање који објекат ће бити креиран се налази на једном месту.




*Слика 3.15 Фабрички образац метода на примеру јединице обраде*

### 3.2.2. AMB физички слој

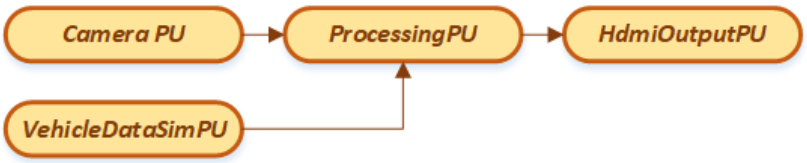
Слој апстракције платформе – ПАЛ (енгл. *Platform Apstraction Layer*) представља спрегу која се састоји од скупа функција који омогућава средњем слоју да се извршава без знања о детаљима архитектуре платформе и хетерогеног

развојног окружења. Ослања се на библиотеке из развојног окружења и захтева посебну реализацију за свако развојно окружење/платформу. ПАЛ апстрахује извршавање јединице обраде, руковање редовима и подацима, оперативним системом и платформом уопште, укључујући сензоре, екране, разне функције оптимизоване за дату платформу и сл.

Овај слој представља конкретну реализацију спрега слоја униформне спреге ка платформи. За сваку платформу, реализује се по један физички слој. За основно функционисање средњег слоја потребно је реализовати спреге који се користе за реализацију језгра средњег слоја, док је реализација сервиса произвољна и зависи од корисничког случаја, потреба и доступних ресурса на платформи.

### 3.2.3. Пример коришћења AMB корисничке програмске спреге

Пример изворног кода креирања апликације од четири градивна блока као на Слика 3.16 користећи AMB програмску спрегу је приказан испод. Провере повратних вредности су прескочене због једноставности приказа.




*Слика 3.16 Пример апликације*

```
using namespace amv;
CameraPU m_camera;
VehicleDataSimPU m_speed;
ProcessingPU m_process;
HdmiOutputPU m_display;

/* стварање блока за прихват података са камере */
CameraCreateParams m_camParams = CameraCreateParams(E_CORE_CPU0);
m_camParams.m_numberOfOutputQueues = 1;
m_camParams.m_outputQueue[0].m_size = 3;
m_camParams.m_outputQueue[0].m_bufferSize = BUFFER_SIZE;
m_camParams.m_numberOfInputQueues = 0;
m_camParams.m_numberOfCameras = 1;
m_camParams.m_cameras = E_CAMERA_MIRROR_RIGHT;
err = m_camera.create(&m_camParams);

/* стварање блока за симулацију брзине возила */
VehicleDataSimParams m_speedParams = VehicleDataSimParams(E_CORE_CPU1);
m_speedParams.m_numberOfOutputQueues = 1;
m_speedParams.m_outputQueue[0].m_size = 3;
m_speedParams.m_outputQueue[0].m_bufferSize = BUFFER_SIZE;
m_speedParams.m_numberOfInputQueues = 0;
m_speedParams.m_data = E_SPEED;
err = m_speed.create(&m_ speedParams);

/* стварање обрадног блока */
ProcessingParams m_procParams = ProcessingCreateParams(amv::E_CORE_DSP0);
m_procParams.m_warning = E_VISUAL;
m_speedParams.m_numberOfOutputQueues = 1;
m_speedParams.m_outputQueue[0].m_size = 3;
m_speedParams.m_outputQueue[0].m_bufferSize = BUFFER_SIZE;
m_procParams.m_numberOfInputQueues = 2;
m_procParams.m_inputQueues[0] = m_camParams.m_outputQueue[0].m_queue;
m_procParams.m_inputQueues[1] = m_speedParams.m_outputQueue[0].m_queue;
m_processing.create( &m_procParams);
```

```
/* стварање блока за приказ резултата */
HdmiOutputCreateParams m_dispParams = HdmiOutputCreateParams();
m_dispParams.m_width = 1920;
m_dispParams.m_height = 1080;
m_dispParams.m_format.m_imageFormat = E_YUV_422;
m_dispParams.m_numberOfInputQueues = 1;
m_dispParams.m_inputQueues[0] = m_procParams.m_outputQueue[0].m_queue;
m_display.create( &m_dispParams);

/* покретање AMB пајплајна */
m_camera.start();
m_speed.start();
m_processing.start();
m_display.start();

...

/* заустављање пајплајна */
m_camera.stop();
m_speed.stop();
m_processing.stop();
m_display.stop();

/* рачунање статистике */
m_camera.calculateStatistics(&begin, &delay, &end);

/* уништавање пајплајна */
m_camera.destroy();
m_speed.destroy();
m_processing.destroy();
m_display.destroy();
```

# 4. ПРЕДЛОГ МЕРА ЗА ОЦЕНУ КВАЛИТЕТА РЕШЕЊА

Предложено решење треба да задовољи разноврсне групе захтева (функционалне и нефункционалне). У складу са тим дефинише се више врста мера којима ће се дато решење верификовати:

- *Експерименталне провере* којима се углавном верификују функционалности софтверске платформе, на најмање две различите хардверске платформе. Експерименталне провере се врше извршавањем специфичних корисничких случајева као и реализацијом система који треба да осликава реалан АДАС систем.
- *Мерење перформанси система* у реализованим корисничким случајевима и примеру система на циљним платформама – могућност рада у реалном времену.
- *Софтверске метрике* за оцену квалитета софтвера (ХИС, објектно-оријентисане метрике, атрибути квалитета софтвера).
- *Додатна разматрања* која се односе на нефункционалне захтеве који нису обухваћени претходним мерама, а дискутују о предложеним концептима.

| **Групе захтева** | | **Начин провере** |
|---|---|---|
| *Апстраховање хетерогености* | *Хардверске платформе (језгра, меморија, комуникација), сензори и актуатори, магистрале* | Експериментална провера – кориснички случајеви и систем апликација |
| *Руковање и интеграција апликација у реалном времену* | *Руковање апликацијама, спрега са другим доменима, коришћење хетерогених спрега* | Експериментална провера – кориснички случајеви и систем апликација |
| | *Рад у реалном времену* | Мерење перформанси система (кориснички случајеви и систем апликација) |
| *Помоћне функције за развој и тестирање* | *Симулатори, логовање, добављање статистичких података итд.* | Експериментална провера – кориснички случајеви и систем апликација |
| *Нефункционални захтеви* | *Архитектурни* | Софтверске метрике |
| | *Безбедносни, сигурносни* | Дискусија концепта |

*Табела 4.1 Предлог начина провере група захтева*

Табела 4.1 приказује основне групе дефинисаних захтева и изазова које је потребно покрити и начин провере којима ће дати захтеви бити оцењени. Свака од ових група захтева доприноси бржем развоју и интеграцији апликација у хетерогеном окружењу модерних возила. У наставку се налази детаљан опис сваке од група мера.

Додатно, на крају овог поглавља је дат предлог поређења са другим решењима. Поређење је одрађено на нивоу целовитости решења - који аспекти су размотрени у самим решењима (узевши у обзир да не постоје доступне детаљне информације за велики број решења).

## 4.1. ЕКСПЕРИМЕНТАЛНА ПРОВЕРА

Експериментална провера обухвата проверу функционалности дефинисану захтевима у реализованом систему на циљној платформи. Провера се обавља дефинисањем и реализацијом AMB апликација.

### 4.1.1. Кориснички тестни случајеви

Кориснички тестни случајеви се заснивају на дефинисању AMB проточног тока података за једну или више специфичних функционалности која треба да се провери. Тестни случајеви су излистани у Додатку Б. На примеру тестног случаја са Слика 4.1 (тестни случај 4 из Додатка Б.) проверава се више ствари, тј. постојање апстракција на хетерогеној циљној платформи:

- за бар две врсте процесорских језгара (*CPU, GPU*),
- за више процесорских језгара исте врсте (*CPU0, CPU1*).
- за комуникацију између поменутих језгара,
- за бар једну врсту сензора – камера,
- за спрегу са приказом на екран (инфо-забавни домен и/или ХДМИ екран).


*Слика 4.1 Кориснички тестни случај 4*

### 4.1.2. Систем апликација

Да би се показале и демонстрирале могућности предложеног средњег слоја, сем саме реализације средњег слоја, потребно је реализовати један или више система апликација преко предложеног средњег слоја, који представљају реалан случај у аутомобилској индустрији. Систем апликација треба да задовољи следеће критеријуме:

- извршавање на два или више хетерогених чипова,
- извршавање на више различитих процесорских језгара,
- реализација две или више апликација, (тестних или комерцијалних) и приказ руковања апликацијама,
- коришћење већег броја камера (више од 5) које нису нужно истих перформанси,
- коришћење више врста магистрала за пренос података између обрадних јединица.

## 4.2. ПРОВЕРА ПЕРФОРМАНСИ

AMB средњи слој представља још један софтверски слој и самим тим уноси додатно кашњење у обради. Као такав, не треба да онемогући извршавање система у реалном времену. Дефинишу се следећа мерења перформанси на циљној платформи у лабораторијским условима: мерење времена размене података између јединица обраде (само извршавање јединице обраде је јако зависно од саме обраде), и мерење броја обрађених слика у секунди за реализовани систем апликација (како би се показала способност извршавања целокупног система, који користи предложени средњи слој, у реалном времену).

### 4.2.1. Мерење времена размене података између јединица обраде

Код прилагођавања апликације одређеној платформи, поред расподеле на разна језгра, потребно је знати колико времена узима комуникација и дистрибуција података између језгара/чипова. Количина података која се размењује (нпр. слике са камере) није занемарива и ова размена података утиче на свеукупне перформансе апликације. Перформансе зависе од циљне платформе (доступни акцелератори,

организација меморије, методе за пренос података, итд.) и од реализације AMB слоја за апстракцију платформе. Предлаже се мерење између две суседне јединице обраде које обухвата време потребно за извршавање AMB апстракција језгра и времена преноса података између ове две јединице обраде у односу на количину података која се размењује.

### 4.2.2. Мерење броја обрађених слика у секунди

У реализованом систему апликација који осликава реални систем потребно је измерити број обрађених слика у секунди за сваку од апликација. На овај начин се показује могућност рада свеукупног система у реалном времену. Прихватљиви број обрађених слика у секунди зависи од саме природе апликација и алгоритама.

## 4.3. СОФТВЕРСКЕ МЕТРИКЕ

Током развоја софтвера за аутомобилску индустрију потребно је применити више од једног стандарда квалитета ( [59], [77]). Норме и стандарди квалитета се односе између осталог и на метрике које треба да помогну у процени квалитета софтвера. Често недостаје јасно разумевање које метрике (разуман скуп метрика) адресирају одређени атрибут квалитета. Такође, у зависности од *ASIL* нивоа, метрике се могу различито тумачити у односу на опсеге вредности. Као резултат тога, присутна је борба са неодговарајућим метрикама, које се морају прилагодити и пажљиво документовати.

Због свега горе наведеног, ХИС конзорцијум је развио стандардизовани скуп метрика за софтвер у аутомобилској индустрији. Ове метрике циљају *C* програмски језик без категоризације по питању величине и типа пројекта као и нивоа безбедности. ХИС метрике не дају дефиницију граничних вредности за разне типове пројеката, као ни пресликавање метрика на атрибуте квалитета софтвера. Узевши у обзир да се данас у аутомобилској индустрији користе и други програмски језици сем *C*-а, примењивост ХИС метрика је ограничена. У пракси ово доводи до сталног преговора око граничних вредности метрика између добављача и произвођача аутомобила.

Због непостојања стандардног скупа метрика квалитет софтвера је тешко упоредити између различитих пројеката, па чак и између различитих софтверских

компоненти у оквиру истог пројекта. Метрике за оцену квалитета софтвера као и њихово рачунање и примена предмет су бројних дискусија [130], [131], али већина се слаже са чињеницом, да је код примене оваквих метрика, потребно разумети шта се тачно мери и у ком контексту како би тумачење добијених вредности било релевантније.

У наставку је дат предлог мера за оцену квалитета решења и осврт на атрибуте квалитета софтвера на које се те мере односе:

- ХИС метрике (као стандардизоване метрике за аутомобилски софтвер),
- објектно-оријентисане метрике (у складу са предложеном архитектуром),
- додатне метрике за атрибуте квалитета софтвера.

Табела 4.2 представља начине провере архитектурних захтева, тј. која од група метрика верификује одређене захтеве. Такође, у наредним потпоглављима, за сваку од група је дата дискусија како које од метрика утиче на време развоја.

| Архитектурни захтеви | ХИС метрике | ОО метрике | Атрибути квалитета |
|---|---|---|---|
| *Захтев 21. Проширивост* | | | X |
| *Захтев 22. Модуларност* | | | X |
| *Захтев 23. Степен сложености одржавања* | X | X | X |
| *Захтев 24. Портабилност* | X | | X |
| *Захтев 25. Тестабилност* | | X | X |
| *Захтев 26. Поновно искоришћење* | | X | X |
| *Захтев 28. Усклађеност са добрим праксама* | X | X | |

*Табела 4.2 Начин провере архитектурних захтева*

### 4.3.1. ХИС метрике

ХИС конзорцијум одређује основни скуп метрика при евалуацији софтвера. За одређени број метрика даје и очекивани опсег вредности. У Табели 4.3 је приказан подскуп ХИС метрика које ће бити мерене, њихов опис као и предложени опсег од стране конзорцијума. ХИС метрике такође утичу на одређене атрибуте квалитета софтвера. Табела 4.4 приказује мапирање утицаја предложених метрика према [130] на атрибуте из ИСО 25010 стандарда.

| Име | Опис | Опсег |
|---|---|---|
| *Густина коментара* | Однос броја линија коментара (унутар и ван функција) у односу на број исказа. Односи се на разумљивост и јасноћу изворног кода. | >0.2 |
| *Број скокова* | Број *goto* наредби. Утиче на велико повећање броја могућих путања извршавања кода и самим тим отежава тестирање. | 0 |
| *Циклична сложеност* | Мера сложености - [104]. Зависи од броја тачака одлучивања или условних исказа. | 1-10 |
| *Број параметара функције* | Представља меру комплексности спреге функције. | 0-5 |
| *Број инструкција по функцији* | Представља меру комплексности функције. | 1-50 |
| *Број излазних тачака* | Представља број излазних тачака из функције. Односи се на мере комплексности и одржавања. | 0-1 |
| *Опсег језика* | Представља индикатор трошкова одржавања и/или промене функција. Рачуна се као однос суме броја свих оператора и операнада и суме броја различитих оператора и операнада. | 1-4 |

*Табела 4.3 ХИС метрике*

| ИСО атрибути | | ХИС метрике | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *Густина коментара* | *Број скокова* | *Циклична сложеност* | *Број параметара функције* | *Број инструкција по функцији* | *Број излазних тачака* | *Учестаност речника* |
| *Способност одржавања* | *модуларност* | | x | x | x | x | | |
| | *поновно коришћење* | | x | | | | | |
| | *могућност анализирања* | x | x | x | x | x | x | x |
| | *променљивост* | | | | x | x | | |
| | *способност тестирања* | | x | x | x | | x | |
| *Портабилност* | *заменљивост* | | x | | x | | x | |
| *Употребљивост* | *могућност учења* | | x | | | | | x |
| *Ефикасност перформанси* | *временско понашање* | | | | | x | x | |

*Табела 4.4 Утицај ХИС метрика на атрибуте квалитета софтвера*

### 4.3.2. Објектно-оријентисане метрике

Предложена архитектура је заснована на објектно-оријентисаним принципима. За оцену квалитета објектног пројектовања, веза између објеката, као и других објектно-оријентисаних принципа постоји више скупова објектно-оријентисаних метрика, као што су Чидамберов и Кемереров скуп метрика [133],

Лоренцов и Кидов скуп метрика [134], МООД скуп метрика [135], и други. У овом раду определили смо се за Чидамберову и Кемерерову групу метрика. Такође, у прегледу литературе за метрике у развоју софтвера у аутомобилској индустрији у [130], међу коришћеним метрикама су метрике из Чидамберове и Кемерерове групе, чији је преглед дат у табели испод.

| Назив метрике | Опис метрике |
|---|---|
| *Сложеност пондерисаних метода – WMC (енгл. Weighted Methods per Class)* | Представља сложеност класе, тј. суму сложености метода у класи. Класе са већом вредности указују да су то потенцијално класе које су специфичне за саму апликацију и смањују могућност поновног искоришћења истих. Метрика указује на то колико времена и труда је потребно за развој и одржавање класе. Различите вредности се дефинишу као граничне за ову метрику. Један од приступа дефинисања граничних вредности је да се ограничи број метода по класи на нпр 50., а други је да се за нпр. највише 10% класа дозволи да имају више од 24 методе. Други приступ омогућава постојање великих класа, али већина класа би требале да имају мању вредност. |
| *Дубина стабла наслеђивања – DIT (енгл. Depth of Inheritance Tree)* | Представља највећи број нивоа наслеђивања од посматране класе до корене класе. Што је дубље класа у хијерархији класа, изгледније је да ће се наследити више метода и атрибута и самим тим повећати сложеност и утицати на одржавање и поновно искоришћење класе. Велика вредност ове метрике означава већу склоност ка грешкама у софтверу и то не нужно за класе које су на дну наслеђивања. Препоручена вредност за дубину стабла наслеђивања је пет и мање, док се већина аутора слаже да је потребно првенствено сагледати контекст код ове метрике, да ли је у питању систем, део софтвера, сврха софтвера итд. |
| *Број подкласа – NOC (енгл. Number of Childern)* | Представља број непосредних подкласа посматране класе у хијерархији наслеђивања. Мери ширину хијерархије класа (за разлику од *DIT*-а који мери дубину). Велики број подкласа може да означава неколико ствари: велико искоришћење основне класе, основна класа захтева веће тестирање, неправилна апстракција основне класе, погрешно коришћење подкласа (нпр. потребно је груписање повезаних класа и увођење још једног нивоа наслеђивања). У зависности од позиције класе у хијерархији зависи и очекивани број подкласа. |
| *Повезаност објеката – CBO (енгл. Coupling Between Object Classes)* | Представља меру повезаности између објеката (нпр. метода првог објекта користи методе другог објекта). Висока вредност ове метрике није пожељна. Претерана повезаност између класа нарушава модуларни дизајн и отежава поновну употребу. Већа повезаност доводи до већег утицаја измена и на друге делове дизајна и самим тим отежава и одржавање софтвера. |
| *Број одговора класе – RFC (енгл. Response for a Class)* | Представља скуп свих метода које могу бити позване као одговор на поруку објекта класе. Велика вредност ове метрике |

| Назив метрике | Опис метрике |
|---|---|
| | указује на комплексне класе које су теже за разумети, што доводи до компликованијег тестирања и одржавања саме класе, јер је већи труд и време потребно уложити за тестирање свих случајева. Ниже вредности указују на већи полиморфизам. |
| *Недостатак повезаности метода у класи – LCOM (енгл. Lack of Cohesion in Methods)* | Представља меру међусобне повезаности метода у класи – заснована је на броју неповезаних парова метода у класи које представљају независне скупове који нису повезани. Ова метрика је наишла на разне критике [136], [137], [138], [139] и развијено је неколико алтернатива: [140], [141]. Вредности се тумаче у односу на варијанту метрике која је измерена. |

*Табела 4.5 Чидамберов и Кемереров скуп метрика*

Са повећањем или смањењем неке од вредности метрика, утиче се на повећање или смањење атрибута квалитета. Приказ утицаја метрика на атрибуте квалитета према [142] дат је у табели испод. Са смањењем вредности сложености пондерисаних класа, повезаности објеката, броја одговора класе и недостатака повезаности метода у класи, смањују се време и труд потребни за тестирање и развој и повећавају степени разумљивости, поновне употребљивости и одржавања софтвера.

| Метрика | Кретање метрике | *Време тестирања* | *Разумљивост* | *Одрживост* | *Време развоја* | *Поновно искоришћење* |
|---|---|---|---|---|---|---|
| *WMC* | ↘ | | | ↗ | ↘ | ↗ |
| *RFC* | ↘ | ↘ | | | | ↗ |
| *LCOM* | ↘ | | ↗ | ↗ | ↘ | ↗ |
| *CBO* | ↘ | ↘ | ↗ | ↗ | | ↗ |

*Табела 4.6 Утицај ОО метрика на атрибуте квалитета ( ↘- опада, ↗- расте)*

Приликом одређивања одговарајућег броја за дубину наслеђивања и броја подкласа потребно је направити компромис. Већа вредност броја за дубину наслеђивања доводи до повећања сложености и веће поновне употребљивости.

### 4.3.3. Атрибути квалитета софтвера

За мерење квалитета софтвера, изабрана су следећи атрибути из ИСО стандарда: сложеност одржавања софтвера, портабилност и проширивост.

### Сложеност одржавања софтвера

Сложеност одржавања софтвера (енгл. *maintainability*), према ИСО стандарду представља степен ефикасности и ефектности до кога се може мењати софтвер како би се побољшао, поправио, прилагодио променама у захтевима или окружењу. Ова особина укључује модуларност, поновно коришћење (енгл. *reusability*) – степен до кога се може користити у више система, могућност анализирања (енгл. *analyzability*) – колико једноставно се могу прикупити конкретне метрике о самом софтверу, променљивост (енгл. *modifiability*) – степен до кога се софтвер може мењати без увођења дефеката или смањења постојећег квалитета софтвера и способност тестирања (енгл. *testability*) – колико једноставно се софтвер може тестирати (укључујући тестирање и од стране оних који развијају софтвер).

Сложеност одржавања софтвера се може израчунати преко степена сложености одржавања софтвера. Постоји више варијација рачунања степена сложености софтвера (представљено у Додатку В.). Већа вредност указује на бољи степен одржавања софтвера, а границе прихватљивости зависе од саме формуле која се користи за рачунање.

### Портабилност

Портабилност (енгл. *portability*) представља степен до кога се софтвер или његов део може пребацити на други хардвер, софтвер, и/или друго оперативно или употребно окружење.

У овом раду бројчана вредност портабилности се рачуна према формули коју су предложили аутори у [143] – израчунава се на основу својих податрибута: модуларности, програмског језика, сложености и процене системске архитектуре. Резултат атрибута и податрибута се вреднује на скали од 1 до 5 (1 – *веома добро*, 2 – *добро*, 3 – *прихватљиво*, 4 – *потребна побољшања*, 5 - *лоше*).

### Проширивост

Проширивост (енгл. *extendability*) представља ефикасност укључивања нових делова функционалности у постојећи средњи слој. То је мера капацитета додавања нових елемената или особине постојећој структури софтвера.

За израчунавање проширивости, користи се исти принцип као и за портабилност - предложен од стране аутора [143]. Проширивост се дефинише као зависност од следећих атрибута: модуларност, сложеност и процена системске архитектуре.

## 4.4. КОМПАТИБИЛНОСТ СА НЕФУНКЦИОНАЛНИМ ЗАХТЕВИМА

За одређене нефункционалне захтеве као што су безбедност и сигурност, потребно је дати кратку дискусију на концептуалном нивоу, да предложени средњи слој није у сукобу са постојећим концептима који омогућавају достизање безбедносних и сигурносних захтева у оквиру целог система - изложити примере на које начине би се предложени средњи слој уклопио са постојећим концептима за ова два аспекта.

## 4.5. ПОРЕЂЕЊЕ СА ДРУГИМ РЕШЕЊИМА

У Поглављу 2. приказана су комерцијална и академска решења средњих слојева која адресирају у потпуности и/или делимично исте или сличне захтеве као и овај рад. Узевши у обзир да за већину решења не постоје детаљни подаци о самој реализацији, апстракцијама, перформансама, поређење са другим решењима ће бити одрађено на нивоу целовитости које пружа дато решење. Целовитост се дефинише као скуп функционалности које дато решење пружа и као скуп аспеката којима се бави и које задовољава на неки од начина.

Решења ће бити евалуирана по разним критеријумима. Табела 4.7 даје изабране критеријуме и могуће опције за сваки од критеријума

| Критеријум | Опције |
|---|---|
| *К1 – Хардверска платформа на којој је решење засновано* | **д** : решење је засновано на наменској платформи (из области потрошачке електронике или аутомобилске индустрије)<br>**с** : решење постоји у форми концепта, не помиње се конкретна платформа, користи се персонални рачунар<br>- : није подржано/нема информација |
| *К2.1 – Заснованост решења на хетерогеним чиповима* | **д** : решење је засновано на хетерогеном чипу/чиповима<br>**с** : вишејезгарни чипови, али не садржи више врста процесорских јединица<br>- : не користи хетерогене чипове/нема информација |
| *К2.2 – Да ли решење обезбеђује рад над више различитих* | **д** : да, обезбеђује |

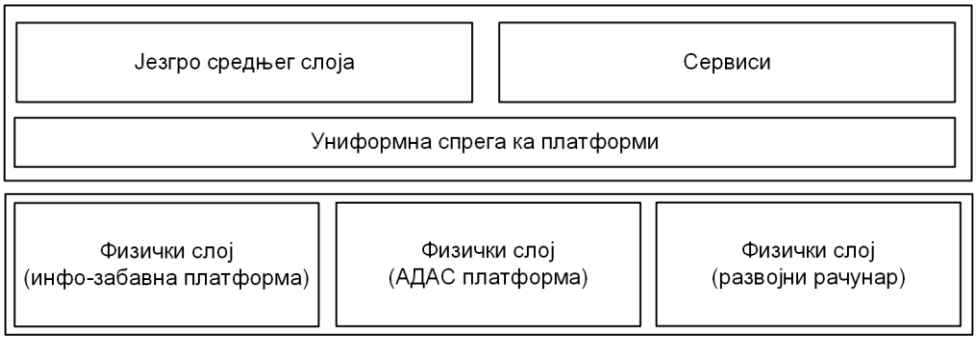
| Критеријум | Опције |
|---|---|
| *процесорских језгара унутар чипа?* | **с**: решење користи више језгара, али то корисник не види<br>- : не обезбеђује/нема информација |
| *К2.3 – Да ли решење обезбеђује апстракције ка апликацијама за рад са хетерогеним чипом?* | **д** : да, обезбеђује<br>**с**: специфично решење, делимично доступно<br>- : не обезбеђује/нема информација |
| *К2.4 – Да ли решење обезбеђује апстракције ка хардверској платформи – тако да је портабилно на друге хетерогене чипове?* | **д** : да, обезбеђује<br>**с**: специфично решење, делимично доступно<br>- : не обезбеђује/нема информација |
| *К2.5 – Да ли решење пружа могућност апстраховања извршавања апликација над више од једним хетерогеним чипом?* | **д** : да, обезбеђује<br>**с**: специфично решење, делимично доступно<br>- : не обезбеђује/нема информација |
| *К3 – Да ли решење обезбеђује рад са сензорима и актуаторима?* | **д** : да, обезбеђује апстракције, са могућношћу проширивања новим сензорима и актуаторима<br>**с**: специфично решење, делимично доступно – само одређени сензори доступни, без могућности проширивања, или сензори доступни директно апликацијама без апстракција<br>- : не обезбеђује/нема информација |
| *К4 – Да ли решење обезбеђује могућност рада са магистралама?* | **д** : да, обезбеђује апстракције, са могућношћу проширивања новим магистралама<br>**с**: специфично решење, делимично доступно – само одређене магистрале доступне, без могућности проширивања, или магистрале доступне директно апликацијама без апстракција<br>- : не обезбеђује/нема информација |
| *К5 – Да ли је решење независно од одређеног оперативног система?* | **д** : да, независно је<br>**с**: специфично решење, делимично доступно<br>- : није независно/нема информација |
| *К6 – У којој мери је решење независно од оперативног система?* | **д** : решење обезбеђује апстракције ка оперативном систему, омогућавајући раздвајање апликација од оперативног система.<br>**с** : решење се извршава на обе врсте оперативних система, али не обезбеђује апстракције ка истом.<br>**х** : решење је предвиђено само за оперативне системе високог нивоа<br>**р** : решење је предвиђено само за оперативне системе за рад у реалном времену<br>- : није независно/нема информација |
| *К7 – Да ли се решење извршава у реалном времену?* | **д** : да, извршава<br>**с**: делимично извршава (један део), концептуално предвиђено за извршавање у реалном времену<br>- : не обезбеђује/нема информација |
| *К8 – Да ли решење обезбеђује механизам за руковање апликацијама (животни циклус, распоређивање, итд.)?* | **д**: решење обезбеђује апстракције и механизам за развој више апликација и руковање њиховим животним циклусом<br>**с**: решење је специфично за одређену апликацију и/или групу апликација и не пружа потребне спреге за развој других апликација, или решење користи друго решење за руковање апликацијама (нпр. адаптивни Аутосар)<br>- : не обезбеђује/нема информација |

| Критеријум | Опције |
|---|---|
| *К9 - Да ли решење омогућава коришћење постојећих широко распрострањених спрега и библиотека?* | **д** : да, омогућава<br>**с** : делимично омогућава<br>- : не омогућава/нема информација |
| *К10 - Да ли решење обезбеђује могућност повезивања са другим доменима у возилу?* | **д** : да, обезбеђује<br>**с** : специфично решење, делимично доступно<br>- : не обезбеђује/нема информација |
| *К11 - Да ли решење обезбеђује подршку за помоћне функције при развоју и тестирању (нпр. симулатори, логовање)?* | **д** : да, обезбеђује<br>**с** : специфично решење, делимично доступно уз коришћење других библиотека<br>- : не обезбеђује/нема информација |
| *К12 - Да ли решење адресира безбедносне аспекте?* | **д** : да, адресира, реализовано у склопу решења<br>**с** : адресира на концептуалном нивоу<br>- : не адресира/нема информација |
| *К13 - Да ли решење адресира сигурносне аспекте?* | **д** : да, адресира, реализовано у склопу решења<br>**с** : адресира на концептуалном нивоу<br>- : не адресира/нема информација |
| *К14 - Да ли решење адресира алате за рад са платформом?* | **д** : да, адресира, реализовано у склопу решења<br>**с**: адресира на концептуалном нивоу<br>- : не адресира/нема информација |

*Табела 4.7 Критеријуми целовитости решења*

# 5. ПРИМЕР РЕАЛИЗАЦИЈЕ

У овом поглављу дат је пример реализације средњег слоја: AMB повезаног дела и реализације слоја за апстракцију платформе - на три платформе: инфо-забавна платформа, АДАС платформа, и персонални рачунар за помоћ при развоју (Слика 5.1). За сваку од платформи дати су преглед платформе и њеног развојног окружења, опис реализације слоја за апстракцију платформе као и примери апликација.




*Слика 5.1 Приказ реализованих модула*

AMB повезани слој је реализован у *C++* програмском језику кроз реализацију основног скупа функционалности који подразумевају језгро повезаног слоја, спрегу ка платформи и одређени број сервиса. За реализацију тестова (појединачни (енгл. *unit*), интеграциони) коришћен је *GoogleTest* радни оквир [144]. На исти начин вршени су и тестови над слојем апстракције платформе. За појединачни тест испод је дат приказ једноставног теста за модул *MemoryPool* , тј. захтев за бафером са неисправним аргументима.

Додатак Б. приказује репрезентативни део тестова корисничких случајева који су реализовани стварањем контролисаних пајплајнова.

```
/* Захтев за бафером неисправне величине */
TEST_F(Test_MemoryPool, requestBufferWrongSizeParam)
{
    int32_t size = -1;
    amv::Error error;
    amv::Buffer* buffer;

    amv::MemoryPool* memoryPool = amv::MemoryPool::getInstance();

    buffer = new amv::Buffer(size, amv::E_MEMORY_DDR_NON_CACHED_SR0,
      amv::E_READ_WRITE);

    error = memoryPool->requestBuffer(buffer);
    EXPECT_EQ(amv::E_ERROR_WRONG_ARGUMENT, error);
}
```

## 5.1. СЛОЈ АПСТРАКЦИЈЕ ПЛАТФОРМЕ *S820A*

Једна од платформи за коју је реализован слој апстракције је *S820a Qualcomm Snapdragon* развојна платформа [23] .

### 5.1.1. Преглед платформе и развојног окружења

*S820a* (Слика 5.2) је хетерогена развојна платформа у аутомобилској индустрији усмерена на умреженост, због чега је нашла већу примену у инфо-забавним системима. Заснива се на *S820a* чипу који поседује следеће процесоре:

- *CPU* – централна процесорска јединица са четири језгра – *Kryo*
- *DSP* – процесорска јединица за дигиталну обраду сигнала - *Hexagon 680 DSP*
- *GPU* – процесорска јединица за графичку обраду - *Adreno 530 GPU*
- *ISP* – две процесорске јединице за обраду слике за сензора - *Spectra ISP*.

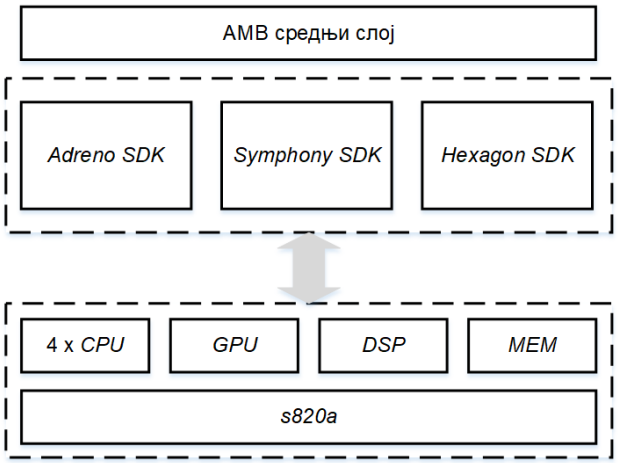
*Слика 5.2 S820a Qualcomm Snapdragon развојна платформа [23]*

Платформа има могућност прикључивања до осам камера, као и подршку за 4К приказ слике/видеа. Означена је као мултимедијална платформа са разним опцијама повезивања са локалним уређајима у возилу, са облаком и преко *WiFi* везе користећи *4G X12 LTE* модем.

**Развојно окружење платформе**

*Snapdragon* фамилија чипова даје развојна окружења која пружају подршку при коришћењу хетерогеног чипа у смислу апстракција над таквим окружењем и контроле како би се постигле жељене перформансе (Слика 5.3). За рад са *DSP* или *GPU* језгрима доступна су развојна окружења *Hexagon SDK* и *Adreno SDK*. Ова два развојна окружења пружају низ алата омогућавајући прецизно руковање подацима и рачунарским ресурсима.

За управљање целим системом доступан је *Symphony System Manager SDK* (у даљем тексту *Symphony SDK*). *Symphony SDK* пружа подршку при коришћењу хетерогеног чипа (вишејезгарни *CPU, GPU* и *DSP*) у смислу расподеле задатака и постизања мање потрошње енергије.




*Слика 5.3 Софтверско развојно окружење платформе*

Андроид оперативни систем налази се на *CPU* језгрима, и пружа могућност коришћења овог развојног окружења преко корисничке библиотеке. Кориснику је доступан *C++* АПИ за паралелизацију и доделу задатака, за контролу потрошње енергије у циљу њеног смањења, као и распоређивач на *CPU, GPU* и *DSP* језгра.

### 5.1.2. Реализација слоја за апстракцију

У наставку је дат опис реализације слоја за апстракцију по логичким целина, користећи развојна окружења и библиотеке доступне на циљној платформи.

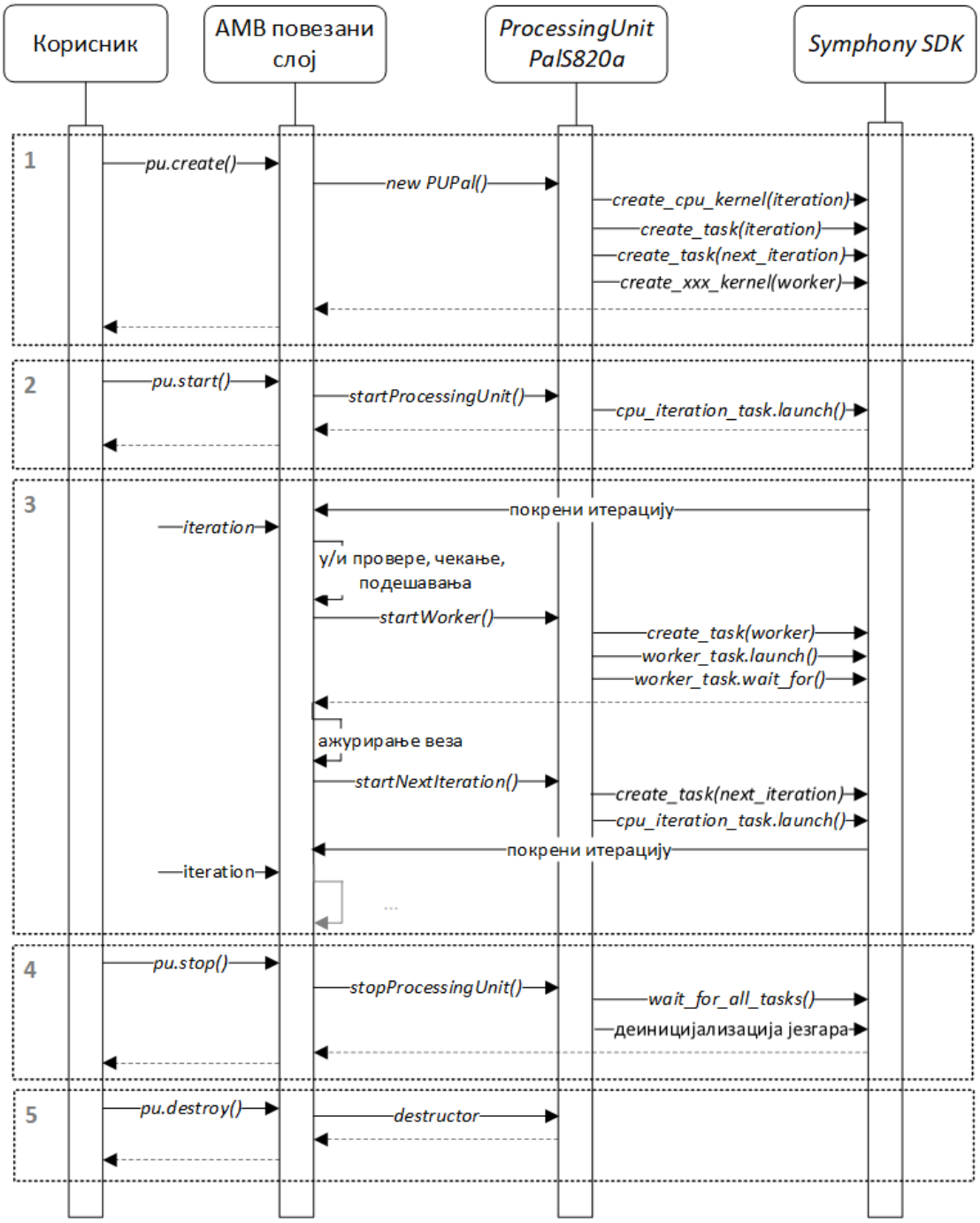
**Јединице за обраду и расподела на језгра**

Јединица за обраду је реализована користећи *Symphony SDK* АПИ. *Symphony Tasks* - задаци представљају начине специфицирања делова обраде који се могу мапирати на различита процесорска језгра. Дефинисање и додељивање кернела датом задатку уједно и одређује и тип језгра на ком ће се извршавати. *Symphony SDK* даје подршку за три врсте кернела: *CPU*, *DSP* и *GPU*. Развојно окружење пружа могућност покретања и синхронизације задатака индивидуално или као групу задатака, такође пружа могућност дефинисања зависности између задатака.

Слика 5.4 приказује логички дијаграм коришћења задатака и кернела из *Symphony SDK*-а за реализацију јединице обраде. AMB итерација јединице обраде је реализована као задатак чији је кернел *iteration* метода. На сличан начин реализована је и сама обрада, покретањем задатка чији је кернел *worker* метода. У зависности од језгра на коме треба да се извршава обрада, зависи и ког типа ће се кернел креирати:

- за креирање *CPU* кернела користи се *symphony::create_cpu_kernel*, чија реализација по платформској спецификацији између осталог може бити и *C++11* функција,
- за креирање *GPU* кернела користи се *symphony::create_gpu_kernel*, две варијанте су подржане, *OpenCL C* и *OpenGL ES compute shaders*,
- за креирање *DSP* кернела користи се *symphony::create_hexagon_kernel*, који подржава *DSP* компатибилне *C* функције.

Задаци могу имати вишеструке претходнике и следбенике у графу. Задатак је спреман за извршавање ако и само ако су се сви његови претходници завршили успешно.




*Слика 5.4 Дијаграм интеракције јединице обраде ка платформи*

**Руковање меморијом и размена података - *MemoryPool, Buffer, Queue***

*Symphony* развојно окружење пружа *Symphony* бафер који је доступан на *CPU*, *GPU*, и *DSP* језгрима. Представљан је као континуални део меморије за одговарајући кориснички тип података. У зависности од типа меморије одређује се где се меморија заузима. *Symphony* радно окружење транспарентно управља смештањем и премештањем података између специјализованих меморијских региона на самом чипу. На пример, *ION* меморија се користи за оптималну размену података између *CPU*-а и *DSP*-а. Идеја је да АМВ средњи слој користи већ уграђену логику у само развојно окружење, подешавајући само зависности између задатака, како би се на најефикаснији начин размењивали подаци. У тренутној реализацији због враћања контекста на *CPU,* након сваког извршавања методе *worker*, и подаци се имплицитно достављају на *CPU*. План за будући рад за унапређење овог слоја за апстракцију је да се ова додатна копирања избегну (или подешавањем платформског окружења или експлицитним заузимањем одговарајуће меморије).

За реализацију реда (енгл. *Queue*), користи се платформски ФИФО ред без закључавања (*symphony::lfqueue<Buffer*>*), због чега је синхронизациони механизам реализован у склопу слоја за апстракцију користећи платформске механизме узајамно искључивог закључавања и варијабли са условним приступом. Додавање новог елемента и преузимање постојећег врше се преко *push* и *pop* метода платформског реда уз одговарајуће поменуте механизме синхронизације. Ослобађање искоришћених бафера у реду се врши преузимањем једног по једног бафера, затим се проверава да ли је бафер искоришћен и спреман за брисање (ако јесте, меморија се ослобађа, у супротном, бафер се враћа назад у ред).

**Системске функционалности**

У табели испод је дат преглед реализације системских функционалности кроз библиотеке или спреге које се користе за реализацију. То су углавном Андроид/Линукс спреге, док се за синхронизацију користе и механизми из *Symphony SDK* радног окружења као платформска реализација.

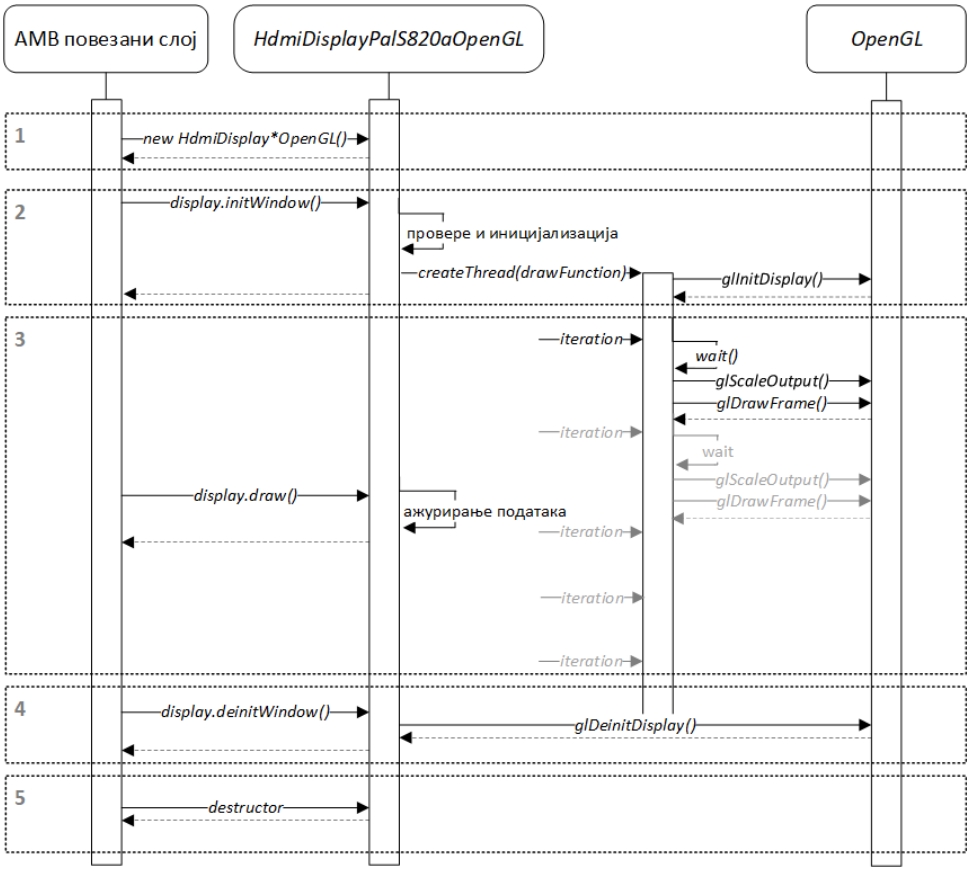
| **Функционалност** | **Библиотека/Спрега** |
|---|---|
| *Рад са нитима* | *pthread* спрега |
| *Рад са временом* | *C* стандардна библиотека *(time)*<br>*posix* стандардна спрега *(unistd)* |

| | |
|---|---|
| *(заустављање извршења у интервалу израженом у микросекундама, добављање и постављање тренутног времена и временске зоне, итд.).* | |
| *Рад са датотекама* | *C* стандардна библиотека (*stdio*) |
| *Логовање*<br>*(ниво логовања по модулу, системски, постављање датотеке за исписе, реализација формата исписа)* | *C* стандардна библиотека (*file*, *stdio*, *stdarg*, *string*)<br>Стандардни контејнери (*map*) |
| *Синхронизација*<br>*(узајамно искључиво закључавање / варијабла са условним приступом)* | Спрега платформског развојног окружења: *SymphonySDK* |

*Табела 5.1 Приказ реализације системских функционалности*

### AMB сервиси

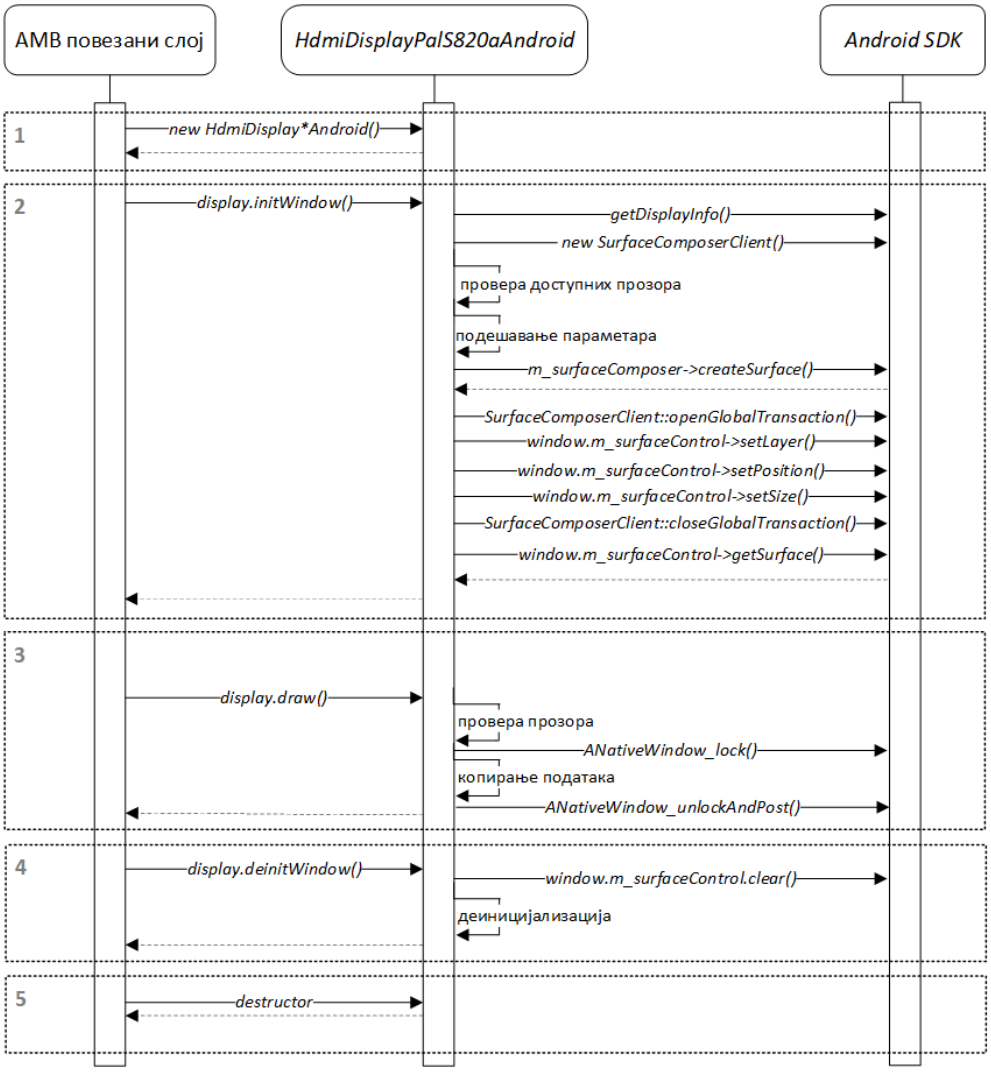
Реализован је одређен скуп AMB сервиса, са фокусом на улазне и излазне сервисе, који укључују камере, приказ на екран, ултразвучне сензоре и сл. У наставку је дат приказ примера двоструке реализације сервиса за приказ на екран на овој платформи.




*Слика 5.5 Приказ на екран OpenGL спрега*

Слика 5.5 приказује реализацију користећи *OpenGL* спрегу. Оно што је специфично за ову реализацију је креирање позадинске нити која извршава *OpenGL* операције из истог контекста, слика на екрану се освежава периодично, док се бафер који се исцртава освежава на позив *draw* методе.

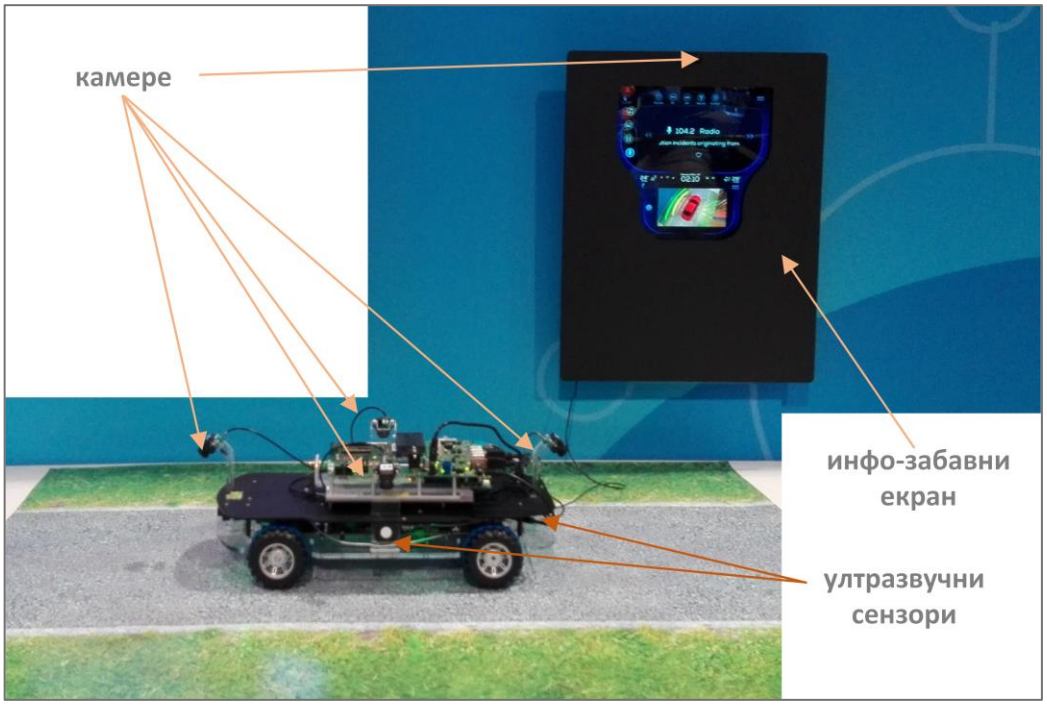
Слика 5.6 приказује реализацију приказа на екран коришћењем Андроид спрега. За разлику од претходне реализације, освежавање екрана се диктира динамиком позива *draw* методе.




*Слика 5.6 Приказ на екран Андроид спрега*

### 5.1.3. Примери апликација

Две корисничке апликације су реализоване и демонстриране користећи AMB средњи слој и демонстратор – прототип возила са циљном хетерогеном платформом, камерама, ултразвучним сензорима и инфо-забавном јединицом из ( [145]). Поред демонстрације исправне функционалности AMB средњег слоја, демонстрира се и могућност рада у реалном времену.

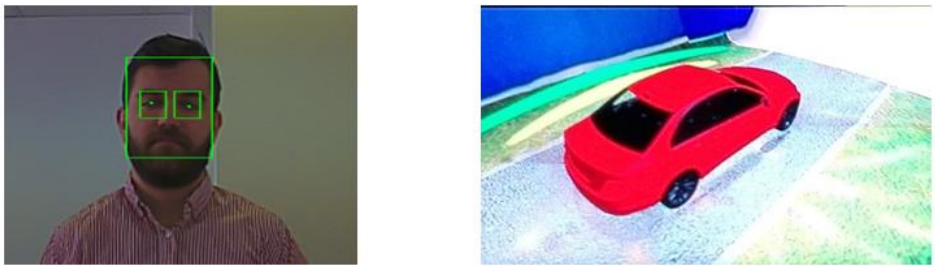



*Слика 5.7 Прототип возила и инфо-забавни систем са интегрисаним приказом апликација*

Следеће две апликације су реализоване преко развијеног средњег слоја:

- праћење возача (енгл. *driver monitoring*) – користи се алгоритам из [146], врши се детекција лица и очију на слици, као и израчунавање индикатора будности возача.
- преглед окружења возила из птичије перспективе (енгл. *bird's eye view*) – користи се алгоритам из [147]. За демонстрацију се користе четири широкоугаоне камере и четири ултразвучна сензора за проналажење препрека. Калибрација се врши под претпоставком да је позиција камера фиксна. У односу на удаљеност нађених препрека исцртавају се линије различитих боја као три нивоа упозорења у 3Д окружењу возила.

Приказ излаза апликација на ХДМИ екран је дат на слици испод.



*Слика 5.8 Приказ резултата апликација на екран*

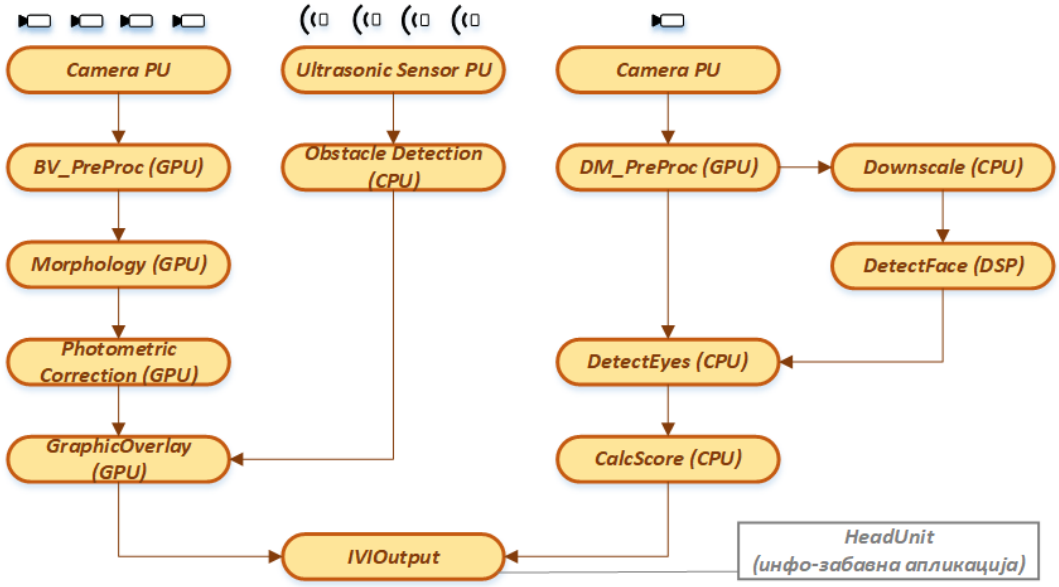
Слика 5.9 приказује АМВ пајплајн за овакав систем. Поред улазних и излазних блокова у систему, присутни су и обрадни блокови на различитим језгрима. За праћење возача то су следеће јединице обраде:

- *DM_PreProc* – предобрада слике која долази са камере: исецање и конверзија формата боја. Ова јединица обраде се извршава на *GPU* језгру.
- *Downscale* – смањење димензија и величине слике. Представља предобраду за детекцију лица и извршава се на *CPU* језгру.
- *DetectFace* – јединица обраде за препознавање лица. Користи се *Viola James* алгоритам и извршава се на *DSP* језгру.
- *DetectEyes* – јединица за препознавање очију и ириса. Након што се препозна центар очију, врши се препознавање да ли су отворене или затворене. Извршава се на *CPU* језгру
- *CalcScore* – на основу процента затворености очију у одређеном временском интервалу врши се ажурирање скора будности возача. Ова јединица обраде се извршава на *CPU* језгру.

Преглед окружења возила из птичије перспективе користи следеће јединице обраде:

- *ObstacleDetection* – препознавање присуства препрека у односу на податке које дају сензори. Извршава се на *CPU* језгру.
- *BV_PreProc* – предобрада слике која долази са камере: конверзија формата боја. Извршава се на *GPU* језгру.

- *Morphology* – геометријско исправљање слике са широкоугаоних камера, њихово спајање и стављање у одређену перспективу. За обраду користи *OpenGL* и извршава се на *GPU* језгру.
- *PhotometricCorrection* – изједначавање осветљења између слика са камера. За обраду користи *OpenCL* спреге и извршава се на *GPU* језгру.
- *GraphicOverlay* – додавање у приказ 3Д модела возила и исцртавање препрека на одговарајућој удаљености, на основу резултата препознавања истих. Извршава се на *GPU* језгру.

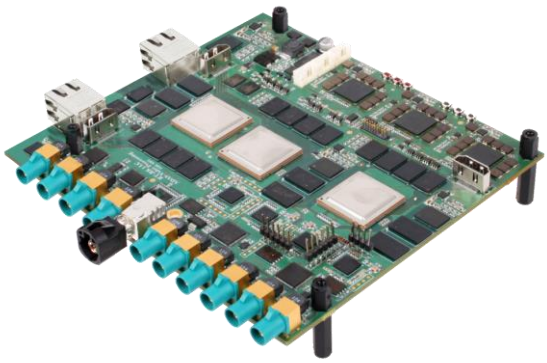



*Слика 5.9 AMB најплајн система: праћење возача и преглед окружења возила*

Овај прототип верификује функционалност интеграције са улазима (камере, ултразвучни сензори) и излазима (инфо-забавни систем, ХДМИ излаз на екран). Исто тако приказује могућност AMB средњег слоја да пружи одговарајуће перформансе у реалном времену (нпр. за преглед окружења возила из птичије перспективе – 30 слика у секунди).

## 5.2. СЛОЈ АПСТРАКЦИЈЕ ПЛАТФОРМЕ *ALPHA AMV*

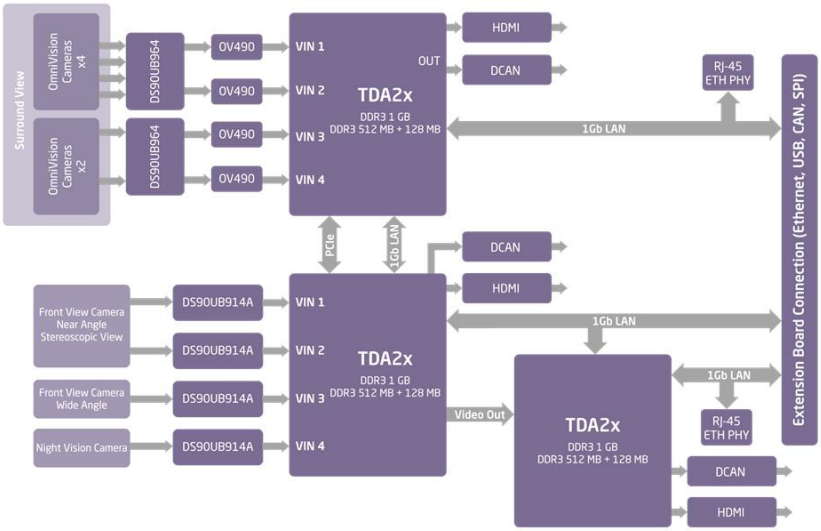
Друга платформа за коју је реализован слој апстракције је *Alpha AMV* развојна платформа [148].



*Слика 5.10 Alpha AMV развојна платформа [148]*

### 5.2.1. Преглед платформе и развојног окружења

*Alpha AMV* платформа је развојна платформа за аутомобилску индустрију заснована на три *TDA2x* вишејезгарна чипа. Оваква архитектура платформе пружа могућност истовременог извршавања већег броја алгоритама. На платформу се може повезати до десет камера. Чипови су међусобно повезани брзим магистралама како би се омогућила брза дистрибуција велике количине података између чипова. Блок дијаграм платформе је представљен на Слика 5.11.




*Слика 5.11 Alpha AMV развојна платформа – хардверски блок дијаграм [148]*

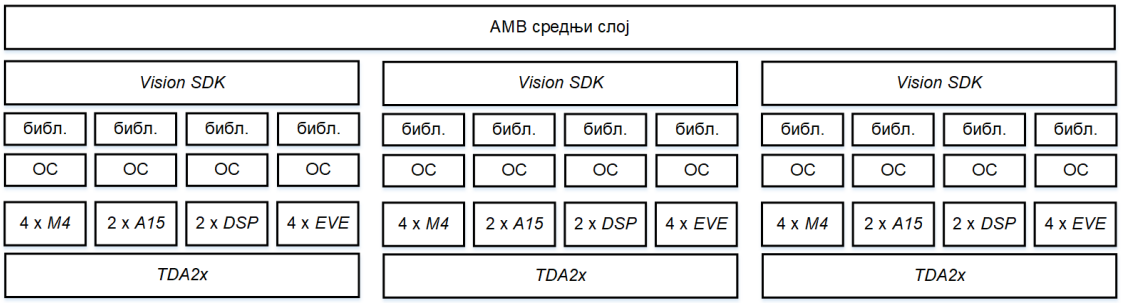
Са слике видимо да су све камере повезане на прва два чипа. Прва два чипа су међусобно повезана *PCIe* магистралом, други и трећи видео магистралом, а сви чипови су увезани етернет магистралом. Сва три чипа су идентична и представљају вишејезгарне чипове који се фокусирају на комбинацију перформанси, мале потрошње и рачунарску визију у системима за помоћ возачу. *TDA2x* чипови се састоје од следећих процесорских јединица:

- 2 x *ARM Cortex-A15* – процесорска јединица опште намене, која се поред обраде такође користи и за разне управљачке задатке,
- 4 x *ARM Cortex-M4* – процесорске јединице опште намене, које се првенствено користе за ефикасно управљање и обраду слике са камера,
- 2 x *DSP C66x* - процесорске јединице за дигиталну обраду сигнала,
- 4 x *EVE* (енгл. *Embedded Vision Engine*) – потпуно програмабилна, наменска архитектура векторског процесора, која има за циљ решавање изазовних захтева рачунарске визије. [149]
- *IVA HD Coprocessor* – хардверски акцелератор за обраду слике и видео кодека (омогућава видео кодовање и декодовање *FullHD* резолуције)
- *GPU SGX544* – процесорска јединица за графичку обраду сигнала. Приступ овој графичкој јединици је омогућен са *A15* језгра под Линукс оперативним језгром.

**Развојно окружење платформе**

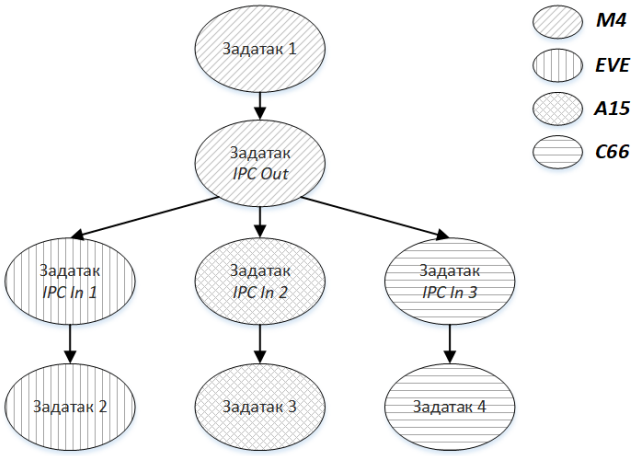
ТДА фамилија чипова пружа развојно окружење *VisionSDK* које омогућава приступ разним процесорским јединицама. Развојно окружење омогућава кориснику да креира АДАС апликације засноване на протоку података, у оквиру овог чипа. Даје примере коришћења различитих процесорских јединица и хардверских акцелератора, као и специфичних подсистема у самом чипу.

*VisionSDK* се ослања на оперативни систем као и на скуп специфичних библиотека за дато језгро (нпр. функције за анализу и обраду слике, контрола система за прихват и приказ слике). На *A15* језгру су могуће две варијанте оперативног система: Линукс и РТОС, док се на свим осталим језгрима извршава РТОС оперативни систем (Слика 5.12)




*Слика 5.12 Софтверско развојно окружење платформе*

Користећи *VisionSDK* развојно окружење креира се увезани ток задатака (Слика 5.13) који се извршавају на различитим процесорима. Међупроцесорска комуникација унутар чипа се реализује посебним *IPC* задацима.




*Слика 5.13 Пример увезаног тока задатака*

### 5.2.2. Реализација слоја за апстракцију

За разлику од претходне платформе где је AMB језгро трчало на централној процесорској јединици, а извршење конкретних обрадних задатака се пребацивало на друга језгра, на *Alpha* платформи концепт је другачији - целокупна AMB обрадна јединица се извршава на додељеном језгру. У ове сврхе реализован је одређени апстракциони слој за све четири врсте процесорских језгара: *M4*, *A15*, *EVE*, *C66*. Приликом реализације коришћене су две варијанте система:

- RTOC оперативни систем на свим језгрима,
- Линукс оперативни систем на *A15* језгру, а на осталим језгрима RTOC оперативни систем.

РТОС, под називом *Sys/Bios,* који се налази на овој платформи је систем који се састоји од једног процеса. За разлику од претходне платформе где АМВ апликација представља посебан процес, овде се компајлира као библиотека, чије функције за покретање и заустављање пајплајна се прозивају из главне РТОС апликације тј. контекста.

Табела 5.2 приказује реализоване целине над сваким од типова језгара, у зависности од тога шта је директно подржано на одређеном језгру. Рад са датотекама је подржан једино на *M4* језгру под РТОС оперативним системом. АМВ сервиси који се односе на прихват слика са камере су означени као подржани на *M4* језгру – означава да се камере физички контролишу са тог језгра, а не значи да је исти садржај недоступан на другим језгрима.

| **Целина** | ***M4*** | ***A15* (*Линукс*)** | ***A15*** **(РТОС)** | ***C66*** **(РТОС)** | ***EVE*** **(РТОС)** |
|---|---|---|---|---|---|
| *Јединица за обраду* | + | + | + | + | + |
| *Руковање меморијом и размена података* | + | + | + | + | + |
| *Систем (рад са нитима)* | + | + | + | + | + |
| *Систем (рад са временом)* | + | + | + | + | + |
| *Систем (рад са датотекама)* | + | + | - | - | - |
| *Систем (логовање)* | + | + | + | + | + |
| *Систем (синхронизација)* | + | + | + | + | + |
| *АМВ сервиси (прихват слика са камера и приказ)* | + | - | - | - | - |
| *АМВ сервиси (обраде)* | + | + | + | + | + |

*Табела 5.2 Реализација слоја за апстракцију по језгрима*

**Јединице за обраду и расподела на језгра**

Јединица за обраду је реализована користећи *Vision SDK* програмску спрегу. *Vision SDK* задаци представљају начине специфицирања делова обраде који се могу мапирати на различита процесорска језгра. Дефинисањем и иницијализацијом задатака одређује се и тип и специфично језгро на ком ће се извршавати обрада.

АМВ јединица обраде реализована је ослањајући се на *VisionSDK* задатке. *VisionSDK* задатак представља нит која се врти на одређеном језгру и која периодично или на одређени догађај извршава дефинисану обраду. АМВ јединица обраде у свом слоју апстракције реализује генерички задатак који се инстанцира и подешава у зависности од параметара специфичне јединице обраде.

Генерички AMB задатак као и други задаци се састоји од нити која је упарена са сандучетом за поруке. Сандуче за поруке омогућава корисничкој апликацији (у овом случају AMB слоју за апстракцију), као и другим задацима да комуницирају са овим задатком. Поред овог сандучета, AMB генерички задатак такође реализује и механизам који омогућава другим задацима да директно размењују видео или друге податке, без уплитања корисничке апликације.

Платформски *SDK* омогућава да сваки задатак реализован на овакав начин (па и AMB задатак) може комуницирати са било којим другим постојећим задатком (на истој или другој процесорској јединици), као и бити инстанциран више пута на једном или више истих и/или различитих процесорских јединица. Управљање задацима је апстраховано платформском спрегом, без обзира да ли се задатак налази на истој процесорској јединици као и управљачки део апликације или на другој процесорској јединици.

**Руковање меморијом и размена података**

За реализацију руковања меморијом и размену података користи се системска спрега из платформског радног окружења. Спрега је доступна на свим језгрима и олакшава руковање меморијом - омогућава заузимање и ослобађање разних типова меморије. AMB средњи слој за реализацију меморијског базена користи ову спрегу. Логика којом се служи за заузимање меморије (и избор типа меморије) је иста она коју препоручује само развојно окружење.

*TDA2x* чип има више ЕДМА (енгл. *Enhanced Direct Memory Access*) контролера:

- Системски ЕДМА контролер је доступан са *M4*, *A15* и *C66* процесорског језгра и подржава оба начина рада: са прекидима и са прозивањем.
- Локални ЕДМАх контролер за *C66/EVE* језгра. Свако језгро има свој локални контролер. Користи се обично за специфичне потребе алгоритама.

Додатно, *C66* и *EVE* језгра имају интерну меморију која се може користити за смештање података самих алгоритама и употребљива је само са ових језгара. Поред ове меморије постоји и *OCMC* (енгл. *On-Chip Memory Controller*) меморија која се може користити са било ког процесорског језгра.

За размену бафера између AMB обрадних јединица користи се ДДР дељени меморијски регион. За управљачке поруке, мање податке који се не шаљу често, користи се некеширана меморија, док се за размену већих количина података као што су слике са камера користи кеширани меморијски регион.

Функционалност *Queue* класе је реализована користећи системску спрегу за размену података између две нити (енгл. *UTILS_BUF_API System Buffer Exchange API*). Интерна платформска реализација овог механизма за размену садржи два реда, један у који се смештају празни, а други у који се смештају пуни бафери података. Оба реда су реализовани као редови фиксне величине која се не може мењати у току извршавања. Раде на следећем принципу:

- Произвођач потражује празан бафер, додељује му одговарајуће податке, и захтева смештање истог у ред са пуним баферима.
- Потрошач потражује пуни бафер, конзумира податке и захтева смештање сада већ празног бафера у ред са празним баферима.

Тип података елемента платформског реда је платформски системски бафер. У слоју за апстракцију врши се конверзија из AMB бафера у платформски бафер (додела одговарајућих показивача, без додатног копирања података). При стварању *Queue*-а ствара се и описани системски бафер механизам са фиксним бројем елемената. Додавање у *Queue* се мапира на захтев за додавање бафера у пуни платформски ред.

**Системске функционалности**

У табели испод је дат преглед реализације системских функционалности кроз библиотеке или спреге које се користе за реализацију. То су углавном RTOC и/или Линукс спреге.

| **Функционалност** | **Библиотека/Спрега** |
|---|---|
| *Рад са нитима* | RTOC: спрега за рад са RTOC тасковима<br>Линукс: *pthread* спрега |
| *Рад са временом*<br>*(заустављање извршења у интервалу израженом у микросекундама, добављање и постављање тренутног времена и временске зоне, итд.)* | RTOC: RTOC наменска спрега са примитивним функцијама слања нити у позадину, као и коришћење *Clock* модула.<br>Линукс: *C* стандардна библиотека (*time.h*) |
| *Рад са датотекама* | RTOC: Рад са класичним датотекама под RTOC оперативним системом подржан је једино на *M4* језгру на овој платформи. За реализацију се |

| | |
|---|---|
| | користи РТОС библиотека која пружа приступ датотекама на сличан начин као и *Posix* спрега.<br>Линукс: *C* стандардна библиотека (*stdio.h*) |
| *Логовање*<br>*(ниво логовања по модулу, системски, постављање датотеке за исписе, формат исписа)* | РТОС, Линукс: Реализација слоја за апстракцију је иста на оба ОС-а и заснива се на *C* стандардној библиотеци. Додатно, за објединење исписа са свих језгара користи се платформска наменска спрега. |
| *Синхронизација*<br>*(узајамно искључиво закључавање / варијабла са условним приступом)* | РТОС: користе се наменски РТОС објекти искључивог приступа.<br>Линукс: *pthread* спрега |

*Табела 5.3 Преглед реализације системске функционалности*

### АМВ сервиси

На овој платформи реализован је одређен скуп АМВ сервиса, са фокусом на улазне и излазне сервисе, који укључују камере, приказ на екран, као и апстракцију три магистрале у склопу комуникације између чипова.

Најзначајнији за поменути је камера сервис који омогућава коришћење до десет камера на сваком од три чипа на овој платформи (не ограничавајући се само на једну платформу већ омогућавајући и пренос слике са камере на друге платформе преко етернет магистрале). Поред избора саме камере могућ је и избор формата слике (у случају да није исти формат као са камере, врши се конверзија формата користећи доступне хардверске блокове), броја слика у секунди и слично.

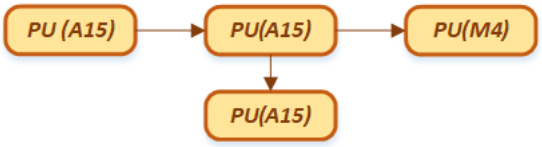
У случају да су камере директно повезане за језгро/чип на коме су захтеване, у оквиру камера сервиса врши се иницијализација одговарајућих хардверских и софтверских блокова и добављање слике. У супротном, врши се пренос података преко разних магистрала, у односу на то где се налази камера, а где су подаци захтевани. Избор магистрале врши се на следећи начин:

1. Да ли је у питању пренос између чипа 1 и чипа 2 на платформи? Ако је одговор да, магистрала је *PCIe* (под условом да магистрала подржава пренос захтеване количине података).
2. Да ли је у питању пренос података од чипа 2 ка чипу 3 на платформи? Ако је одговор да, магистрала је видео (под условом да магистрала подржава пренос захтеване количине података).
3. Ако није испуњено 1 или 2, користи се етернет магистрала, као увек подржана, јер су сва три чипа повезана гигибитном етернет везом.

**Суживот AMB слоја и платформског радног оквира**

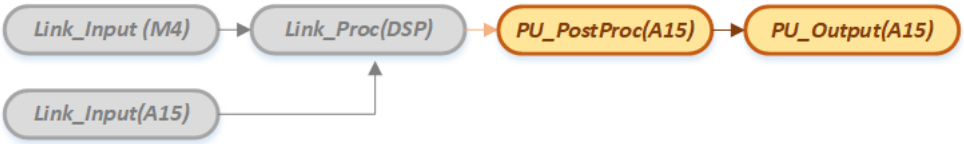
Како су AMB јединице обраде засноване на *VisionSDK* задацима, на овој платформи се пружа могућност коришћења AMB средњег слоја на два начина:

- апликација се креира користећи AMB пајплајн (регуларни начин коришћења) (Слика 5.14),




*Слика 5.14 Пример регуларног пајплајна*

- апликација се креира комбинацијом коришћења AMB пајплајна са *VisionSDK* задацима (Слика 5.15).




*Слика 5.15 Пример комбинованог пајплајна*

Предност могућности комбиновања ова два система је вишеструка:

- Омогућен је постепени прелазак са *VisionSDK* радног окружења на AMB средњи слој;
- Прелазак целе апликације на AMB средњи слој није потребан услов за коришћење предности AMB слоја. Омогућена је бржа расподела апликације на више чипова/ресурса, користећи улазне и излазне јединице из AMB слоја, при чему обраде могу остати у форми платформских задатака.

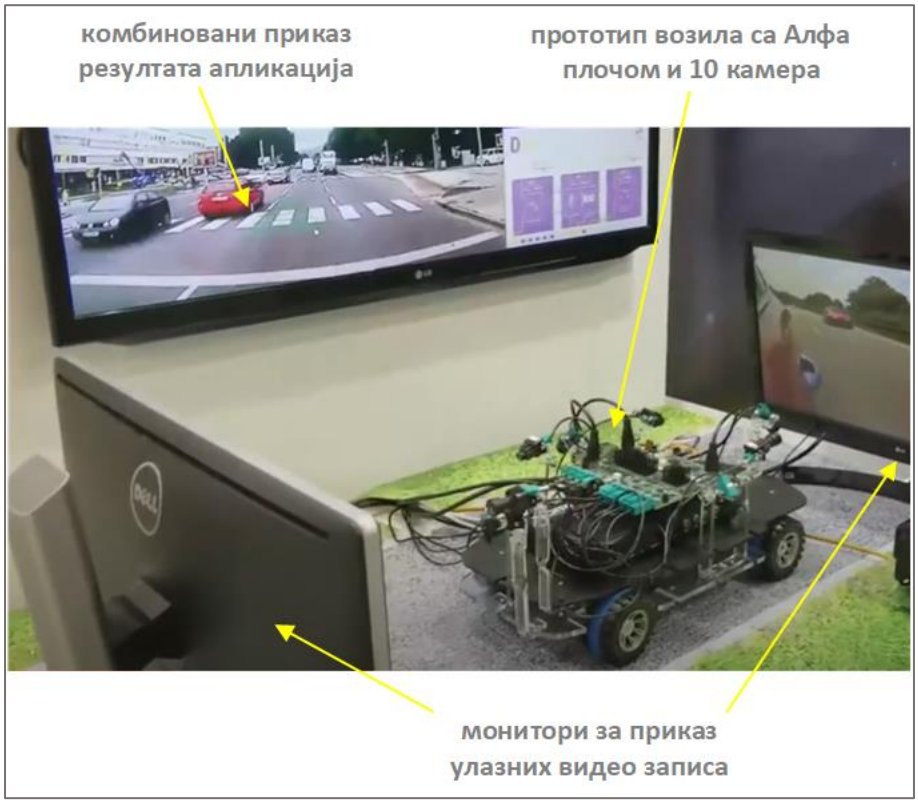
### 5.2.3. Примери апликација

Више корисничких апликација је реализовано и демонстрирано користећи AMB средњи слој и демонстратор – прототип возила са циљном хетерогеном платформом на коју је прикључено десет камера. Поред демонстрације исправне функционалности AMB средњег слоја, демонстрира се и могућност рада у реалном времену, као и размена података између више чипова.

Слика 5.16 приказује тестно окружење са прототипом возила са *Alpha* платформом. Камере су повезане и распоређене на следећи начин:

- четири широкоугаоне камере на четири бочне стране возила под углом на доле – како би се обухватило окружење возила,
- две камере уместо два ретровизора,
- једна камера за снимање возача – у овом случају монтирана на монитор,
- две камере за снимање испред возила,
- једна камера за снимање иза возила.

Одређене камере су монтиране на такав начин да снимају видео запис са монитора (нпр. ретровизори, камера која снима испред возила), како би се што боље осликао улаз у апликације у реалном времену и демонстрирао целокупни пајплајн.

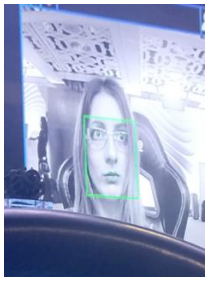


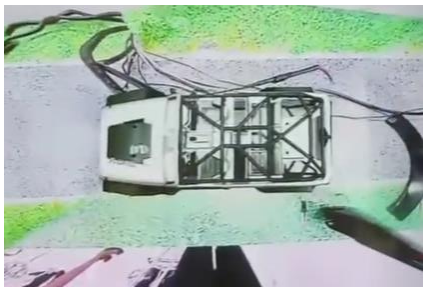

*Слика 5.16 Прототип возила са Alpha платформом*

На овој платформи реализовано је више апликација и корисничких случајева од којих ће неки бити приказани испод.

### Систем апликација за помоћ возачу

Систем апликација за помоћ возачу је реализован користећи девет камера са поменутог прототипа. Апликације које су коришћене (Табела 5.4) су или доступне као део примера платформског радног окружења, или су комерцијални алгоритми, или су развијени за сврхе демонстрације. Сам систем је због порекла алгоритама/апликација реализован комбиновањем AMB јединица обраде и платформских задатака. Платформски задаци су углавном коришћени за саму обраду и језгро алгоритма, док се AMB средњи слој користи за улазно/излазне, управљачке и комуникационе делове апликација и система.

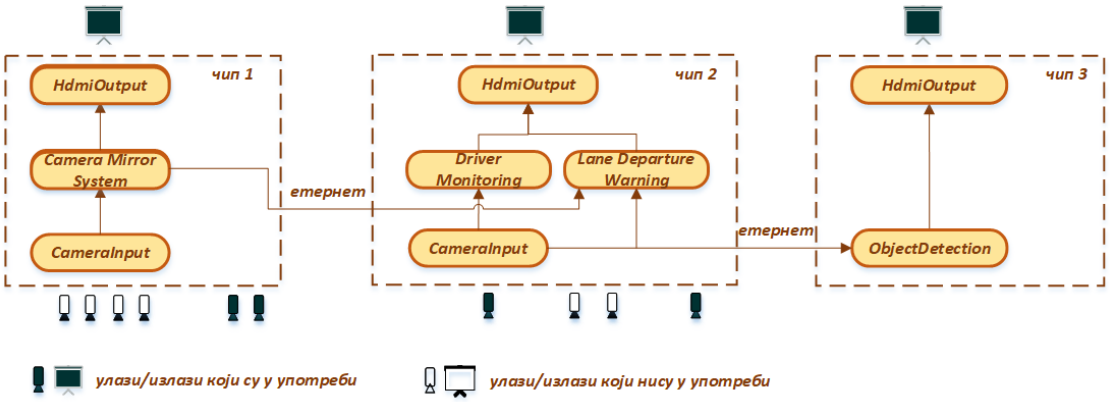
| Апликација | Приказ |
|---|---|
| Назив: Дигитални ретровизор *(енгл. Camera Mirror System)*<br>Опис: Детектује прилазак другог возила са задње стране, његову удаљеност и убрзање. Користи две камере (леви и десни ретровизор) које снимају видео запис који се приказује на монитору. Приказ резултата на екран. |  |
| Назив: Праћење возача (енгл. *Driver Monitoring*)<br>Опис: Детектује присуство возача, анализира карактеристике лица, очију и зеница, и даје информације о будности возача (поспан, активан, не гледа на пут). Користи инфрацрвену камеру у возилу. Приказ резултата на екран. |  |
| Назив: Упозорење о напуштању траке (енгл. *Lane Departure Warning*)<br>Опис: Детектује напуштање текуће саобраћајне траке, и сигнализира исто. Користи предњу камеру у возилу (у случају прототипа, камера снима монитор на коме се емитује видео запис). Приказ резултата на екран. |  |
| Назив: Детекција објеката (енгл. *Object Detection*)<br>Опис: Детектује објекте који се налазе испред возила (врсту, позицију и удаљеност објекта: возило, пешак, саобраћајни знак). Користи предњу камеру (у случају прототипа, камера снима монитор на коме се емитује видео запис). Приказ резултата на екран. |  |

| Апликација | Приказ |
|---|---|
| Назив: Приказ окружења возила (енгл. *Surround View*)<br>Опис: Приказ окружења возила са свих страна. Користи четири широкоугаоне камере које су постављене на четири бочне стране прототипа, обрађује слике и спаја у једну како би дао приказ возила и окружења одозго. Приказ резултата на екран. |  |
| Назив: Приказ задње камере (енгл. *Rear View*)<br>Опис: Приказ окружења возила са задње стране и детекција препреке. Користи се једна камера која је постављена на задњу страну прототипа возила. Приказ резултата на екран. |  |

*Табела 5.4 Апликације за помоћ возачу*

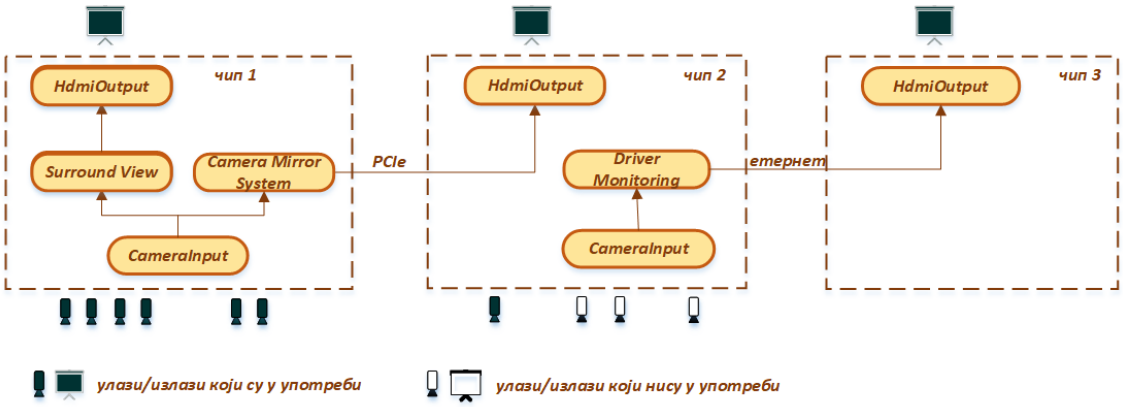
Демонстрациони случај је усмерен на рад у реалном времену, окружење више чипова, доступност ресурса апликацији са свих чипова као и пренос података путем разних магистрала. У систему постоје три режима рада који у зависности од потребе користе одређене камере и покрећу одређене алгоритме:

- Режим вожње – активни су алгоритми који су потребни приликом вожње – дигитални ретровизор, детекција објеката испред возила, праћење возача и упозорење о напуштању траке. Због већег броја обрадних јединица при реализацији и због једноставности приказа, на слици испод је приказан логички пајплајн за овај режим. Саме обраде су приказане у једној обрадној јединици ради поједностављења слике. Овим се демонстрира пренос видео садржаја и догађаја преко етернет магистрале, као и доступност исте камере на два чипа.



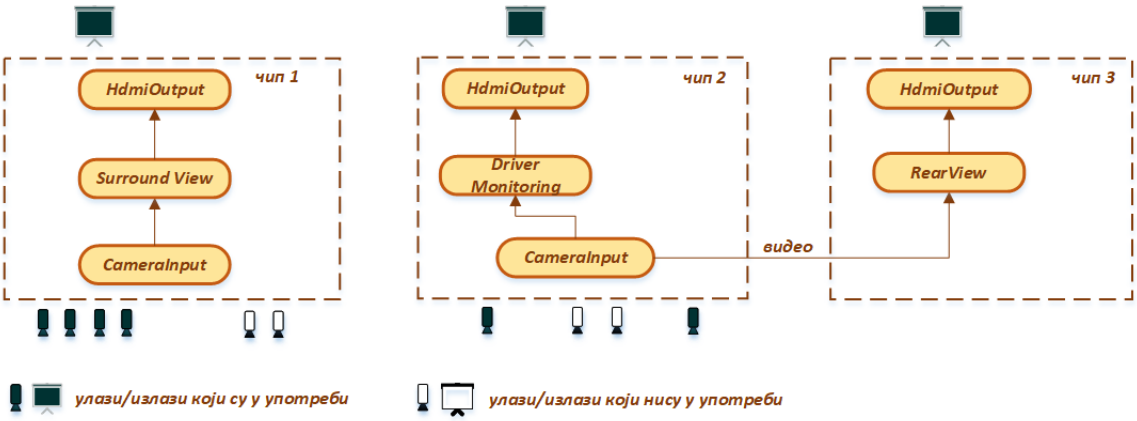

*Слика 5.17 Режим вожње - најплајн*

- Режим паркирања - активни су алгоритми који су потребни приликом паркирања – окружење возила, праћење возача и дигитални ретровизор. Демонстриран је пренос видео садржаја преко *PCIe* и етернет магистрала између чипова.




*Слика 5.18 Режим паркирања - најплајн*

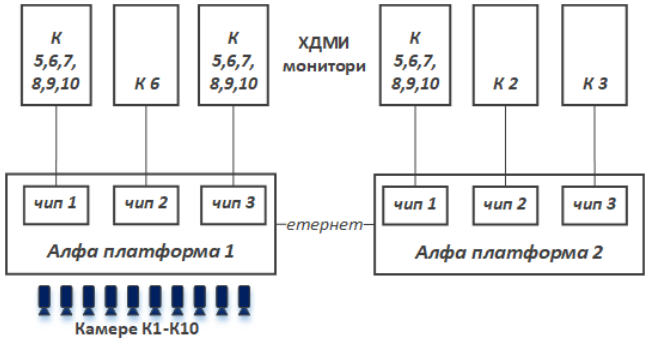
- Неутрални режим – активни су следећи алгоритми: приказ окружења возила, праћење возача, приказ задње камере. Демонстриран је између осталог пренос садржаја преко видео магистрале.




*Слика 5.19 Неутрални режим - најплајн*

**Дистрибуција велике количине података у реалном времену**

Коришћењем средњег слоја, рад са свих десет камера као и конфигурација истих у реалном времену приказана је у [150]. У овом случају коришћене су две *Alpha* платформе, где је десет камера повезано на једну плочу, а демонстрира се пренос свих десет камера између шест чипова на ове две платформе, од чега је један тестни конфигурациони случај приказан на слици испод. Додатно, направљен је алат којим се демонстрира конфигурабилност самих обрадних јединица, у овом случају улазних јединица – камера.




*Слика 5.20 Кориснички случај са две Alpha платформе*

## 5.3. СЛОЈ АПСТРАКЦИЈЕ ПЛАТФОРМЕ ЛИНУКС *X86*

Животни циклус развоја алгоритма обично почиње на персоналном рачунару где се тестира исправност самог алгоритма, након чега се исти преноси на циљну платформу и врши се оптимизација. Након преноса на циљну платформу, потребно

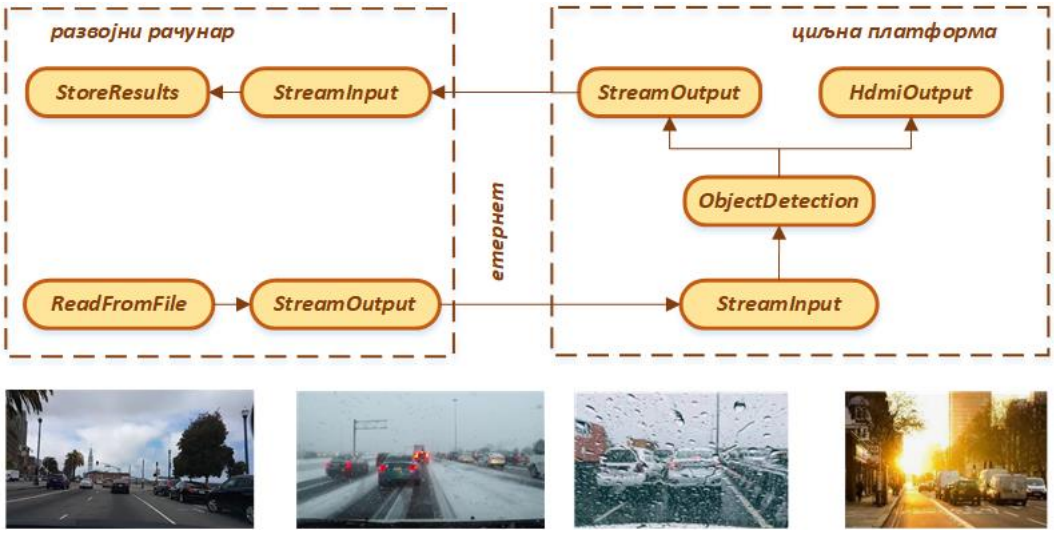
је потврдити очување исправности алгоритма. Коришћење АМВ спреге омогућава лакши процес тестирања и верификације. АМВ спрега омогућава симулацију улазних података са персоналног рачунара, као и чување резултата алгоритма (или међукорака) за даље проучавање. Често се са персоналног рачунара симулира камера (са мета-подацима релевантним за алгоритам) коришћењем унапред снимљених видео записа.

Слој за апстракцију је реализован и за x86 Линукс платформу, углавном за сврхе помоћи при развоју самог алгоритма (иницијални развој, лакше дебаговање, симулирање, постепено преношење на циљну платформу, итд.).

При реализацији слоја коришћен је *Ubuntu* оперативни систем и стандардне *C* и Линукс библиотеке. Јединица за обраду је реализована за општенаменску процесорску јединицу.

### 5.3.1. Примери апликација

На примеру испод дат је приказ тестирања алгоритма за детекцију објеката. Алгоритам се побуђује са стране рачунара разним видео записима који представљају различите реалне услове (у овом случају разне временске прилике). Резултати алгоритма се приказују на ХДМИ излазу циљне платформе а у исто време детектовани објекти и информације о њима се шаљу на персонални рачунар.




*Слика 5.21 Пример најплајна и тестних снимака за тестирање апликације*

## 5.4. ДИСКУСИЈА

У оквиру овог рада средњи слој је реализован на три различите хардверске платформе (тестни рачунар, инфо-забавна и АДАС платформа). Физички слој се ослања на спреге које нуде саме платформе (оперативни систем, развојна окружења и библиотеке). Реализација на платформу на којој се налази хипервизор није демонстрирана. Реализација физичког слоја над оперативним системом изнад хипервизора би се одрадила на исти начин као и без хипервизора, а реализација физичког слоја над хипервизором се своди на реализацију функција спрегама које су специфичне за хипервизор.

Реализована су два реална система апликација којима се демонстрира рад у реалном времену, коришћење спрега средњег слоја, добављање информација са сензора, дистрибуција података до циљне процесне јединице, вишеструке обраде и приказ података. Приказује се коришћење хетерогених платформи кроз униформне спреге, управљање апликацијама, рад у реалном времену. Алати за рад са предложеним средњим слојем нису реализовани, али постоји добра основа за њихову реализацију на нивоу саме платформе - спреге које су дефинисане у оквиру AMB апликације су прилично униформне и предвидљиве и постоји могућност генерисања истих. У раду [150] је реализован један мањи део алата који се ослања на овај средњи слој. Алат се односи на комуникацију између чипова и конфигурацију специфичних јединица обраде које су задужене за ову комуникацију – чиме се показује да је развој алата за овај средњи слој изводљив. Алат за цијелу платформу је део будућег рада и обухватао би све аспекте платформе.

Иако је реализован један подскуп предложеног решења, гледало се да подскуп обухвати све релевантне делове у одређеном облику како би реализација била репрезентативна (разноврсне хетерогене платформе, магистрале, сензоре, спрегу са другим доменима, помоћне функције, симулатори, рад у реалном времену итд.). Одређени аспекти су реализовани у мањој мери, али и даље представљају адекватну имплементацију у смислу основе за евалуацију хипотезе овог рада.

# 6. ПРОВЕРА РЕШЕЊА

Исправност решења и његова усклађеност са захтевима се проверавају на више начина, који су представљени у Поглављу 4. Део захтева је такође директно осликан у самој предложеној архитектури.

## 6.1. ЕКСПЕРИМЕНТАЛНА ПРОВЕРА

Реализацијом средњег слоја на три платформе описане у поглављу 6, експериментално је потврђено да је предложена архитектура средњег слоја изводљива и употребљива. Поред апликација приказаних у поглављу 6, реализован је већи број корисничких случајева коришћења овог средњег слоја, од чега су репрезентативни тестни случајеви приказани у Додатку Б.

У наставку је табеларно дато, који експериментални тестни случајеви се односе и утичу на проверу сваког од функционалних захтева. Колоне представљају тестне корисничке случајеве представљене у Додатку Б., и два система апликација реализована на инфо-забавној платформи – поглавље 5.1.3 (инфо) и АДАС платформи - поглавље 5.2.3 (адас).

| Тестни случајеви | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | инфо | адас |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Апстр. хетерогености* | *Захтев 1.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 2.* | x | x | x | x | x | x | x | x | x | | x | | | | | | | | x | | | x | | | | x | x | x |
| | *Захтев 3.* | x | x | x | x | x | | | x | x | | | | | | | | | | | | | x | | | | x | x | x |
| | *Захтев 4.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 5.* | | | | | | | | | | | | | | | | | x | x | x | x | | | x | x | x | | | x |
| | *Захтев 6.* | | | | | | | | | | x | x | x | x | x | x | x | x | x | x | x | x | x | | x | | | x | x |
| | *Захтев 7.* | | | | | | | | | | | | | | | | | x | x | x | x | x | | x | x | x | | x | x |
| | *Захтев 8.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 9.* | | | | | | | | | | | | | | | | | x | x | x | x | | | x | x | x | | | x |
| | *Захтев 10.* | | | | | | | | | | | | | | | | | | | | x | | | x | | x | | x | x |
| | *Захтев 11.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 12.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| *Рук. апл.* | *Захтев 13.* | | | | | | | x | x | x | x | x | x | | | | x | | | x | x | | x | | | | | x | x |
| | *Захтев 14.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 15.* | | | | | | | | | | | | | | | x | | | | | | | | | | | | x | |
| | *Захтев 16.* | | | x | x | x | | | | | | | | | | | | | | | | | | | | | x | x | x |
| *Пом. функц.* | *Захтев 17.* | | | | | | | | | | | | | | | | | | | | | | | x | x | x | | | |
| | *Захтев 18.* | | | | | | | | | | | | | | x | | | | | | | | | x | x | x | | | |
| | *Захтев 19.* | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x | x |
| | *Захтев 20.* | | | | | | | | | | | | | x | | | x | | | | | | | | | | | x | x |

*Табела 6.1 Експериментална провера захтева*

Средњи слој обезбеђује униформну спрегу која апстрахује хетерогеност платформе (*Захтев 1.*). Корисници на јединствен начин користе разне процесорске јединице (*CPU, GPU, DSP*) – постављањем одговарајућег параметра обрадне јединице (*CoreAffinity*). Извршавање обрадне јединице на одређеној процесорској јединици зависи и од њене реализације (тј. да ли се користе обраде специфичне за одређену процесорску јединицу). Ако је реализација уопштена, онда је могуће инстанцирати исту обрадну јединицу више пута на истим и/или различитим процесорским јединицама. На примеру тестног случаја 26 (Додатак B), обрадна јединица за копирање слике се инстанцира на *CPU, GPU* и *DSP* језгру. Корисницима је доступна спрега за размену података користећи представљени механизам редова и не морају да воде рачунара о реализацији преноса података између процесорских јединица (*Захтев 8.*).

Подршка за различите типове процесорских јединица (*Захтев 2.*) је експериментално потврђена на више платформи. Подршка за следеће процесорске јединице је реализована у пракси:

- *CPU: ARM A15, ARM M4, Kryo, x86* (тестни случајеви 6, 7, 8, 9),
- *DSP: C66, Hexagon 680* (тестни случајеви 1, 2, 5, 11, 22),
- *GPU: Adreno 530* (тестни случајеви 3, 4, 5, 26),
- *VPU: EVE* (тестни случај 22).

Подршка за вишејезгарне процесорске јединице (*Захтев 3.*) је потврђена на примеру централних процесорских јединица: двојезгарне ARM процесорске јединице на *Alpha* платформи и централне процесорске јединице са четири језгра на инфо-забавној платформи. Избор језгра се врши на исти начин као и избор процесорске јединице (преко параметра за афинитет језгра). Тестни случајеви 2, 8, 9, 22, 26 показују коришћење вишејезгарних процесорских јединица.

Слој апстракције платформе средњег слоја је реализован на два наменска чипа: *TDA2x* и *S820a.* Примери реализације су приказани у Поглављу 6, чиме је експериментално потврђен *Захтев 4.* – подршка за више врста чипова. На *TDA2x* чипу средњи слој се извршава под Линукс и RTOC оперативним системима, док се на *S820a* користи Андроид оперативни систем. Овим се испуњава захтев да се подржи извршавање на две врсте оперативних система: за рад у реалном времену и општијих оперативних система (*Захтев 11.*).

Демонстрирано је и извршавање апликације над више чипова (*Захтев 5.*). На примеру на АДАС платформи - систем апликација за помоћ возачу, види се коришћење ресурса више од једног чипа за извршавање апликације (са другог чипа се користи недостајућа камера или приказ на екран). Тестни случајеви 18, 19 и 20 приказују респективно коришћење камере са другог чипа, коришћење процесорских ресурса за обраду са другог чипа, као и коришћење ресурса три чипа за реализацију апликације.

Подршка за сензоре и актуаторе (*Захтев 7.*) је омогућена креирањем улазних и излазних обрадних јединица. У тестним примерима реализовано је неколико сензора, који се користе кроз примере апликација у Поглављу 6:

- Сензор камере на инфо-забавној платформи (подршка до 6 камера, резолуција: 1280x720, формат слике: YUV422, број слика у секунди: 30),
- Сензор камере на АДАС платформи (подршка до 6 камера, резолуција: 1280x720, формат слике: YUV420, број слика у секунди: 30),
- Сензор камере на АДАС платформи (подршка до 4 камере, резолуција: 1280x720, формат слике: YUV422, број слика у секунди: 30),
- Ултразвучни сензор на инфо-забавној платформи (подршка до 4 сензора).

Средњи слој пружа подршку за више различитих магистрала (*Захтев 7.*), реализујући *Bus* модул. Примере коришћења магистрала од стране АМВ средњег слоја налазимо у примеру система АДАС апликација, где се користе три магистрале за пренос слика између чипова: *PCIe*, видео и етернет магистрала. Етернет магистрала се користи и за комуникацију са персоналним рачунаром (тестни случајеви 23, 24, 25). Поред ове три магистрале, реализоване су и *CAN* и *UART* магистрале. Коришћење магистрала од стране апликација је демонстрирано у тестном случају 21 (читање са *CAN* магистрале).

Један од разлога коришћења разних магистрала од стране АМВ средњег слоја је у сврху комуникације између чипова (*Захтев 9.*). Поред поменутог преноса слика између чипова у реализацији камера сервиса (тестни случајеви 17, 18), омогућени су и демонстрирани преноси произвољних података (тестни случајеви 19, 20).

Проток веће количине података (*Захтев 10.*) је демонстриран у примерима апликација које врше обраду слика са камера (систем апликација за помоћ возачу

који користи девет камера на АДАС платформи и систем од две апликације на инфо-забавној платформи који користи пет камера). Демонстрирано је достављање података у оквиру истог чипа, као и на други чип, као и спрезање са другим доменом (у овом примеру приказ резултата на инфо-забавном екрану)(*Захтев 15.*).

Реализоване апликације у поглављу 6. користе постојеће хетерогене библиотеке/спреге, нпр. у апликацијама инфо-забавног система користе се *OpenGLES* и *OpenCL* спреге (*Захтев 16.*).

Управљање животним циклусом апликације демонстрира се кроз било који од реализованих токова података, користећи спреге за покретање и заустављање обрадних јединица. Параметризовање апликације при покретању, као и током извршавања апликације користећи одговарајуће спреге је демонстрирано у тестним случајевима 14, 15 и 16 (*Захтев 14.*).

Захтев за подршку за извршавање на развојном рачунару у сврху развоја апликација (*Захтев 17.*) представља реализацију још једног слоја за апстракцију чија реализација и пример апликације су описани у поглављу 5.3. Тестни случајеви 23, 24 и 25 демонстрирају примере коришћења развојног рачунара и циљне платформе у сврхе бржег развоја и верификације. Врши се симулација разних података са развојног рачунара (*Захтев 18.*). Симулацију је могуће вршити и директно са циљне платформе, али то зависи од величине података који треба да се симулирају и од ресурса саме платформе. У случају симулације већих количина података (нпр. камера), обично је једноставније симулацију радити са развојног рачунара или у комбинацији са развојним рачунаром.

## 6.2. ПЕРФОРМАНСЕ РЕШЕЊА

AMB средњи слој представља још један софтверски слој чиме уноси додатно кашњење у обради. Измерене су следеће ствари: свеукупне перформансе AMB-а на циљној платформи у приближно реалним корисничким случајевима у лабораторијским условима и кашњење у обради унето додатним слојем софтвера.

### 6.2.1. Перформансе реализованих система

Табела 6.2 приказује број обрађених слика у секунди за апликације у реализованим системима. Овим се демонстрира могућност извршавања реалних корисничких случајева у реалном времену (*Захтев 13.*).

| **Систем** | *Инфо-забавни* | | *АДАС* | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Апликација** | 3Д окружење возила | Праћење возача | Дигитални ретровизор | Праћење возача | Упозорење о напуштању траке | Детекција објеката | Окружење возила | Приказ задње камере |
| **Број слика у секунди** | 30 | 20 | 27 | 14 | 30 | 16 | 30 | 30 |

*Табела 6.2 Број слика у секунди по апликацији*

Аутори радова [151] и [152] износе перформансе њихових решења мерено бројем слика у секунди. У [151] број слика у секунди за различите перспективе детекције трака и објеката износи од 13-16 (обрада се врши над подацима једне *USB* камере) што сматрају радом у реалном времену за апликације према [153]. У [152] број слика у секунди за имплементирани систем (упозорење на промену траке, детекција траке) зависи од броја коришћених камера. Аутори користе једну (16.5 слика у секунди) и две камере (12 слика у секунди). У оба рада резолуција камере је 640 x 480.

Ако упоредимо резултате са резултатима система апликација из овог рада, можемо видети да је број слика у секунди за одређене апликације исти или већи, док је количина података са камера које се обрађују значајно већа у овом раду (пет до девет камера веће резолуције).

### 6.2.2. Размена података између јединица обраде

У наставку следе измерени подаци за две платформе које су описане у Поглављу 6. Мерено је утрошено време између две суседне јединице обраде – тј. њихових *worker* метода. Мерено време обухвата време потребно за извршавање AMB апстракција језгра и време преноса података између ове две јединице обраде. Прва колона у табелама описује језгра на којима се извршавају ове две јединице обраде. У осталим колонама је приказано измерено време у милисекундама између датих јединица обраде за различите количине података (количина података је

назначена на почетку сваке колоне). Количина података представљена као С, представља једну слику резолуције 1280x720, 2 бајта по пикселу, што свеукупно чини 1843200 бајтова (~1.75 мегабајта). Мерења су вршена за једну, две и четири слике, што представља уобичајене бројеве камера који се користе за разне алгоритме за визију.

| Величина података: | 1КБ | 1С | 2С |
|---|---|---|---|
| *CPU <-> CPU* | ~0.3ms | ~0.46ms | ~0.47ms |
| *DSP <-> DSP* | 0.5-1.5ms | 3-12ms | 5-14ms |
| *GPU* <-> GPU** | 1-5ms | 1-14ms | 1-19ms |
| *CPU <-> DSP* | 0.5-1.5ms | 2-6ms | 4-10ms |
| *CPU <-> GPU** | 0.7-5ms | 1-13ms | 2-17ms |
| *DSP <-> GPU** | 1-5ms | 3-12ms | 5-15ms |

*Две различите реализације за GPU су коришћене током мерења. Вредности за једну су блиске нижем броју, док су за другу блиске вишем броју

*Табела 6.3 Измерена времена при размени података – платформа S820a*

| Величина података: | 1С | 2С | 4С |
|---|---|---|---|
| *CPU1 <-> CPU1* | ~0.162ms | ~0.279 ms | ~0.509 ms |
| *CPU2 <-> CPU2* | ~0.357 ms | ~0.357 ms | ~0.359 ms |
| *DSP <-> DSP* | ~0.610 ms | ~1.107 ms | ~2.074 ms |
| *VPU <-> VPU* | ~5.264 ms | ~5.313 ms | ~5.337 ms |
| *CPU1 -> CPU2* | ~0.504 ms | ~0.569 ms | ~0.831 ms |
| *CPU2 -> CPU1* | ~0.541 ms | ~0.712 ms | ~0.930 ms |
| *CPU2 -> DSP* | ~0.816 ms | ~1.136 ms | ~1.723 ms |
| *DSP -> CPU2* | ~0.961 ms | ~1.411 ms | ~2.425 ms |
| *CPU1 -> DSP* | ~0.587 ms | ~0.998 ms | ~1.838 ms |
| *DSP -> CPU1* | ~0.747 ms | ~1.234 ms | ~2.211 ms |

*Табела 6.4 Измерена времена при размени података – платформа Alpha*

Мерења за платформу *S820a* су дата у облику опсега, због чињенице да времена варирају са сваким покретањем – процеси у позадини користе исте ресурсе. За веће количине података времена су приметно већа. Ово је делимично због специфичне реализације AMB пајплајна за ову платформу, где се подаци интерно враћају на *CPU*, као контролну процесорску јединицу, након сваке обраде у јединици обраде (без обзира на ком језгру се обрада дешавала). Додатно, за ову платформу, мерења су извршена и за мању количину података (1КБ) како би се

проверило да је кашњење изазвано преносом веће количине података, а не нечим другим. У већини случајева време се смањило, као што је и очекивано, сем за *GPU* реализацију где је кориштен *OpenCL* – тачан узрок треба да буде утврђен.

Измерене вредности из Табела 6.4 указују да су перформансе јако зависне од саме реализације слоја апстракције платформе за дато језгро. На пример, линеарна зависност времена и количине података за *CPU1* и *DSP* језгра показује да се праве дубоке копије података у преносима података у којима учествују та два језгра, у односу на друга језгра где ова зависност није присутна и где се само плитке копије праве. Ако су дубоке копије података стварно потребне, разне оптимизације могу бити урађене у зависности од тога шта подржава платформа (нпр. ДМА копирање). Велико време потребно за трансфер на *VPU* језгру је последица чињенице да је кориштен неоптимизовани/генерички слој за апстракцију платформе – користи се апликативни део *VPU* језгра, без векторског копроцесора.

Сва мерења су извршена у оквиру једног чипа. Време размене података између чипова зависи од комуникационог канала који се користи (*PCIe*, Етернет) и од саме реализације AMB слоја за апстракцију платформе на датим чиповима.

У [154] аутори дају преглед системских захтева за пренос разних података (контролних, видео, аудио) између *ECU*-ова – од камера до крајњег места где се обрада врши. За видео податке дефинишу највеће време преноса и обраде података до 45ms, скрећући пажњу да би време преноса података требало да буде највише 33ms. Пренос података се заснива над компресованим подацима, и количина података коју преносе је значајно мања него ли што је мерено у овом раду. Поредећи добијена времена овог решења видимо да су далеко испод дефинисаних времена за пренос видео података. Главни разлог за оваква времена је централна платформа високих перформанси тј. довољно ресурса за извршавање апликације и сензори повезани на централну платформу, као и чињеница да је комуникација између процесорских језгара реализована користећи меморију, а не етернет везу. Додатно, у овом решењу за кодовање и декодовање слике користе се платформски хардверски блокови. За контролне податке (реда величине од 32 до 512 бајта), према ауторима, дефинише се према најстрожијим критеријумима, време преноса до 2.5ms, док је у већини случајева ово време веће од 10ms. Измерена времена за

веће количине података су углавном у границама и овог најстрожијег критеријумима, са изузецима који су већ адресирани.

Овим смо показали да референтна реализација даје задовољавајуће перформансе за развој на циљној платформи, и да и даље има места за напредак и усавршавање реализације слоја за апстракцију.

## 6.3. ОЦЕНА КВАЛИТЕТА У ОДНОСУ НА ПРЕДЛОЖЕНЕ МЕТРИКЕ

За израчунавање софтверских метрика коришћени су доступни бесплатни софтверски алати PCM [155] и CM [156]. У Додатку B. се налази опис кориштених метрика доступних у алатима, добијени резултати тих метрика, као и преглед додатних изведених метрика за чије израчунавање се користе резултати које дају алати. PCM алат пружа анализу на нивоу датотека, свих датотека у пројекту (која обухвата три реализације слоја за апстракцију платформи), док CM алат узима у обзир само реализацију слоја за апстракцију платформе на Линукс оперативном систему – развојни рачунар.

### 6.3.1. ХИС софтверске метрике

У наставку је дата дискусија и тумачење добијених резултата у односу на препоручене резултате од стране ХИС конзорцијума. Детаљни приказ добијених резултата се налази у Додатку B.

**Густина коментара**

Препоручена граница од стране ХИС конзорцијума је 0.2. Са графика (Слика B.1) се види да пар *.cpp* датотека има мању вредност од прописане (између 0.1 и 0.2) То је надомештено њиховим припадајућим *.h* датотекама. Такође, приметно је да *.h* датотеке имају већи проценат коментара и документације, што је очекивано јер постоји већи број виртуелних метода и интерфејса који су детаљније документовани у овим датотекама. Просечна вредност густине коментара за цели софтверски пакет је 0.51 у односу на број линија кода и 0.58 у односу на ефективни број линија кода. Из овога можемо закључити да је изворни код одговарајуће документован и да је разумевање истог за потребе одржавања, мењања и тестирања унапређено у односу на одсуство истих коментара.

**Број скокова**

Претрагом изворног кода утврђено је да се нигде не користи *goto* наредба, чиме је испуњен критеријум за ову метрику – тј. укупан број скокова једнак је нули.

**Циклична сложеност**

Оптимална вредност цикличне сложености према ХИС конзорцијуму је у опсегу од 1 до 10, док одређени аутори тврде да је за *C++* опсег до 20.

Просечне вредности сложености по функцији у датотеци (Слика В.2) су у оквиру оригиналног опсега за ХИС метрике. Што се тиче максималне вредности по функцији за дату датотеку, постоји укупно седам функција чија вредност излази ван опсега 1-10, тј. две чија вредност прелази 20.

Гледајући хистограм вредности цикличне сложености по методама (Слика В.6) број метода које имају вредност цикличне сложености већу од 10 је 23, што чини 0.07% метода. У случају узимања границе од 20, број метода ван опсега је 7. (0.02%).

**Број параметара функција**

Препоручени опсег вредности од стране ХИС конзорцијума је од 0 до 5. На графицима (Слика В.3) се на три места се види да је присутно шест параметара као максимални број параметара функције по датотеци. На ова три места просечна вредност је знатно мања од 6, тако да се може закључити да је у питању по један изузетак у свакој од датотека.

**Број инструкција по функцији**

Оптимална вредност према ХИС конзорцијуму је у опсегу од 1 до 50

Вредности броја инструкција по функцији су измерене користећи оба алата. PCM алат даје вредности просечног и највећег броја инструкција по функцији у датотеци (Слика В.4, Слика В.5), док CM алат даје вредности по методи (Слика В.9). Пет функција има више од 50 линија кода посматрајући број линија кода, а када се посматра ефективни број линија кода, онда три функције излазе ван опсега. Број метода које имају *LOC* већи од 50 је седам, тј. ~ 0.02%.

**Број излазних тачака из функције**

Препоручене вредности од стране ХИС конзорцијума су једна или ниједна излазна тачка из функције. Вредности за број повратних тачака из функције су добијене кроз PCM алат. Из добијених резултата се види да је углавном у питању једна излазна тачка из функција. У једној функцији број излазних тачака из функције је већи од један, што би требало исправити у даљем развоју.

**Опсег језика**

Опсег језика представља индикатор трошкова одржавања/мењања функција. Препоручене вредности од стране ХИС конзорцијума су у опсегу 1 до 4. На приказаним графицима (Слика В.10) 12 метода има индекс преко 4, али близак броју 4, тј. 0.02% метода излазе из опсега док је већина метода у дефинисаном опсегу.

**Дискусија**

Прегледом резултата ХИС метрика може се закључити да је реализација у складу са ХИС метрикама, испоштоване су добре праксе дефинисане за развој софтвера у аутомобилској индустрији. Усклађеност са ХИС метрикама доприноси бољој вредности степена одржавања софтвера и утиче позитивно на портабилност софтвера, што свеукупно доприноси краћем времену развоја и краћем времену потребном за тестирање софтвера. Овим се верификују *Захтеви 23., 24.* и *28.*

### 6.3.2. Објектно-оријентисане софтверске метрике

Разни аутори дефинишу различите прагове за Чидамберове и Кемерерове метрике. Износе се вредности које указују само на граничне вредности које су специфичне за стварни контекст (програмски језик, примена, итд.) и стога не дозвољавају апсолутне изјаве у односу на дате вредности. Како би илустровали разноликост опсега, у Табели 6.5 су приказане граничне вредности разних аутора.

Постоји више смерница од разних аутора како да се интерпретирају ове метрике, али не постоји довољно података који показују да дупла вредност неке метрике уједно представља и дупло бољи/лошији показатељ. Аутори у [142] предлажу интерпретацију која се заснива на поређењу вредности, са освртом на

вредности које одскачу, како би се анализирале и како би се утврдило зашто се разликују од других модула. Наглашавају да ако нешто одскаче, не значи нужно да је лоше, него је индикатор различитости који треба да се даље анализира.

| **Метрика** | **Извор** | | | | | |
|---|---|---|---|---|---|---|
| | *CK* [133] | *Rosenberg* [142] | *Benlarbi* [157] | *Vogel* [130] | *Together Soft* [158] | *SD Metrics* [158] |
| *WMC* | | 100 | 100 | 26* | 100 | |
| *DIT* | | 6 | / | 6 | 4 | 0-3 |
| *NOC* | | / | / | 6 | | |
| *CBO* | 14 | 5 | 5 | 25 | 30 | 0-31 |
| *RFC* | | 100 | 100 | 34-74 | | 3-365 |
| *LCOM* | | / | / | 40-108* | | |

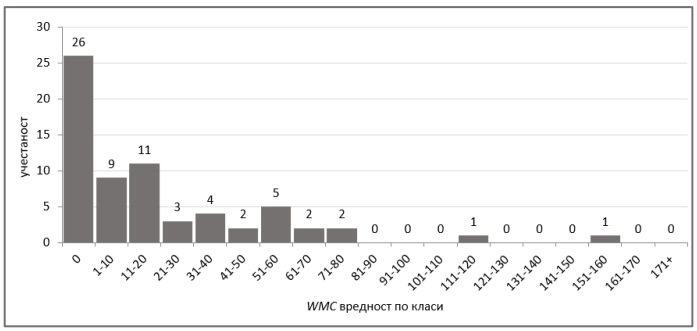
*представља број класа, ** нема детаља у односу на који LCOM су ово границе

*Табела 6.5 Препоручене граничне вредности за ОО метрике*

У наставку је дат преглед и дискусија добијених резултата за реализовани средњи слој.

### Сложеност пондерисаних метода – WMC

*WMC* метрика мери сложеност класе и показује колико времена и труда је потребно да се развије и одржава одређена класа. Слика 6.1 приказује хистограм добијених вредности по класи. Уопштено, према [142] пожељна вредност ове метрике је испод 100, а не би требала да пређе 200. Већа вредност ове метрике указује на већу потребу за тестирањем и одржавањем. Са хистограма се види да се две класе разликују у односу на остале и да имају вредност већу од 100, док остале потпадају под критеријуме дефинисане од већине аутора.

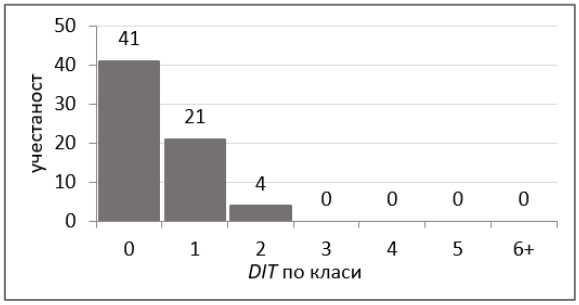



*Слика 6.1 Хистограм на нивоу класе*

**Дубина стабла наслеђивања – *DIT* и број подкласа - *NOC***

Метрике које проучавају хијерархијску структуру су приказане на сликама испод. Класе које имају ниску вредност дубине наслеђивања имају више потенцијала за поновно коришћење и теже ка томе да буду апстрактне класе, што је овде и случај. Апстрактне класе (већински из слоја за апстракцију платформе) имају вредност 0, као и класе из самог језгра. Јединице обраде које се изводе из датих класа као и сама реализација слоја за апстракцију платформе наслеђују дате класе и њихова вредност ове метрике расте. Интерфејси такође имају вредност нула за дубину наслеђивања (нису урачунати у график са слике).

Највећа вредност измерене метрике је два. Мерења су вршена са основним јединицама обраде. Граница коју већина аутора поставља за ову метрику је шест, што даје простора за даља проширења јединица обраде, а да целокупни систем и даље буде у прихватљивом стању за одржавање.

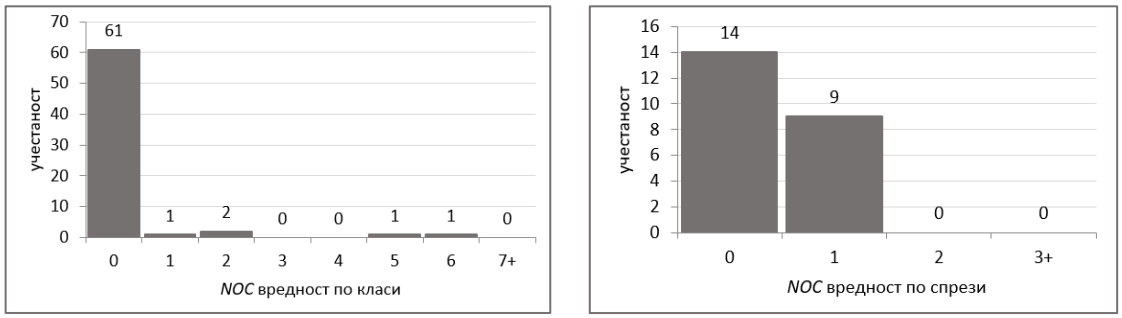



*Слика 6.2 Хистограм дубине стабла наслеђивања (класе)*

Број подкласа се може користити да се евалуира потенцијални утицај класе на целокупни дизајн. Класе које имају велики број подкласа се сматрају лоше дизајнираним класама. Слика 6.3 а) показује да већина класа нема подкласе, што је и очекивано. Већина основних класа које представљају обрадне јединице, а које су овде укључене у резултате немају подкласе, као ни класе које реализују слој апстракције платформе. Слична ситуација је и са интерфејсима и њиховим бројем подкласа (Слика 6.3 б)).

Резултати мерења ове две метрике би се знатно разликовали ако би се анализирали над целокупним системом (тј. укључујући апликације које се извршавају над конкретном платформом и које имају реализоване сложеније

обрадне јединице). У овом случају анализирано је језгро средњег слоја и показна основна апликација на персоналном рачунару, због ограничења самог алата.

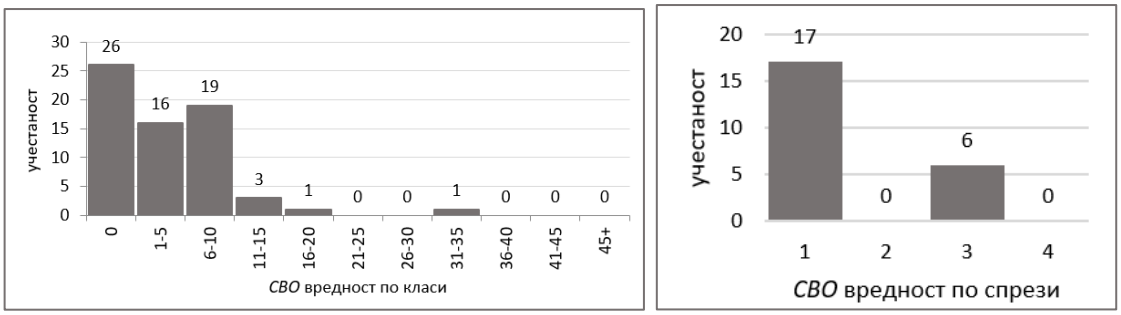



*а)* *б)*

*Слика 6.3 Број подкласа а) класа б) спрега*

**Повезаност објеката – *CBO***

Слика 6.4 а) приказује хистограм вредности повезаности објеката на нивоу класе. Са хистограма се види да је више од трећине класа самостално. Веће вредности указују на то да се смањује вероватноћа поновног искоришћења одређене класе. Вредности ове метрике би требало држати на ниском нивоу. Међу резултатима, једна класа одскаче од остатка вредности и то је *PALFactory* класа, што је и очекивано. Ова класа је због своје природе задужена за креирање одговарајућих објеката других класа.



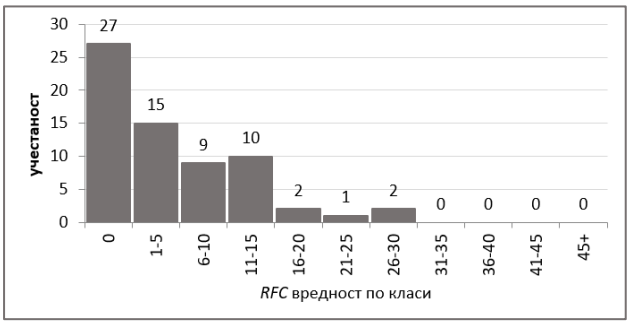

*а)* *б)*

*Слика 6.4 CBO на нивоу а) класа б) спрега*

*CBO* вредност на нивоу интерфејса је у складу са очекивања – мала вредност (Слика 6.4 б)). Већина интерфејса (74%) су самостални, а остали имају малу вредност, што иде у прилог поновном искоришћењу.

**Број одговора класе – *RFC***

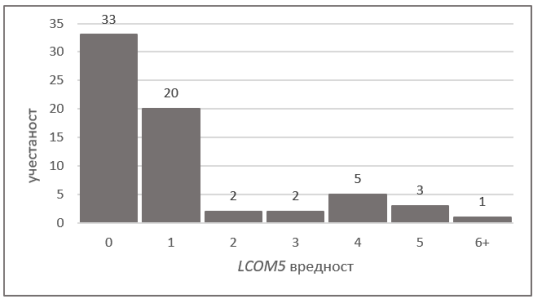
Слика 6.5 приказује хистограм вредности броја одговора класе. Вредности ове метрике нису велике и не прелазе генерално дефинисане горње границе (већина аутора дозвољава вредност до 100). Оно што се може приметити са хистограма је да ова метрика има ниже вредности што указује на већи полиморфизам. Ниске вредности броја одговора за класу утичу на повећање степена одржавања, разумљивости и поновног искоришћења.




*Слика 6.5 Хистограм броја одговора класе*

**Недостатак повезаности метода у класи – *LCOM5***

CM алат рачуна ову вредност као *LCOM5* (опис у Додатку B.), што представља једну од алтернатива у односу на оригиналну метрику. Ниже вредности указују на већу кохезију међу класама. Вредности јединица обраде (сервисима) теже ка већим вредностима у односу на остале – што је очекивано због разноликости обрадних јединица.




*Слика 6.6 LCOM5 на нивоу класе*

**Дискусија**

У овом поглављу дата је анализа резултата објектно-оријентисаних метрика. Већина резултата је на задовољавајућем нивоу, тј. у оквиру устаљених граница са којима се слаже већина аутора. Додатно дата је анализа резултата који одскачу од устаљених вредности и дати су разлози одскакања појединих вредности – који су углавном због природе средњег слоја и намене истог.

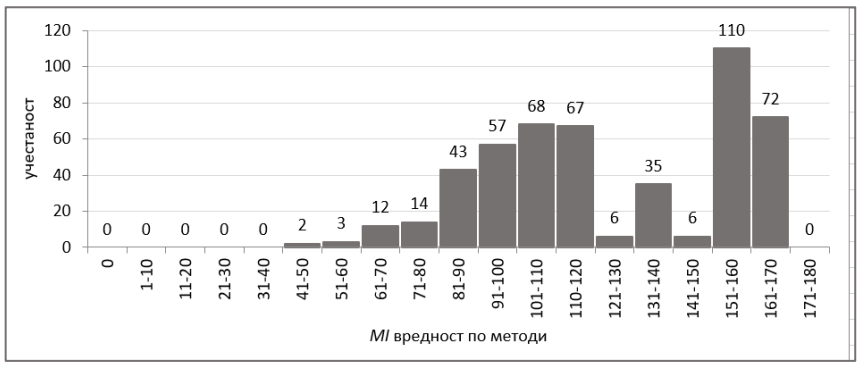
Гледајући хистограме вредности резултата, примећујемо да су већински резултати метрика *WMC, RFC, LCOM* и *CBO* на нижем нивоу, што доприноси смањењу времена тестирања и времена развоја, већој разумљивости изворног кода, повећању степена одржавања, и повећању степена поновног искоришћења софтвера. Овим се верификују *Захтеви 23., 25., 26.* , као и захтев за усклађеност са добрим праксама, у овом случају са објектно-оријентисаним праксама (*Захтев 28.*).

### 6.3.3. Атрибути квалитета софтвера

У овом потпоглављу представљено је тумачење квантитативни за предложене атрибуте квалитета. Детаљно израчунавање сваког од атрибута је представљено у потпоглављу В.3.

**Степен сложености одржавања софтвера**

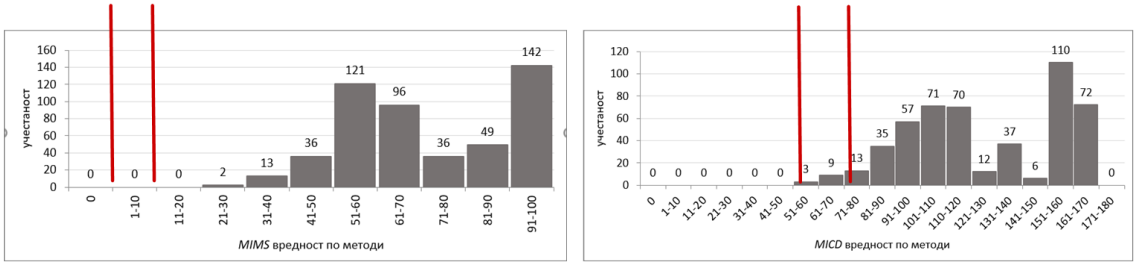
Сложеност одржавања се посматра преко степена сложености одржавања који је израчунат на основу резултата које је дао CM алат. Слика испод приказује вредности овога индекса у облику хистограма на нивоу метода.




*Слика 6.7 Степен сложености одржавања - MI*

Хистограми алтернативних метрика са границама предложеним од стране аутора имају исту расподелу као и оригинални степен сложености одржавања, само

су вредности другачије. Упоређујући са границама од стране аутора, већина метода потпада под критеријум прихватљивог и високо одрживог кода.




*Слика 6.8 Степен сложености одржавања - MIMS и MICD*

**Портабилност**

Вредност портабилности се мери за горњи део средњег слоја, без укључивања реализације самог слоја апстракције за дату платформу. Делови слоја апстракције за дату платформу су портабилни, а делови који се односе на специфичности саме платформе (нпр. добављање слике са камере) су специфични само за дату платформу, нису портабилни и није очекивано да буду портабилни. За разлику од њих горњи део средњег слоја не садржи делове који су специфични за платформу и самим тим се квалификује за одређивање степена портабилности. Вредности атрибута од којих зависи портабилност и њихово израчунавање су образложене у Додатку В.

Добијене су вредности портабилности у најбољем (вредност 1.28) и најгорем случају (вредност 1.95). Две вредности су последица чињенице да за неке од вредности потребних за израчунавање податрибута није јасно дефинисано на шта се односе, па је нпр. за спрегнутост узет у обзир најбољи и најгори случај. Закључак је да је степен портабилности на високом нивоу - између доброг и врло доброг.

**Проширивост**

На сличан начин као и код портабилности на основу вредности податрибута добијамо најбољу и најгору вредност за степен комплексности и за проширивост. Вредности степена проширивости су 1.88 и 2.5. Закључак је да је степен проширивости на добром нивоу – у опсегу између прихватљивог и веома доброг.

**Дискусија**

Увидом у резултате три представљена атрибута квалитета софтвера, може се закључити да су ови атрибути на задовољавајућем нивоу. Задовољавајућа вредност сложености одржавања софтвера подразумева задовољавајуће вредности особина које укључује овај атрибут, међу којима су и модуларност, поновно коришћење, тестабилност. Такође, модуларност је квантитативно израчуната као податрибут за рачунање проширивост и портабилности – вредност 1, што представља одличну вредност. Овим се верификују *Захтеви 21.-26.*

## 6.4. ПОРЕЂЕЊЕ СА ДРУГИМ РЕШЕЊИМА

На тржишту постоје разна решења која теже да реше један или више текућих проблема са којима се аутомобилска индустрија сусреће у транзицији на централну рачунарску платформу. Тренутно не постоји једно решење које решава све проблеме. Произвођачи аутомобила покушавају да направе свеобухватне платформе за развој апликација возила следеће генерације.

У овом потпоглављу је дат преглед решења по групама (њихове основне предности и мане, као и разлике и уклопљивост са AMB средњим слојем). Фокус се ставља на развој апликација за системе помоћи возачу и интеграцију са другим блиско-повезаним доменима, као и концепт софтверске платформе која се извршава на више хардверских платформи.

Табела 6.6 даје преглед постојећих решења (академских и комерцијалних) по критеријумима који су дефинисани у Поглављу 4. Табела је попуњена на основу информација које су понате аутору и до којих се дошло претраживањем интернета. Аутору овог рада приступ већини комерцијалних платформи није доступан.

Решења се могу поделити у више група. Што се тиче истраживачког рада, у домену система за помоћ возачу у модерним возилима, и у домену средњих слојева који се могу сврстати у сличну категорију као и AMB средњи слој, истичу се две веће групе: средњи слојеви засновани на технологијама потрошачке електронике (машинска визија и фузија сензора) и средњи слојеви засновани на проточној обради података.

Велики број решења која се налазе на тржишту потичу од самих произвођача чипова (који покушавају да подрже широк спектар софтверских решења), или су везани за одређени оперативни систем. Затим ту су и решења која се баве повезивањем више система и коришћењем услуга других система. На крају је дат приказ и поређење са пар концептуалних решења која су примењива на концепт централних платформи.

| Решења | К1 | К2.1 | К2.2 | К2.3 | К2.4 | К2.5 | К3 | К4 | К5 | К6 | К7 | К8 | К9 | К10 | К11 | К12 | К13 | К14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *Trivedi* [159] | с | - | - | - | - | - | д | - | - | - | д | с | с | - | д | - | - | - |
| *Lu* [160] | д | д | - | - | - | д | с | - | - | х | д | с | - | - | - | - | - | - |
| *Satzoda* [152] | д | д | - | - | - | - | с | с | - | - | д | - | с | - | с | - | - | - |
| *Moris* [161] | - | - | - | - | - | - | с | с | - | - | с | - | - | с | с | - | - | - |
| *Omerovic* [162] | д | д | - | - | - | - | - | с | - | - | д | - | с | д | - | - | - | - |
| *Marina* [121] | д | д | д | д | - | - | с | | - | х | д | - | с | - | с | с | - | - |
| *Yoo* [116] | д | д | - | - | - | с | - | - | - | х | д | д | с | с | - | с | - | - |
| *Cochlovius* [117] | д | д | с | - | - | - | - | - | - | х | д | - | - | - | - | - | - | - |
| *Hammond* [163] | д | д | д | с | - | - | - | - | - | р | д | с | - | - | - | - | - | - |
| *Jindal* [164] | д | д | д | д | - | - | - | - | - | х | д | д | д | - | - | - | - | с |
| *NVIDIA* [95] | д | д | д | д | - | - | д | с | д | х | д | д | д | - | д | д | д | с |
| *Snapdragon* [165] | д | д | д | д | - | - | д | с | д | д | д | с | д | - | д | д | - | д |
| *Matt* [166] | д | д | с | д | д | д | - | - | д | д | д | - | д | - | - | д | д | - |
| *TI* [96] | д | д | д | д | - | - | д | с | д | д | д | д | с | - | д | с | - | д |
| *Covesa* [99] | д | д | - | - | - | - | д | с | с | х | с | д | д | с | д | - | с | - |
| *AAOS* [167] | д | д | - | - | д | - | - | с | - | х | с | д | с | - | д | - | д | д |
| *Linux AGL* [98] | д | д | с | - | д | - | - | с | - | х | с | д | д | с | д | - | д | - |
| *QNX Car* [90] | д | д | с | - | с | - | с | с | - | х | с | с | д | - | - | - | д | - |
| *QNX ADAS* [84] | д | д | с | - | д | - | д | с | - | х | д | с | д | - | д | д | с | с |
| *Microsar* [87] | д | д | - | - | с | - | с | д | - | р | д | д | - | - | с | д | д | д |
| *GreenHils* [94] | д | д | с | - | д | - | с | с | - | р | д | с | - | - | д | д | д | д |
| *VSTAR* [85] | д | д | - | - | с | - | с | д | - | р | д | д | - | - | с | д | д | д |
| *EB Tresos* [86] | д | д | - | - | с | - | с | д | - | р | д | д | - | - | с | д | д | д |
| *EB Corbos* [89] | д | д | с | - | - | - | с | д | - | х | д | д | д | с | д | д | д | д |
| *Volker* [168] | д | д | - | - | - | - | - | с | д | д | д | - | - | с | - | - | - | - |
| *Iorio* [120] | д | с | - | - | - | - | - | с | д | д | д | - | - | с | - | - | д | - |
| *AutoWare* [102] | с | - | - | - | - | - | с | с | - | х | - | с | д | - | д | с | с | д |
| *Kato* [169] | д | д | - | - | - | - | с | с | - | х | - | с | д | - | д | с | с | д |
| *Connext* [93] | д | д | - | - | - | с | с | д | д | д | д | - | д | д | д | д | д | д |
| *MotionWise* [88] | д | д | с | с | д | с | с | д | д | д | д | д | с | - | д | д | с | д |
| *OVERSEE* [103] | с | - | - | - | - | - | с | д | д | д | - | с | - | - | - | - | д | - |
| *OpenRobinos* [170] | д | д | - | - | - | - | д | с | - | х | д | д | д | д | с | д | - | д |
| *Mundhenk* [114] | д | с | - | - | - | - | д | с | д | х | с | с | - | - | с | с | с | - |
| *Berger* [115] | с | - | - | - | - | - | д | с | с | х | - | - | с | - | с | - | - | с |
| *Kampmann* [113] | д | д | - | - | - | - | д | с | д | д | д | д | с | - | - | с | - | д |
| *Chaaban* [111] | с | - | - | - | - | с | с | д | д | д | с | д | - | - | д | с | - | - |
| *Becker* [109]/ *Mundhenk* [108] | д | с | с | - | - | с | д | д | д | р | д | д | с | - | д | с | с | д |
| *AMB* | д | д | д | д | д | д | д | д | д | д | д | д | д | д | д | с | с | с |

К* : Критеријуми поређења детаљно су описани у Табели 4.7.

К1 наменска платформа
К2* хетерогени чипови
К3 сензори и актуатори
К4 магистрале
К5 ОС независност?
К6 мера ОС апстракција
К7 извршавање у реалном времену
К8 руковање апликацијама
К9 коришћење спрега/библиотека
К10 повезивање са другим доменима
К11 помоћне функције
К12 безбедносни аспекти
К13 сигурносни аспекти
К14 алати

Опције испуњавања критеријума:

д : да, испуњава/обезбеђује/омогућава
с : специфично решење/делимично доступно
х : само за ОС високог нивоа (само за К6)
р : само за ОС за рад у реалном времену (само за К6)
- : не испуњава/не обезбеђује/нема информација

*Табела 6.6 Преглед решења по критеријумима*

*Решења заснована на технологијама потрошачке електронике*

Рачунарска визија је добро познато мулти-дисциплинарно поље и постоје разна истраживања везана за разне аспекте АДАС апликација заснованих на сликама и/или видеу [171]. Велики део истраживања се фокусира на робустне алгоритме [172], [173], који се развијају и верификују на персоналном рачунару. Ови алгоритми и средњи слојеви [159] углавном се не евалуирају на наменској платформи за рад у реалном времену. Исто се односи и на фузију сензора [174].

Поред ових алгоритама/средњих слојева за персонални рачунар, постоје и средњи слојеви за рачунарску визију за наменске платформе. Постојећи средњи слојеви [160], [152] су углавном засновани на мобилним платформама, посвећени једном специфичном алгоритму и не подржавају све аспекте потребне за лакши развој апликација (проточна обрада, апстракција улаза, излаза и разних обрадних јединица, интеграција са остатком окружења у возилу, итд.).

Важност интеграције АДАС и инфо-забавног домена је препозната у радовима [161] и [162], али не пружа се универзално решење. Аутори [161] дискутују визуелно представљање АДАС сензора и опције приказа у смислу ометања возача и транспарентности, док се у [162] представља једно решење интеграције са инфо-забавним системом, али се ограничава на Андроид инфо-забавни систем.

*Решења заснована на проточној обради података*

Постоје бројни приступи за алгоритме за обраду слике, од чега се посебно издвајају они засновани на графовима тока података – због природе алгоритама за обраду слике, који су вођени подацима. Приступи засновани на графовима могу лако да се користе да се реализује механизам тока података. Један приступ заснован на графовима за развој и распоређивање АДАС апликација заснованих на визији, на хетерогену вишејезгарну платформу је представљен у [163]. Како би доказали свој приступ, аутори су развили и распоредили једну апликацију.

Произвођачи хетерогених платформи пружају и платформска развојна окружења, која поред чињенице да су уско повезана за чип, пружају и механизме проточне обраде података [96], [164]. Овим се омогућава корисницима да се креирају токови обраде података од камера, преко разних обрадних блокова до

самог приказа [96], или да се омогући приступ разним јединицама за обраду за распоређивање задатака, као и за управљање меморијом и потрошњом [164].

Ови средњи слојеви нуде приступ за ефикасну обраду слике, али углавном не разматрају интеграцију са остатком окружења у возилу, апстракције улаза и излаза, као ни портабилност на друге платформе.

*Решења специфична за чип*

Поред поменутих платформских окружења за проточну обраду података, произвођачи чипова пружају и обухватнија решења [95], [165], [166], [96] која укључују већи скуп функција за боље искоришћење самог чипа (наменски оперативни систем, сигурносне функције покретања софтвера, сигурно ажурирање преко мреже, наменске библиотеке за приступ хардверским акцелераторима и слично). Међутим, ова решења су везана за одређени чип и/или језгро и не могу се користити на другим чиповима и/или језгрима за које нису предвиђена, док AMB средњи слој кроз физички слој подржава извршавање на разним чиповима. Предложено решење у овом раду омогућава и охрабрује коришћење платформских окружења на одређеном чипу, као комплемент AMB средњем слоју, јер садрже функционалности које су од стране произвођача чипа на ефикасан начин реализоване за дати чип.

*Решења везана за оперативни систем*

Већи број инфо-забавних решења средњих слојева ( [99], [175], [167], [98], [90]) који нуде разне спреге за развој апликација је доступан за извршавање над циљним оперативним системом који је или је засновани на Линукс, Андроид или *QNX* оперативном систему. Ова решења нуде разне спреге за развој апликација од мултимедијалних, преко корисничког интерфејса возила, до добављања основних информација са магистрала у возилу као што је брзина возила. Такође се фокусирају и на функције умрежености са другим возилима. Спрега са АДАС доменом није у фокусу, не ставља се пуно пажње на приступ резултатима АДАС апликација. Ова решења не адресирају приступе ресурсима као што су додатне обрадне јединице (доступни су процесорски ресурси који су иначе доступни преко постојећих широко распрострањених спрега/библиотека). AMB средњи слој може

да се искористи да повеже ова два домена, као и да пружи додатне ресурсе за обраду уколико то чип омогућава.

За АДАС домен доступна је платформа на *QNX* оперативном систему, која пружа разне спреге везане за сензоре, мрежу, визуелизацију. Интегрише такође библиотеке отвореног кода. Слично као и за инфо-забавне системе AMB средњи слој може да се искористи да унапреди приступ ка обрадним јединицама - брже искоришћење, а истовремено доноси повезаност са осталим платформама. QNX платформа за АДАС, као и AMB средњи слој омогућава рад са разним обрадним јединицама. Код ове платформе из апликација је потребно користити платформске спреге које одређени чип нуди за приступ овим обрадним јединицама, за разлику од AMB средњег слоја који пружа апстракције. Такође ово је комерцијална платформа, и као таква није доступна свима за брзи иницијални развој алгоритама (енгл. *prototyping*). Суживот AMB предложеног слоја са овим платформама је могућ, тј. AMB је могуће користити као софтверски слој који пружа додатне функционалности овим платформама.

Такође, у овој групи се налазе решења која имплементирају AUTOSAR стандард. Имплементација класичног AUTOSAR-а је представљена у [87], [94], [85], [86] и ова решења су намењена за безбедносно-критичне микроконтролере, не узимајући у обзир хетерогене чипове. Адаптивни AUTOSAR је имплементиран у решењима [89], уводи рад са платформама високих перформанси, дефинише руковање апликацијама, извршавање на оперативном систему високог нивоа и могућност рада са широко распрострањеним библиотекама и спрегама, али не пружа апстракције ка хетерогеним чиповима.

*Решења за комуникацију између сервиса*

Решење које се најчешће помиње као комуникациона спрега преко Етернет магистрале је *SOME/IP* [168]. Може се користити за размену управљачких порука. Дизајниран је да може да подржи различите уређаје и разне оперативне системе (од AUTOSAR уређаја, до инфо-забавних и уређаја за удаљену комуникацију). Ово решење се користи такође у AUTOSAR стандарду (класичном и адаптивном). Ово решење представља само комуникациону спрегу и не нуди друге функционалности за лакши развој апликација. AMB средњи слој не користи ову спрегу као

комуникациони канал преко етернета, али не спречава корисника да исти користи у оквиру јединица обраде.

РОС [101] је средњи слој отвореног кода који оригинално потиче из роботског развојног окружења. Заснован је на механизму објављивања и претплаћивања (енгл. *publish/subscribe*) који омогућава размену података између РОС процеса. Базира се на графовима, где се дефинишу чворови и везе између њих које представљају комуникационе механизме. Чворови могу бити у оквиру истог процеса, у различитим процесима или на различитим уређајима. Као такав представља механизам за развој сложенијих апликација. Један од пројеката отвореног кода за аутономну вожњу који је заснован на РОС-у је *AutoWare* [102]. Пружа модуле који омогућавају реализацију сценарија аутономне вожње: добављање података са сензора, локализација, детекција објеката, предикција, планирање. Дизајниран је за аутономну вожњу у урбаним срединама. Реализација других корисничких сценарија, као што су вожња ауто-путем, може захтевати реализацију додатних модула. Иницијално је развијен на персоналним рачунарима, један пример прилагођења за наменску платформу дат је у [169], где аутори закључују да планирају да унапреде дизајн и реализацију овог средњег слоја за рад у реалном времену.

Комерцијална платформа *ConnextDrive* [93] пружа софтверско окружење за развој и интеграцију апликација аутономне вожње. Заснива се на ДДС стандарду који такође ради на принципу објављивања/претплаћивања. Основни циљ платформе је повезивање различитих технологија омогућавајући аутомобилским компанијама да раде са стандардима/технологијама које најбоље задовољавају њихове потребе у различитим тачкама развојног циклуса. Платформа реализује виртуалну магистралу за повезивање разних екосистема (адаптивни АУТОСАР, класични АУТОСАР, системи обраде слике, системи за симулацију). Подржава комуникацију преко дељене меморије и етернета.

Још једна комерцијална платформа *MotionWise* [88] фокус ставља на комуникацију између различитих домена (класични и адаптивни АУТОСАР, разне обрадне јединице). Подржава различите оперативне системе, а заснива се на детерминистичком распоређивању и детерминистичкој комуникацији између чипова преко Етернет магистрале.

Поменута решења имају разрађеније комуникационе механизме у односу на AMB средњи слој, али не пружају униформне спреге ка процесорским ресурсима. У зависности од платформе онај ко развија алгоритам принуђен је да учи спреге за сваку платформу посебно.

*Решења у развоју намењена за централне платформе*

Одређени број решења адресира концепт централне платформе представљајући софтверску платформу са или без хардверске платформе. Ове софтверске платформе су представљене у форми концепта или иницијалне реализације. Пружају скуп сервиса (распоређивање, пренос података, синхронизација) који се заснивају на механизмима клијент/сервер. Такође пружају спреге са сензорима и актуаторима, али се спрега са њима као и комуникациони механизми углавном заснивају искључиво на етернет вези [112], [108]. Решења углавном адресирају безбедносне аспекте, док су сигурносни аспекти слабије адресирани.

Хардверске платформе над којима се решења заснивају су засноване углавном на више хардверских јединица (микроконтролер и јединица за обраду – наменска платформа или персонални рачунар [111], [112]) или на централној платформи са вишеструким обрадним јединицама [108]. Решења не пружају апстракције хардвера – хетерогених платформи (у смислу апстракције процесорских јединица). Аутори нису представили перформансе рада у реалном времену.

## 6.5. ДОДАТНА РАЗМАТРАЊА

### 6.5.1. Компатибилност са нефункционалним захтевима

*Компатибилност са постојећим библиотекама и спрегама*

Данас су доступне разне библиотеке отвореног кода које користе хетерогену архитектуру чипова и оптимизоване су за њих. Неке од најраспрострањенијих су *OpenGL|ES* [176], *OpenGL SC* [177] за графичке операције, *OpenCL* [178] за паралелну обраду, као и средњи слојеви који су засновани на њему [179], *OpenCV* [180] за компјутерску визију, *OpenVX* [181] за компјутерску визију, као и оптимизације на нивоу компајлера *OpenMP* за *CPU*, и *OpenACC* за *CPU/GPU*

системе. Спрега *SOME/IP* [168] се користи као спрега за размену управљачких података.

Све ове библиотеке и спреге наглашавају специфичне аспекте и не обухватају све аспекте важне за развој АДАС апликација. Средњи слој предложен у овом раду подржава коришћење библиотека отвореног кода у градивним блоковима апликација, и као додатак пружа и лакшу интеграцију са инфо-забавним системима и сензорима, апстрахује специфичности чипова и реализује механизам за проток података.

*Безбедносни аспекти*

Тренутна иницијална реализација AMB средњег слоја оставља интегратору система да задовољи крајње безбедносне захтеве у односу на целокупни систем. Решење је реализовано као *QM* решење. Примери реализације се могу користити у тестне сврхе, у сврхе поређења платформи ради избора циљне платформе и слично.

За укључивање предложеног средњег слоја у возила, на интегратору је да приликом интеграције у циљни систем одреди безбедносне нивое, провери како утиче на AMB средњи слој и обезбеди додатне механизме надгледања и изолацију од безбедносно-критичних функција у возилу, који ће задовољити функционални безбедносни концепт на нивоу система.

На пример, уобичајени захтев за чип високих перформанси који се користи за захтевне обраде је строго раздвајање специфичних софтверских домена. Увођење хипервизора омогућава испуњавање безбедносних и сигурносних захтева, тако да се у таквом окружењу, софтверске функције различитих *ASIL* нивоа могу лако раздвојити. Хипервизор омогућава одвајање како малих апликација (нпр. за надгледање), тако и целокупних оперативних система са припадајућим руковаоцима у виртуелним машинама. Са становишта AMB средњег слоја, обезбеђен је слој апстракције платформе, у коме се реализују специфичности за саму платформу (са или без хипервизорског слоја), што чини AMB повезани слој независним од платформе.

У аутомобилској индустрији, безбедносни елемент ван контекста (енгл. *Safety Element out of Context - SEooC*) дефинисан у ИСО26262-10 је метод за коришћење компоненти у возилу које нису првобитно дизајниране за специфични пројекат.

Елементи ван контекста су дизајнирани да обезбеде одређену функционалност без свести о томе како ће се она користити у циљном систему.

AMB средњи слој је осмишљен као безбедносни елемент ван контекста. Један од праваца будућег рада је управо проширење решења тако да задовољава одређене *ASIL* захтеве. Потребно је дефинисати безбедносне циљеве, претпоставке и механизме. На пример, ако претпоставимо да је безбедносни циљ *ASIL* комуникација, дефинисаће се механизми који гарантују поуздан пренос порука, тј. софтверски слој за пренос треба да гарантује испоруку порука одређеног квалитета (нпр. да се очува интегритет и редослед порука или да се ниједна порука не пошаље два пута).

*Сигурносни аспекти*

Тренутна реализација AMB средњег слоја оставља интегратору система да задовољи крајње сигурносне захтеве у односу на целокупни систем.

Ако је потребно ажурирати апликацију која користи AMB спреге уз одређене сигурносне механизме, њено ажурирање се врши на исти начин као што је предвиђено и за остале апликације у систему. На пример, ако узмемо случај апликације адаптивног Аутосара, која користи AMB спреге, ажурирање апликација се врши преко групе функција *UCM* модула (енгл. *Update and Configuration Management*). Ова група функција је задужена за ажурирање, инсталирање и уклањање софтвера, са посебним освртом на сигурносне и безбедносне аспекте. *UCM* модул користи *Crypto* спрегу адаптивног Аутосара како би верификовао интегритет и аутентичност пакета и декриптовао податке.

Интегратор употребом хипервизора и раздвајањем софтверских домена такође може побољшати сигурност и сигурну комуникацију ка интернету и бекенд системима.

### 6.5.2. Време развоја и интеграције

Употребом AMB средњег слоја, развој апликација је олакшан, време интеграције се смањује, али и даље је потребно уложити време за реализацију слоја за апстракцију на циљној платформи, пре развоја апликација преко AMB спрега. У

зависности од крајњег циља, време развоја апликација са AMB средњим слојем се може смањити (развој више апликација преко предложеног средњег слоја на једној или више платформи) или чак повећати (нпр. развој једне мање захтевне апликације на једној платформи).

Након реализације слоја за апстракцију, процес интеграције разних апликација на чипу је бржи, као и мапирање делова апликације на сама језгра. AMB средњи слој омогућава лакше пребацивање апликација са персоналног рачунара на циљну платформу, као и лакше пребацивање делова обраде са једног на друго језгро, у сврху бољег балансирања оптерећења језгара.

У овом поглављу, показано је да је предложени средњи слој адресирао изазове и циљеве који доприносе бржем и једноставнијем развоју и интеграцији апликација. Једноставније коришћење хетерогене природе чипова, као и портабилност апликација су адресирани кроз  униформне спреге и апстракције, као и раздвајање апликативног софтвера од хардвера. Интероперабилност (способност различитих делова система да се лако повежу и размењују податке) и проширивост (функцијама, додатним хардверским блоковима) су верификовани експериментално и преко софтверских метрика. При реализацији референтног система  апликација који представља реалан кориснички случај демонстрирано је коришћење стандардизованих библиотека и спрега. Рани развој на циљној платформи је адресиран кроз помоћне функције/симулаторе и кроз могућност развоја на персоналном рачунару користећи исте спреге и дајући могућност брзог преноса апликација или њихових делова на циљну платформу

# 7. ЗАКЉУЧАК

У оквиру дисертације је предложена архитектура средњег слоја софтвера са циљем да се обезбеди софтверска платформа као први корак у преласку на једну или више централних платформи, што доноси бржи и једноставнији развој апликација у окружењима са присуством хетерогених платформи. Такође је извршена верификација решења. Решење је дефинисано кроз скуп модула и спрега за развој апликација, водећи рачуна о платформској независности.

Током истраживања извршена је анализа постојећих решења, потреба и захтева тржишта и произвођача, и доступних технологија у циљу утврђивања постојећег стања. Анализа показује да тренутно не постоји јединствено свеобухватно решење које омогућава развој апликација на централним интеграционим платформама уважавајући све захтеве возила нове генерације. Током процеса анализе препозната је структура решења по питању очекиване сложености средњег слоја и скупа функционалности од интереса које ће допринети бржој транзицији на централну интеграциону платформу. Основни реализовани циљеви су омогућавање развоја апликација независно од саме платформе/језгра на коме се извршавају, омогућавање на једноставан начин спајања са другим компонентама и омогућавање раног развоја апликација на циљној платформи, уместо на персоналном рачунару.

За примере реализације одабран је скуп од три циљне платформе који покрива различите домене и корисничке случајеве демонстрирајући независност решења од саме хардверске платформе (инфо-забавни домен, домен за системе помоћи возачу, и развојни рачунар (у случају развоја алгоритма и бржег пребацивања на циљну платформу)).

Верификација је извршена користећи метрике квалитета реализованог решења: ХИС метрике (као стандардизоване метрике за аутомобилски софтвер), објектно-оријентисане метрике, и метрике за атрибуте квалитета софтвера (сложеност одржавања софтвера, портабилност, проширивост). Предложена спрега је верификована такође и експерименталним путем у лабораторијским условима.

Примери реализације су претходних година приказани на светским сајмовима потрошачке електронике и аутомобилске индустрије: *CES* у Лас Вегасу, *Embedded World* у Нирнбергу, *ELIV* у Бону, и представљају потврду дисертације.

Решење из ове докторске дисертације се такође може користити за поређење и одабир платформи при развоју апликација. Будући кораци у развоју средњег слоја могу тежити ка корисничким случајевима који укључују спреге са безбедносно-критичним системима, комуникацију између возила и/или инфраструктуре (енгл. *V2X*), као и ка проширењу слоја за апстракцију платформе подршком за више чипова (нпр. *nVidia* чиповима).

# ДОДАТАК А: ПРЕГЛЕД ПОСТОЈЕЋИХ РЕШЕЊА И ИНИЦИЈАТИВА

## А.1 ИНИЦИЈАТИВЕ ЗА СТАНДАРДИЗАЦИЈУ

Иницијативе за стандардизацију одређених сегмената постоје у целом свету. Најпознатије су излистане у табели испод са описом главних циљева. Такође су означене функционалности којима се баве: контроле (управљање вратима, прозорима, мотором, кочницом, скретањем, итд.), информације (навигација, пређена дистанца, аудио, мобилни уређаји и сл.) и комуникација (уграђени комуникациони системи у возилу).

| Име | Контроле | Информације | Комуникација | Циљеви |
|---|---|---|---|---|
| *AUTOSAR* [81] | x | x | x | Развој софтверског стандарда који повећава поновно коришћење постојећег софтвера. Циљ је да допринесе развоју комплексних и нових *ECU*-ова на ефикасан начин уз одговарајући квалитет. |
| *FlexRay конзорцијум* | | | x | Развој комуникационих стандарда за магистрале велике брзине за системе управљања возилом. Успостављање магистрале (10*Mbps)* која је десет пута бржа од *CAN* магистрале и има већу поузданост. |
| *OPEN Alliance* [182] | | | x | Развој стандарда који ће применити Етернет технологије (које су се показале као добре у персоналним рачунарима) на системе у возилима. Коришћење мреже у возилу чија изградња није скупа. |
| *CE4A* [183] | | x | | Развој стандарда за спрегу између мобилних уређаја (мобилни телефони, навигациони системи, музички плејери итд.) и система у возилу. |
| *CCC* [184] | | x | | Развој стандарда за повезивање паметних телефона са системима у возилу. |

| Име | Контроле | Информације | Комуникација | Циљеви |
|---|---|---|---|---|
| *GENIVI* [99] | | x | | Повећање ефикасности развоја инфо-забавних технологија. Развој отворене платформе засноване на Линукс оперативном систему који се користи и у другим информационим уређајима. |
| *JASPAR* [83] | x | x | x | Повећање ефикасности развоја и поузданости све комплексније система *ECU*-ова у возилу, кроз стандардизацију и заједничко коришћење софтвера и магистрала. |

*Табела А.1 Преглед иницијатива за стандардизацију*

АУТОСАР стандард (класични) се односи на статичку, унапред конфигурисану и самим тим ограничену платформу. Углавном се користи на микроконтролерима где је потребно обезбедити веће *ASIL* нивое. Постоји више реализација класичног АУТОСАР стандарда. Произвођачи углавном сем самог софтвера за наменску платформу, пружају и алате за дизајнирање, конфигурацију, генерисање и интеграцију софтвера, као и додатне модуле који представљају проширења. У табели испод је дат упоредни приказ неких од решења реализације овог стандарда.

| **Име** | **АУТОСАР верзија** | ***ASIL* ниво** | **Алати** | ***BSW*** | ***OS*** | **Виртуална платформа** | **Додатно** |
|---|---|---|---|---|---|---|---|
| *MICROSAR [59]* | 3.x, 4.x | *ASIL D* | *DaVinci Configurator DaVinci Developer* | *MICROSAR.BSW MICROSAR.RTE* | *MICROSAR.OS* (проширени *OSEK/VDX-OS*) | да | *MICROSAR. OTA* |
| *Capital VSTAR* [85] | 4.x | *ASIL D* | *Capital VSTAR Integrator* | *Capital VSTAR RTE* | *Capital VSTAR OS* | нп* | |
| *EB Tresos* [86] | 3.x, 4.x | *ASIL D* | *EB tresos Studio* | *EB tresos AutoCore* | *EB tresos AutoCore OS EB tresos Safety OS* | нп* | *EB tresos V2G ChargeIn* |
| *ETAS Autosar* [185] | 4.x | *ASIL D* | *ISOLAR-A/B/EVE ASCET-DEVELOPER* | *RTA-BSW* | нп* | да | |
| *NeuSAR* [186] | 4.2 | *ASIL D* | *cCore Configurator* | *cCore* | нп* | нп* | |
| *HiQuanten* [187] | 4.0 | нп* | *HiQuanten.Geno HiQuanten.SIM* | *HiQuanten.<модул>* | *HiQuanten.OS (OSEK/VDX-OS)* | нп* | |
| *KSAR* [188] | 4.x | *ASIL D* | *C4K Classic* | *KSAR Classic BSW* | *KSAR OS* | нп* | |

нп* - није познато, информација није доступна

*Табела А.2 Преглед реализација АУТОСАР класичне платформе*

Аутосар адаптивна платформа се односи на функције захтеване у возилима нове генерације, у окружењу са већом процесном моћи и већим протоком података, где је потребно обезбедити динамички развој нових функционалности, интеракцију са не-АУТОСАР апликацијама, и ажурирање софтвера преко мреже. Као и за класични АУТОСАР, и овде постоји више реализација, а неке су приказане упоредо у табели испод. За разлику од класичног АУТОСАР-а, основна решења су недавно развијена и и даље су у развоју.

| **Име** | ***ASIL* ниво** | **Алати** | ***Софтверско решење*** | ***Развијено на платформи*** | **Додатно** |
|---|---|---|---|---|---|
| *NeuSAR* [189] | нп* | *aCore Configurator* | *aCore adARA* | *Linux QNX NeuSar hypervisor* | |
| *MICROSAR Adaptive* [190] | *ASIL D* | *DaVinci Developer Adaptive PREEvision* | *MICROSAR ARA* | *Linux QNX PikeOS* | |
| *WindRiver Autosar Adaptive* [191] | *ASIL D* | нп* | нп* | *VxWorks WindRiver Linux Helix Virtualization Platform* | |
| *EB corbos* [89] | *ASIL D* | *EB corbos Studio* | *EB corbos AdaptiveCore* | *EB corbos Linux EB corbos Hypervisor* | *AUTOSAR AP18-03* |
| *RTA-VRTE* [192] | нп* | *Adaptive Studio ISOLAR-A_ADAPTIVE* | *RTA-VRTE* | *Linux QNX* Виртуална платформа | У развоју |
| *AUBIST Adaptive Platform* [193] | нп* | *AUBIST Tool AP* | *AUBIST AP* | *AUBIST AP OS Linux* | Компатибилна са верзијом *R19-03* и надаље |

нп* - није познато, информација није доступна

*Табела А.3 Преглед реализација АУТОСАР адаптивне платформе*

## А.2 ДОСТУПНА РЕШЕЊА

Табела А.4 приказује доступна решења средњих слојева и даје опис главних особина.

| Назив | Инфо-забавни | АДАС | Облак | Опис |
|---|---|---|---|---|
| *COVESA* [99] | x | | | *COVESA* (енгл. *Connected Vehicle Systems Alliance*), раније *GENIVI* референтна платформа развијена је од стране *Genivi* алијансе, која укључује OEM-ове и добављаче. Ова платформа се фокусира на инфо-забавне апликације и сценарије умрежених возила. Састоји се од Линукс централних сервиса, средњег слоја и спреге апликативног слоја. У последње време усмерили су фокус на интеграцију више оперативних система за потребе централног умреженог кокпита. |
| *ARM SOAFEE* [166] | x | x | x | *ARM SOAFEE* (енгл. *Scalable Open Architecture for Embedded Edge*) представља унапређену архитектуру увезану са облаком за разне апликације различитих безбедносних нивоа. На постојеће пројекте који обезбеђују сигурно покретање над ARM архитектурама, *SOAFEE* доноси функционалну безбедност и функционалност извршавања аутомобилских апликација у реалном времену. *SOAFEE* користи *OCI* (енгл. *Open Container Initiative*) компатибилне контејнере. |
| *MotionWise* [194] | | x | | *MotionWise* представља безбедносну софтверску платформу за аутономну вожњу. Платформа се састоји од софтвера за наменску платформу, алата, прототипа хардвера. Компатибилна је са АУТОСАР стандардом. Подржава више оперативних система и заснива се на детерминистичком Етернету као вези између чипова. |
| *Automotive Grade Linux* [98] | x | | | *AGL* (енгл. *Automotive Grade Linux*) [98] представља пројекат отвореног кода који прави отворени оперативни систем и средњи слој за апликације у возилима. Кључни аспекти прве спецификације се односе на софтверску платформу засновану на Линуксу за инфо-забавне системе. Дефинише сервисе као што су WiFi, блутут, мултимедија, руковање животним циклусом апликација, дефинише нативни и *HTML5* средњи слој, као и повезаност и интеракцију са магистралама у возилу (*CAN, MOST*). |

| | | | | |
|---|---|---|---|---|
| *Android Automotive OS* [167] | x | | | *Android Automotive OS* (*AAOS*) представља инфо-забавну платформу, уграђену у возила. Корисници могу да скину апликације директно у возила без потребе за додатним уређајима као што су телефони, и да користе интерфејсе који су дизајнирани за екране у возилима. |
| *Open Robinos* [170] | | x | | *OpenRobinos* је отворена спецификација за функционалну софтверску архитектуру са добро дефинисаним спрегама, софтверским модулима и управљачким механизмима. У току је процес спајања са SOFAM иницијативом (енгл. *Standardized, open framework for autonomous mobility*). Главни циљ је стварање референтне платформе за аутоматизовану вожњу. |
| *Open Fusion Platform* [195] | | x | | *Open Fusion Platform* пројекат се бави развојем платформе за функције аутономне вожње са нагласком на опажање и фузију података, као и отворене спреге. У питању је референтна платформа, са циљем да се њене могућности покажу кроз један главни циљни кориснички случај: електрични аутомобил се аутономно паркира и позиционира прво на паркинг место са механизмом за бежично пуњење, а затим, након што се аутомобил потпуно напуни, вози се и позиционира на нормално паркинг место без механизма за пуњење. |
| *ROS2* [101] | | x | | *ROS2* (енгл. *Robot Operating System*) представља флексибилни средњи слој за писање софтвера за роботе. Састоји се од скупа алата, библиотека и конвенција које имају за циљ да олакшају стварање сложеног и робусног понашања на разним роботским платформама. Све више се помиње у контексту аутомобилске индустрије, и служи као база за прављење софтверских платформи углавном за напредне системе помоћи возачу. |
| *AutoWare* [102] | | x | | *AutoWare* је софтверски пројекат отвореног кода који пружа скуп модула за самостално управљање возилом укључујући локализацију, детекцију, предвиђање, планирање и контролу. Заснива се на *ROS* средњем слоју. |
| *OVERSEE* [103] | x | | | *OVERSEE* (енгл. *Open Vehicular Secure Platform*) пројекат се фокусира на сигурносне проблеме које доносе отворене платформе и сигурносну интеграцију широког спектра комуникационих канала. Омогућава развој апликација за возила и понуду ових апликација преко продавнице апликација. Осигурава да апликације не могу наудити једна другој као ни било ком другом ИТ систему у возилима. |
| *Processor SDK TDAx* [96] | | x | | *Processor SDK TDAx* представља развојно окружење за ТДА фамилију чипова за АДАС домен. Окружење пружа кориснику могућност креирања различитих АДАС токова података који |

| | | | | |
|---|---|---|---|---|
| | | | | укључују прихват слика са камера, пред-обраду слике, разне алгоритме за анализу слике и приказ слике. Састоји се од заједничког слоја који представља корисничку спрегу за све процесоре на чипу, као и од наменских обрада за *DSP* и *EVE* процесоре. |
| *Snapdragon Ride SDK* [165] | | x | | *Snapdragon Ride SDK* представља развојно софтверско окружење које са одговарајућим *Snapdragon Ride* хардвером чини *Snapdragon Ride Autonomous Stack.* Платформско окружење омогућава развој прилагођених АДАС апликација. Корисницима пружа модуле који апстрахују *Snapdragon* хардверске блокове, пружа руковаоце који имају добре индикаторе перформанси по питању кашњења, меморије и пропусног опсега. Такође подржава широк спектар спрега од спрега ка систему камера, дубоких неуралних мрежа, оптимизованих библиотека за АДАС алгоритме до алата за развој и тестирање. |
| *NVIDIA DRIVE OS* [95] | | | | *NVIDIA DRIVE OS* представља референтни оперативни систем са припадајућим софтверским проширењем које омогућава развој апликација за аутономна возила на *DRIVE AGX* хардверским платформама. Основни софтвер се састоји од наменском оперативног система за раду у реалном времену, хипервизора, *CUDA, TensorRT* и других компоненти намењених да пруже оптимизован приступ и извршавање на хардверским акцелераторима. Платформа пружа и сигурносне сервисе за покретање, као и ажурирање преко мреже. |
| *Automotive OPTStack Middleware* [175] | x | x | | *Connected OS* представља платформу за повезивање са мрежом у возилу као и са уређајима потрошачке електронике. Заснива се на модуларној платформи заснованој на *GENIVI* Линукс платформи, са унапређеним пакетом за руковање самом хардверском платформом, као и оптимизованим средњим софтверским слојем – *OPTStack*. Средњи слој софтвера пружа оптимизоване аудио, графичке и видео функционалности. |
| *Connext Drive* [93] | | x | x | *Connext Drive* представља софтверску платформу за аутономну вожњу. Развојно окружење за развој и интеграцију апликација у возила се заснива на *DDS* (енгл. *Data Distribution Service*) стандарду, и реализује повезивање засновано на подацима које нуди виртуалну дистрибуирану дељену меморију. Представља се као платформа која може повезати различите екосистеме у возилу (*DDS*, *ROS2*, *AUTOSAR* класични и адаптивни). |
| *QNX Car Platform* [90] | x | | | *QNX Car Platform* представља инфо-забавну платформу. Платформа интегрише *QNX* оперативни систем и средњи слој који пружа инфо-забавне функционалности као што су навигација, препознавање |

| | | | | |
|---|---|---|---|---|
| | | | | гласа, повезивање са паметним телефонима, ажурирање преко мреже, као и мултимедијални скуп функционалности и повезивање са корисничким интерфејсом возила. |
| *QNX Platform for ADAS* [84] | | x | | *QNX Platform for ADAS* представља софтверску платформу за развој и извршавање АДАС алгоритама. Платформа интегрише *QNX* оперативни систем и пружа разне сервисе потребне за развој АДАС апликација као и демо алгоритме. Од сервиса су присутни сервиси за сензоре, мрежу, визуелизацију података, интеграција са *ROS*-ом у сврхе тестирања прототипа, као и интегрисане библиотеке отвореног кода као што су: *OpenCV*, *SOME/IP* итд. |
| *Integrity* [94] | | | | *Integrity RTOS* представља оперативни систем у реалном времену који омогућава извршавање апликација на сигуран, поуздан и брз начин. Поред оперативног система у понуди је и одређени број средњих слојева који пружају следеће функционалности: систем датотека, етернет веза, *WiFi* сервиси, веб сервисе, 2Д и 3Д графика. |
| *EB connected vehicle* [196] | x | x | x | *EB connected vehicle* представља групу софтверских решења за увезана возила нове генерације. Пружа скуп флексибилних и скалабилних сервиса и софтверских компоненти за сигурно међусобно повезивање возила и омогућавање сервиса као што су удаљена дијагностика и ажурирање софтвера преко мреже. Подржава разне оперативне системе: АУТОСАР класични и адаптивни, Линукс или *QNX*. |
| *HERCULES Software Stack* [197] | | x | | Пројекат *HERCULES* предлаже технолошку инфраструктуру за возила нове генерације користећи најновије доступне хетерогене платформе. Предлаже такође интегрисано софтверско окружење које омогућава суживот више врста апликација (апликације које се извршавају у реалном времену и које не захтевају извршавање у реалном времену). Пројекат се не веже за специфични систем/чип али захтева постојање одређеног шаблона по питању система/чипа. |
| *V2V средњи слој* [198] | | x | x | Аутори представљају прошириви средњи слој који омогућава развој апликација за комуникацију између возила. Фокус је на бежичној комуникацији и систематској анализи за потребе решења заједничке интелигенције међу возилима и решења за анализу велике количине података. Средњи слој пружа спреге ка четири главна модула: управљачки делови возила, спољна бежична комуникација, вештачка интелигенција, унутрашња виртуална магистрала. |

| | | | | |
|---|---|---|---|---|
| *Дистрибуирани систем* [151] | | x | | Аутори представљају дистрибуирани наменски систем за рачунарску визију који омогућава анализу окружења возила и процену опасности од возила. Предложени систем користи камера сензоре као једине сензоре за анализирање окружења возила. Систем се фокусира на специфичну платформу (тј. више инстанци исте платформе рад повећања ресурса за обраду) и на специфичне алгоритме. |

*Табела А.4 Преглед решења средњих слојева*

## А.3 БИБЛИОТЕКЕ ОТВОРЕНОГ КОДА

Данас су доступне разне библиотеке отвореног кода које експлоатишу хетерогену природу чипа и које су оптимизоване за сами чип. Ове библиотеке се фокусирају на одређени аспект (нпр. *OpenGL* истиче графички аспект, *OpenVX* истиче аспекат обраде слике).

| Назив | Опис |
|---|---|
| *OpenGL\|ES\|SC* [176], [177] | *OpenGL\|ES* представља графички АПИ нижег слоја. Главна примена му је за интеракцију са графичком процесорском јединицом ради постизања веће брзине рендеровања користећи хардверске акцелераторе. *OpenGL\|SC* представља подскуп *OpenGL\|ES* – а, који је дизајниран да испуни захтеве за безбедносно критичне примене, укључујући и примене у аутомобилској индустрији. |
| *OpenCL* [178] | *OpenCL* представља АПИ нижег слоја за паралелну обраду у хетерогеним системима. Подржан је од стране многих хетерогених платформи. На њему је заснован и одређен број средњих слоја за обраду на хетерогеним платформама, на пример средњи слој из [179] за обраду на *CPU/GPU* у оквиру истог чипа. |
| *OpenCV* [180] | *OpenCV* представља библиотеку за рачунарску визију високог нивоа (визуелна анализа слике и виде записа). Фокусира се на апликације у реалном времену користећи предности вишејезгарних платформи. Библиотека није погодна за наменске системе, због чега произвођачи чипова нуде прилагођене верзије или подскуп ове библиотеке за њихове чипове. |
| *OpenVX* [181] | *OpenVX* представља средњи слој заснован на графовима. Намењен је за убрзавање извршавања апликација за рачунарску визију на различитим платформама. Заснива се на реализацијама рутина за обраду слика које |

| | |
|---|---|
| | дизајнирају произвођачи чипова како би што боље искористили архитектуру хардвера за убрзање обраде. Постоје разни истраживачки радови (као што су [199], [200]) који се баве оптимизацијом *OpenVX*-а у погледу убрзања протока података за апликације које се заснивају на обради слике. |
| *OpenMP* [201] | *OpenMP* припада групи спрега која врши оптимизацију на компајлерском нивоу. Представља спрегу која подржава паралелно програмирање за више платформи са дељеном меморијом. Састоји се од скупа компајлерских директива, рутина библиотека, и промењивих у окружењу које утичу на понашање током самог извршавања. |
| *OpenACC* [202] | *OpenACC* припада групи спрега која врши оптимизацију на компајлерском нивоу. Представља програмски стандард за паралелну обраду који су развили *Cray*, *CAPS*, *NVIDIA* и *PGI*. Стандард је дизајниран да поједностави паралелно програмирање хетерогених *CPU/GPU* система. Састоји се од скупа компајлерских директива које означавају петље и регионе у коду чије извршавање треба да се пребаци са главног *CPU*-а на доступне акцелераторе. |
| *SOME/IP* [168] | *SOME/IP* (енгл. *Scalable service-Oriented MiddlewarE over IP*) припада групи спрега која се односи на комуникациони домен. Дизајнирана је да одговара разним врстама уређаја, оперативних система и домена (укључујући и класични АУТОСАР стандард). Користи се за размену управљачких порука преко етернет магистрале. Реализује функционалности серијализације, спровођења удаљеног позивања функција, динамичко проналажење сервиса и тражених/објављених података, као и сегментацију *UDP* порука. |

*Табела А.5 Преглед библиотека*

## А.4 СВЕОБУХВАТНЕ ПЛАТФОРМЕ У РАЗВОЈУ

Већина произвођача аутомобила је и даље у фази надоградње хардверских платформи. Тренутно се у фокусу такође налази основни софтвер (језгра оперативног система, стандардизоване платформе, средњи слојеви итд.). Линукс, *QNX* и други РТОС-и обезбеђују само језгро. Над тим језгром, произвођачи аутомобила реализују апстракцију хардвера, формирају оперативни систем који подржава развој апликација, дефинише логику интеракције и гради апликативне слојеве, тзв. само-развојни оперативни системи (енгл. *self-developed operating systems*). Коначни циљ произвођача аутомобила је да се развој софтвера за возила отвори свима, поједностављењем самог развоја и повећањем фреквенције ажурирања софтвера. Све више произвођача аутомобила следе путању развоја свог само-развојног оперативног система, тј. свеобухватне платформе. У наставку је дат преглед неких од њих.

| Назив | Опис |
|---|---|
| *Tesla.OS* [203] | Компанија *Tesla* не даје пуно детаља о својој платформи, али основни слој свога решења заснива на модификованом Линукс систему отвореног кода, и не ослања се на друге добављаче софтвера. Једна од основних функционалности које су подржане је управо удаљено ажурирање софтвера. Акценат се такође ставља и на неуралне мреже. |
| *VW.OS* [204] | *VW.OS* платформа омогућава коришћење више оперативних система. Са једне стране омогућава се коришћење наменских функција у односу на изабрани оперативни систем, а са друге стране спреге нису потпуно стандардизоване. *VW.OS* се угледа на АУТОСАР платформу у циљу достизања стандардизације средњег слоја. Прва реализација очекује се 2024. године. |
| *MB.OS* [205] | Главни циљ *MB.OS* платформе је повезивање возила са облаком и ИоТ светом(енгл. *Internet of Things*), обухватајући четири домена: аутономна вожња, инфо-забавни системи, систем погона, и системи шасије и удобности. Оперативни систем је вођен подацима (енгл. *data-driven*) и омогућава флексибилност приликом ажурирања. Очекивани почетак коришћења је 2024. година. |
| *BMW.OS* [206] | Компанија *BMW* већ дужи низ година има софтвер који се зове оперативни систем. Са верзијом која је најављивана крајем 2021./почетком 2022., циља увођење ажурирања апликација и више могућности прилагођавања самог возила кориснику. Руковање *BMW.OS*-ом ће и даље бит подржано преко |

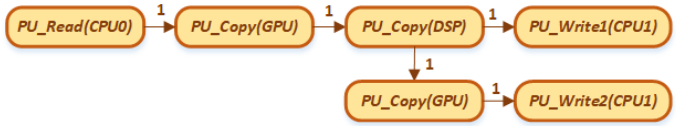
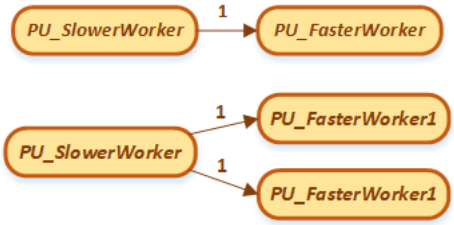
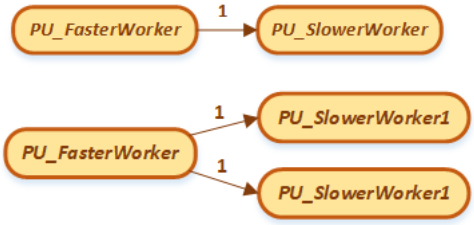
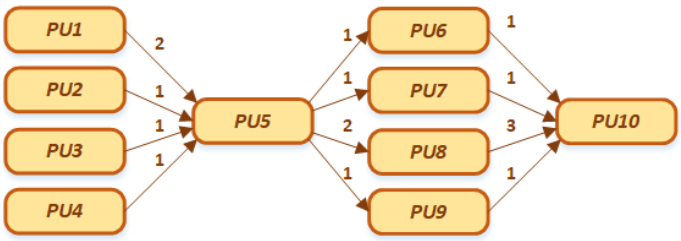
| | |
|---|---|
| | *iDrive* контролера, али се такође уводе подршке за управљање преко сензора додира и гласа. |
| *Toyota.OS* [207] | *Toyota Arene* платформа пружа алате, спреге ка возилу и градивне блокове за безбедносне аспекте у циљу убрзања целокупног развоја. Представља детерминистичку платформу у реалном времену која спаја основне функције возила са функцијама напредних система помоћи возачу и са инфо-забавним системом. |
| *SAIC Z-One SOA* [208] | Компанија *SAIC Motor* је покренула дигиталну сервисно-оријентисану платформу 2021. године. Ова платформа омогућава да се апликације развијају и објаве у продавници апликација, и да власници возила исте могу инсталирати у својим возилима. Платформа се заснива на четири основна техничка стуба: централизована архитектура у возилу, сервисно-оријентисана архитектура, велики број паметних возила који производе податке (енгл. *intelligent vehicle data plant*), целокупно решење за ажурирање и за сајбер сигурност. Поред поменутог, платформа такође садржи и сервисе за приказ интелигентних сцена. |
| *Volvo Cars.OS* [209] | *VolvoCars.OS* се промовише у сврху бржег и флексибилнијег развоја софтвера за електрична возила. Укључује многобројне оперативне системе (од возила до облака), стварајући кохерентно софтверско ОС окружење. Оперативни системи који се налазе испод њега укључују *Android Automotive OS*, *QNX*, *AUTOSAR* и *Linux*. *VolvoCars.OS* пружа спреге за приступ функцијама возила (подаци са сензора, кориснички интерфејс возила, итд.) и функцијама заснованим на облаку (подаци за групу возила) уз сагласност корисника. |

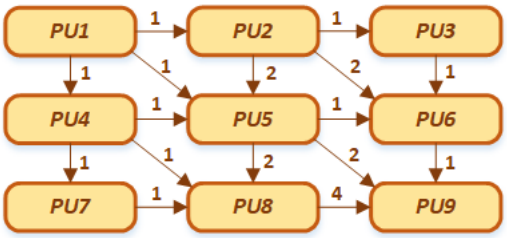
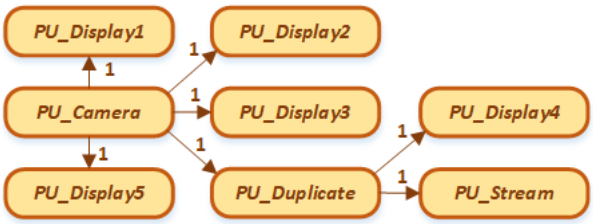
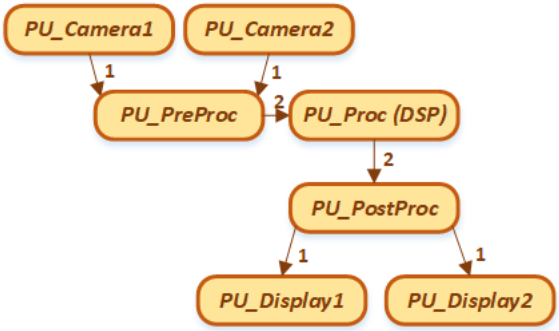
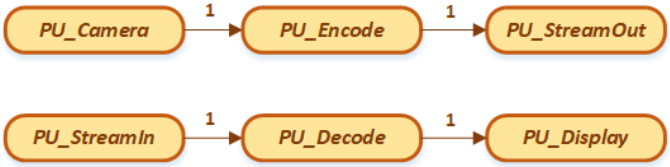
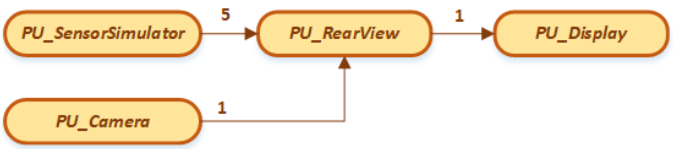
*Табела А.6 Преглед нових оперативних система као целокупних развојних решења*

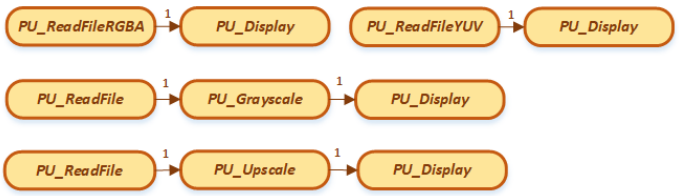
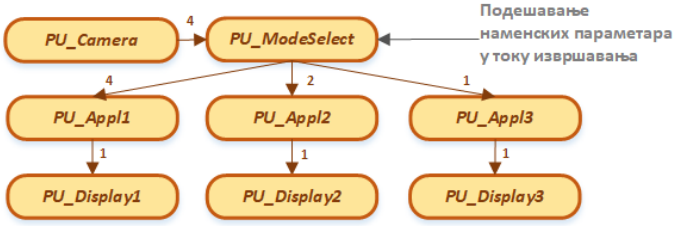
# ДОДАТАК Б: ПРЕГЛЕД ТЕСТНИХ КОРИСНИЧКИХ СЛУЧАЈЕВА

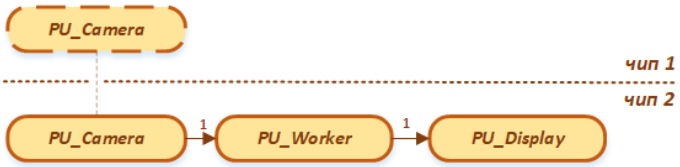
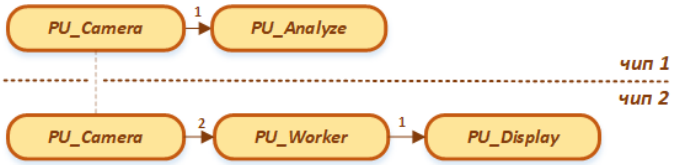
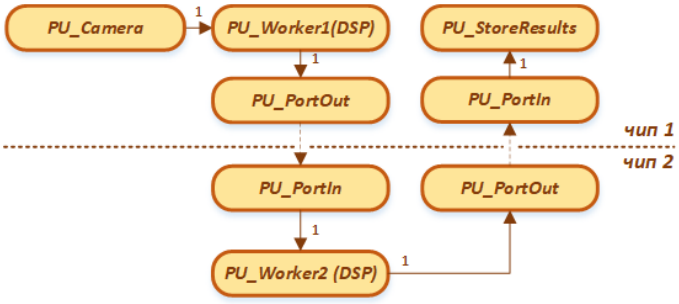
У табели испод је дат преглед репрезентативних основних корисничких тестних случајева. У графичком приказу број изнад AMB везе између јединица обраде означава број AMB редова у оквиру дате везе.

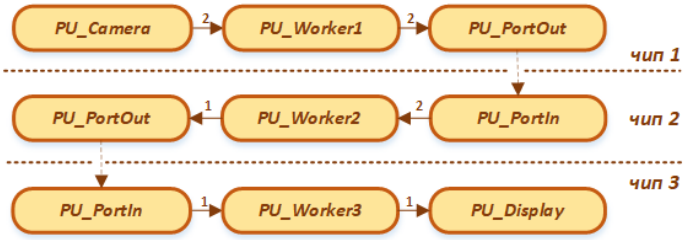
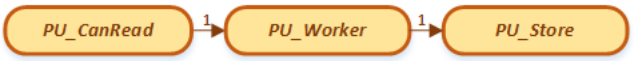
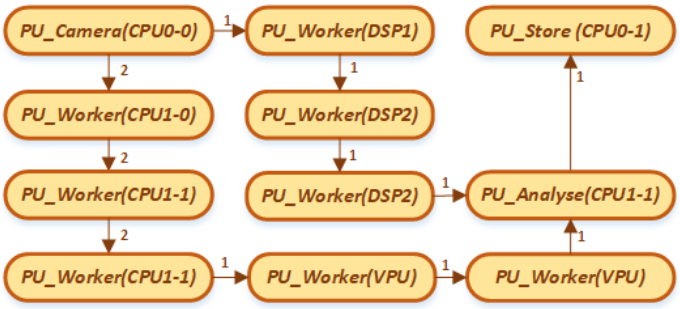
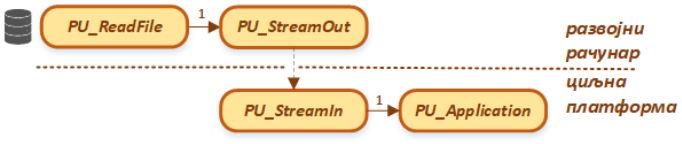
| Тестни случај | Приказ |
|---|---|
| *Тестни случај 1*<br>Учитавање података из улазне датотеке, допремање истих до *DSP*-а (користећи један улазни *Queue*) који само врши копирање и упис података у излазну датотеку. |  |
| *Тестни случај 2*<br>Учитавање података из улазне датотеке, допремање истих до *DSP*-а (користећи два улазна *Queue*-а) који само врши копирање ка два излазна *Queue*-а и упис података у излазне датотеке. |  |
| *Тестни случај 3*<br>Учитавање података из улазне датотеке (величина бафера 1280*720*2), допремање истих до *GPU*-а (користећи један улазни *Queue*) који само врши копирање и упис података у излазну датотеку. |  |
| *Тестни случај 4*<br>Добављање података са једне камере (величина бафера 1280*720*2), допремање истих до *GPU*-а (користећи један улазни |  |

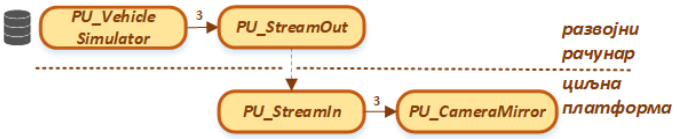
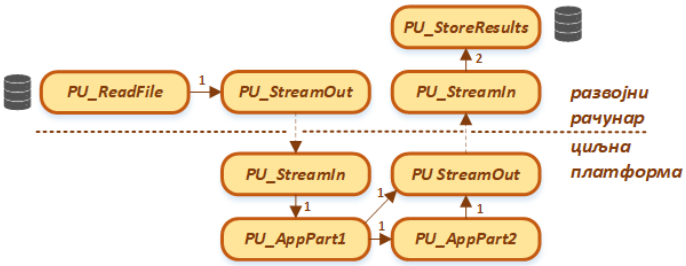
| | |
|---|---|
| *Queue*) који само врши копирање и приказ слике на екран. | |
| *Тестни случај 5*<br>Учитавање података из улазне датотеке, допремање истих до *GPU* и DSP обрадних јединица и упис података у излазне датотеке. Верификује се исправност протока података између *GPU*-а и *DSP*-а у оба смера. |  |
| *Тестни случај 6*<br>Улазна обрадна јединица спорије обрађује податке у односу на излазне обрадне јединице. Верификује се исправност функционисања најплајна у овом случају – очекивање је да излазна обрадна јединица чека на податке. |  |
| *Тестни случај 7*<br>Улазна обрадна јединица брже обрађује податке у односу на излазну обрадну јединицу. Верификује се исправност функционисања најплајна у овом случају – очекивање је да у једном тренутку дође до попуњености реда за размену података и одбацивања података. |  |
| *Тестни случај 8*<br>Обрадна јединица прихвата податке од више обрадних јединица и доставља ка више обрадних јединица. Верификује се исправност функционисања најплајна у овом случају када потенцијално постоји уско грло у систему. |  |

| | |
|---|---|
| *Тестни случај 9*<br>Више обрадних јединица прихвата податке од нула или више обрадних јединица и достављају исте ка нула или више обрадних јединица. Верификује се исправност функционисања најплајна. |  |
| *Тестни случај 10*<br>Улазна обрадна јединица доставља податке са пет камера које се приказују на екран, од чега се садржај једне камере одашиља такође преко етернета. |  |
| *Тестни случај 11*<br>Две улазне обрадне јединице достављају податке са по једне камере. Врши се предобрада података, затим се на *DSP* језгру врши обрада (смањење резолуције), затим додатна обрада при чему се додаје графички слој на саму слику о на крају приказ на екран. |  |
| *Тестни случај 12*<br>Улазна обрадна јединица прикупља податке са камере, затим се врши енкодовање података у *mjpeg* или *h264* формат и исти се емитују на етернет магистралу. Други део најплајна прихвата податке са етернет магистрале, врши декодовање истих и приказ на екран. |  |
| *Тестни случај 13*<br>Информативни приказ камере за задњи поглед са графичким приказом препрека и пројектоване путање кретања возила на основу |  |

| | |
|---|---|
| података са сензора (4 ултразвучна сензора и угао волана). Приказ се врши на екран инфо-забавног система. | |
| *Тестни случај 14*<br>Параметризација обрадне јединице на примеру приказа на екран. Подешавање параметара слике (приликом креирања обрадне јединице) као што је формат боје, резолуција и коришћење исте обрадне јединице у више различитих случајева. |  |
| *Тестни случај 15*<br>Управљање обрадном јединицом у току самог извршавања на примеру приказа слике са камере на инфо-забавни екран. Улазна обрадна јединица доставља снимке са четири камере. Обрадна јединица у односу на тренутна подешавања или у односу на контроле корисника прослеђује на приказивање једну од доступних камера. |  |
| *Тестни случај 16*<br>Управљање обрадном јединицом у току самог извршавања на примеру недовољно ресурса за истовремену обраду свих апликација. Реализована је додатна обрадна јединица која одлучује у односу на режим рада које апликације ће бити активне, тј. коме ће се прослеђивати слике са камера на обраду. Обрадне јединице које не добијају улазне податке су у режиму чекања и не троше |  |

| | |
|---|---|
| процесорско време. Прекључивање између режима рада је могуће на нивоу сваке слике. | |
| *Тестни случај 17*<br>Апликација користи слику са камере иако се физички камера налази повезана на другом чипу. Интерна реализација сервиса омогућава достављање слике са камере на други чип преко доступних магистрала. На првом чипу се имплицитно креира одговарајућа обрадна јединица. |  |
| *Тестни случај 18*<br>Истовремено коришћење слике са камере на чипу на коме се налази повезана камера, као и на другом чипу за који камера није директно повезана. Интерна реализација сервиса омогућава достављање слике са две камере на други чип преко доступних магистрала. На првом чипу корисник већ креира одговарајућу обрадну јединицу тако да њено имплицитно креирање није неопходно. |  |
| *Тестни случај 19*<br>Коришћење додатних процесорских ресурса са другог чипа, због недостатка потребних ресурса на чипу на коме се извршава апликација. Са другог чипа користи се додатно *DSP* језгро за обраду. Пренос података између чипова врши се доступним магистралама на датој платформи. |  |

| | |
|---|---|
| *Тестни случај 20*<br>Коришћење ресурса више чипова за реализацију захтевне апликације. Алгоритамска обрада се извршава на три чипа. Пренос података између чипова врши се доступним магистралама на датој платформи. |  |
| *Тестни случај 21*<br>Прихват података са *CAN* магистрале. Улазна обрадна јединица прихвата податке са *CAN* магистрале и прослеђује даље оне поруке које су подешене у параметрима саме обрадне јединице. |  |
| *Тестни случај 22*<br>Извршавање једне или више обрадних јединица на више истих и/или процесорских јединица. Обрадне јединице у примеру се извршавају на различитим језгрима исте процесорске јединице (*CPU0-0/CPU0-1*, као и *CPU1-0/CPU1-1*), на више инстанци исте процесорске јединице (*DSP0* и *DSP1*), као и на наменској процесорској јединици (*VPU*). Више јединица обраде извршава се на истом језгру (*DSP2*, *CPU1-1*). |  |
| *Тестни случај 23*<br>Симулација видео улаза у апликацију на основу унапред снимљених видео записа. Видео записи се чувају на развојном рачунару и са њега се допремају као улаз у апликацију која се извршава на циљној платформи. Комуникација између циљне |  |

| | |
|---|---|
| платформе и развојног рачунара реализује се преко етернет магистрале. | |
| *Тестни случај 24*<br>Симулација улаза у апликацију на основу групе информација (означених у тренутку времена) који се тичу одређене секвенце вожње. Поред слике са камера, мета-информације се могу односити на одређене *CAN* поруке, брзину возила, положај волана и сл. Симулатор возила се налази развојном рачунари и кроз подешавања се бира које податке ће да шаље апликацији на циљној платформи (у овом примеру слику са две камере и брзину возила). Комуникација између циљне платформе и развојног рачунара реализује се преко етернет магистрале. |  |
| *Тестни случај 25*<br>Симулација улаза у апликацију са развојног рачунара, извршавање апликације на циљној платформи и чување резултата (и/или међурезултата) алгоритма на развојном рачунару у сврхе накнадног прегледања и/или обраде. |  |
| *Тестни случај 26*<br>Креирање исте обрадне јединице на различитим језгрима на примеру обрадне јединице за копирање – као примера обраде који није специфичан за одређено процесорско језгро. Четири исте |  |

| | |
|---|---|
| обрадне јединице се креирају, са једином разликом подешавања параметра афинитета језгра | |

*Табела Б.1 Преглед корисничких тестних случајева*

# ДОДАТАК В: СОФТВЕРСКЕ МЕТРИКЕ

У овом додатку су дати резултати мерења софтверских метрика коришћењем доступних алата. Два алата су коришћена за мерење: *RSM* и *SourceMeter*. У наставку су дати сумирани прегледи добијених резултата и опис метрика које сами алати дају.

## В.1 РЕЗУЛТАТИ МЕРЕЊА – PCM АЛАТ

*Resource Standard Metrics – RSM* је алат за анализу метрика и квалитета изворног кода, који између осталих подржава и *C++* програмски језик. За потребе овог рада коришћена је бесплатна верзија 7.75 у којој нису све опције доступне кориснику. У наставку је дат преглед метрика и резултати истих. Мерења су вршена на нивоу датотеке.

### В.1.1 Преглед метрика доступних у алату

**Број линија кода (*LOC, maxLOC, avgLOC*)**

Број линија кода се дефинише као линија у изворном коду која није коментар или празна линија. Линије које садрже и изворни код и коментар се рачунају у обе метрике. Приликом мерења три вредности су израчунате:

- *LOC* – представља број линија кода у датој датотеци
- *maxLOC* – представља максимални број линија кода по функцији у датој датотеци
- *avgLOC* - представља просечну вредност броја линија кода по функцији у датој датотеци

**Ефективни број линија кода (*eLOC, maxeLOC, avgeLOC*)**

Ефективни број линија кода се дефинише као LOC који није самостална заграда ({} или () ). Метрика је уведена од стране PCM-а. Приликом мерења три вредности су израчунате:

- *eLOC* – представља ефективни број линија кода у датој датотеци
- *maxeLOC* – представља максимални ефективни број линија кода по функцији у датој датотеци
- *avgeLOC* - представља просечну вредност ефективног броја линија кода по функцији у датој датотеци

**Логички број линија кода (*lLOC, maxlLOC, avglLOC*)**

Логички број линија кода се дефинише као број исказа у коду или оних линија које се завршавају са ‘;’. “For” петља се рачуна као једна логичка линија кода иако садржи више ;. Приликом мерења три вредности су израчунате:

- *lLOC* – представља логички број линија кода у датој датотеци
- *maxlLOC* – представља максимални логички број линија кода по функцији у датој датотеци
- *avglLOC* - представља просечну вредност логичког броја линија кода по функцији у датој датотеци

**Број линија кода који представља коментаре у датотеци (*Comments*)**

Број линија кода се израчунава као број линија које садрже коментаре без обзира да ли се у истој линији налази и изворни код.

**Број физичких линија кода у датотеци (*Lines*)**

Број физичких линија кода представља број линија, редова у самом фајлу, без обзира на садржај истих. Број линија кода заједно са коментарима и празним линија може бити већи од физичког броја линија. Ово је последица чињенице када у истој линији постоји логички код и коментар.

**Густина коментара (енгл. *Code Density*): *CD* и *eCD*.**

Представља однос броја коментара (унутар и ван функција) у односу на број исказа/линија кода. Формуле за израчунавање су следеће:

$$CD = \frac{Comments}{LOC + Comments}$$

$$eCD = \frac{Comments}{eLOC + Comments}$$

**Број функција у датотеци (*ffc*)**

Представља број функција у датој датотеци.

**Циклична сложеност (*CC*)**

Степен логичког гранања у функцији се изражава преко *CC*-а. Логичко гранање се дешава када се у функцији појављују следеће кључне речи: "*while*", "*for*", "*if*", "*case*" и "*goto*". CC представља број појављивања ових конструкта. Постоје разна мишљења да ли "for" петља треба да се рачуна у ову групу. PCM алат за рачунање *CC*-а укључује конструкте као у табели испод.

| Конструкт | Укључен у рачунање *CC*-а |
|---|---|
| *Позив функције (енгл. function call)* | да |
| *'while' петља* | да |
| *'for' петља* | да |
| *'switch' конструкт* | не |
| *'case' конструкт* | да |
| *'if' конструкт* | да |
| *'else' конструкт* | не |
| *'?' конструкт – угњеждено 'if'* | да |
| *'goto' конструкт* | да |
| *'\|\|' или 'or' конструкт* | да |
| *'&&' или 'and' конструкт* | да |

*Табела В.1 Конструкти цикличне сложености*

Две вредности су израчунате за сваку од датотека:

- *maxCC* – представља максималну вредност CC-а по функцији у датој датотеци

- *avgCC* - представља просечну вредност CC-а свих функцији у датој датотеци

**Број параметара функције (*maxParam, avgParam*)**

Представља број аргумената дате функције и даје увид у сложеност спреге функције.

Две вредности су израчунате за сваку од датотека:

- *maxParam* – представља максимални број параметара по функцији у датој датотеци
- *avgParam* - представља просечну вредност броја параметара свих функцији у датој датотеци

**Број излазних тачака из функције (*maxRet, avgRet*)**

Представља број излазних тачака из дате функције.

Две вредности су израчунате за сваку од датотека:

- *maxRet* – представља максимални број излазних тачака по функцији у датој датотеци
- *avgRet* - представља просечну вредност броја излазних тачака свих функцији у датој датотеци

### В.1.2 Преглед добијених вредности

У табели испод су приказане добијене вредности из овог алата. Поред самог средњег слоја, анализирале су се и датотеке које припадају слојевима апстракције платформе (три реализације за три платформе).

Колона “валидно” није колона која се оригинално добија из алата, него представља тумачење резултата (да ли су резултати валидни за одређену датотеку или не). Датотеке које су излистане као датотеке које не садрже функције, немају ни вредности које се рачунају по функцији Такође, последња датотека у табели је означена као неисправна. Сама датотека садржи реализацију класе и метода, али је алат не препознаје као такву, за разлику од других њој сличних, и не даје валидне резултате.

Такође, колоне ‘*CD* ’ и ‘*eCD* ’, не представљају вредности које даје алат као резултат анализе, него израчунате вредности за густину коментара – као што је описано у потпоглављу Густина коментара (енгл. *Code Density*): *CD* и *eCD*.

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 74 | 53 | 33 | 33 | 146 | 0.31 | 0.38 | 8 | да | 2 | 1.25 | 4 | 1 | 1 | 1 | 11 | 6.75 | 7 | 4.25 | 6 | 4 |
| 2 | 191 | 118 | 76 | 39 | 276 | 0.17 | 0.25 | 16 | да | 8 | 2 | 3 | 0.63 | 1 | 1 | 39 | 10.4 | 21 | 5.88 | 13 | 4.75 |
| 3 | 60 | 39 | 24 | 29 | 113 | 0.33 | 0.43 | 10 | да | 1 | 1 | 0 | 0 | 1 | 1 | 6 | 4.4 | 4 | 2.4 | 4 | 2.4 |
| 4 | 113 | 86 | 56 | 31 | 194 | 0.22 | 0.26 | 9 | да | 2 | 1.44 | 2 | 0.67 | 1 | 1 | 19 | 10.8 | 15 | 7.89 | 12 | 6.11 |
| 5 | 46 | 29 | 14 | 31 | 94 | 0.40 | 0.52 | 7 | да | 2 | 1.14 | 5 | 1.29 | 1 | 1 | 7 | 4.29 | 4 | 2 | 4 | 1.86 |
| 6 | 71 | 51 | 37 | 28 | 132 | 0.28 | 0.35 | 9 | да | 1 | 1 | 2 | 1.11 | 1 | 1 | 10 | 6.11 | 8 | 4.11 | 8 | 4.11 |
| 7 | 66 | 45 | 30 | 29 | 124 | 0.31 | 0.39 | 10 | да | 1 | 1 | 1 | 0.2 | 1 | 1 | 6 | 5 | 4 | 3 | 4 | 3 |
| 8 | 206 | 134 | 93 | 27 | 332 | 0.12 | 0.17 | 25 | да | 5 | 1.4 | 3 | 0.88 | 1 | 1 | 22 | 6.88 | 14 | 4.08 | 11 | 3.68 |
| 9 | 14 | 7 | 2 | 28 | 48 | 0.67 | 0.80 | 3 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 2.67 | 1 | 0.67 | 1 | 0.67 |
| 10 | 100 | 69 | 44 | 30 | 165 | 0.23 | 0.30 | 8 | да | 4 | 1.88 | 3 | 1 | 1 | 1 | 22 | 9.75 | 14 | 6 | 12 | 5.5 |
| 11 | 3 | 2 | 0 | 30 | 38 | 0.91 | 0.94 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 12 | 548 | 353 | 216 | 61 | 771 | 0.10 | 0.15 | 28 | да | 28 | 3.5 | 3 | 0.5 | 1 | 1 | 144 | 17.9 | 90 | 11 | 61 | 7.71 |
| 13 | 19 | 17 | 1 | 33 | 56 | 0.63 | 0.66 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 14 | 34 | 30 | 14 | 101 | 156 | 0.75 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 15 | 42 | 38 | 27 | 131 | 208 | 0.76 | 0.78 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | 30 | 25 | 14 | 79 | 144 | 0.72 | 0.76 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 17 | 37 | 33 | 18 | 98 | 156 | 0.73 | 0.75 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 18 | 41 | 27 | 16 | 97 | 149 | 0.70 | 0.78 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 19 | 24 | 21 | 11 | 92 | 143 | 0.79 | 0.81 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 20 | 27 | 25 | 3 | 54 | 69 | 0.67 | 0.68 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 21 | 85 | 84 | 60 | 47 | 140 | 0.36 | 0.36 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 22 | 31 | 26 | 14 | 81 | 144 | 0.72 | 0.76 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 23 | 69 | 63 | 41 | 243 | 358 | 0.78 | 0.79 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 24 | 19 | 16 | 6 | 55 | 87 | 0.74 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 25 | 80 | 73 | 35 | 169 | 254 | 0.68 | 0.70 | 1 | да | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 26 | 14 | 11 | 4 | 56 | 81 | 0.80 | 0.84 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 27 | 96 | 90 | 65 | 321 | 465 | 0.77 | 0.78 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 28 | 34 | 31 | 10 | 78 | 128 | 0.70 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 29 | 21 | 17 | 9 | 79 | 120 | 0.79 | 0.82 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 30 | 22 | 17 | 10 | 71 | 120 | 0.76 | 0.81 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 31 | 20 | 16 | 7 | 80 | 117 | 0.80 | 0.83 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 32 | 34 | 27 | 8 | 88 | 142 | 0.72 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 33 | 22 | 18 | 5 | 63 | 98 | 0.74 | 0.78 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 34 | 66 | 61 | 41 | 163 | 263 | 0.71 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 35 | 25 | 21 | 10 | 113 | 160 | 0.82 | 0.84 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 36 | 25 | 19 | 11 | 75 | 126 | 0.75 | 0.80 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 37 | 23 | 20 | 9 | 87 | 132 | 0.79 | 0.81 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 38 | 3 | 2 | 0 | 28 | 32 | 0.90 | 0.93 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 39 | 9 | 6 | 1 | 30 | 42 | 0.77 | 0.83 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 40 | 13 | 8 | 2 | 28 | 45 | 0.68 | 0.78 | 2 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 41 | 3 | 2 | 0 | 28 | 33 | 0.90 | 0.93 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 42 | 6 | 3 | 0 | 28 | 37 | 0.82 | 0.90 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 2 | 2 | 0 | 0 | 0 | 0 |
| 43 | 9 | 6 | 1 | 28 | 41 | 0.76 | 0.82 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 44 | 186 | 155 | 54 | 28 | 263 | 0.13 | 0.15 | 14 | да | 2 | 1.07 | 3 | 1.71 | 1 | 1 | 10 | 8.5 | 8 | 6.36 | 5 | 3.71 |
| 45 | 9 | 6 | 1 | 28 | 40 | 0.76 | 0.82 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 46 | 13 | 8 | 2 | 28 | 45 | 0.68 | 0.78 | 2 | да | 1 | 1 | 0 | 0 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 47 | 14 | 9 | 3 | 33 | 53 | 0.70 | 0.79 | 2 | да | 1 | 1 | 1 | 0.5 | 1 | 1 | 4 | 3.5 | 2 | 1.5 | 2 | 1.5 |
| 48 | 48 | 28 | 10 | 30 | 98 | 0.38 | 0.52 | 5 | да | 2 | 1.4 | 4 | 1.6 | 1 | 1 | 12 | 6.4 | 6 | 2.8 | 4 | 2 |

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 49 | 108 | 83 | 65 | 29 | 192 | 0.21 | 0.26 | 10 | да | 2 | 1.2 | 1 | 0.7 | 1 | 1 | 18 | 9.3 | 14 | 6.9 | 13 | 6.5 |
| 50 | 71 | 43 | 22 | 29 | 133 | 0.29 | 0.40 | 10 | да | 2 | 1.3 | 0 | 0 | 1 | 1 | 13 | 5.6 | 9 | 3 | 7 | 2.2 |
| 51 | 111 | 83 | 60 | 33 | 193 | 0.23 | 0.28 | 6 | да | 4 | 2.17 | 2 | 1 | 1 | 1 | 45 | 16.2 | 37 | 11.8 | 32 | 10 |
| 52 | 86 | 57 | 35 | 32 | 147 | 0.27 | 0.36 | 6 | да | 9 | 3 | 5 | 1.5 | 1 | 1 | 36 | 12 | 26 | 7.33 | 19 | 5.83 |
| 53 | 101 | 65 | 31 | 27 | 158 | 0.21 | 0.29 | 9 | да | 9 | 1.89 | 3 | 1.22 | 1 | 1 | 45 | 9.22 | 27 | 5.44 | 17 | 3.44 |
| 54 | 81 | 51 | 26 | 29 | 147 | 0.26 | 0.36 | 10 | да | 2 | 1.4 | 1 | 0.2 | 1 | 1 | 13 | 6.6 | 9 | 3.8 | 6 | 2.6 |
| 55 | 671 | 515 | 377 | 130 | 1080 | 0.16 | 0.20 | 29 | да | 13 | 3.24 | 5 | 1.28 | 1 | 1 | 97 | 21.3 | 87 | 16 | 63 | 12.9 |
| 56 | 12 | 12 | 8 | 28 | 44 | 0.70 | 0.70 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 57 | 23 | 19 | 6 | 72 | 113 | 0.76 | 0.79 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 58 | 39 | 36 | 16 | 98 | 166 | 0.72 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 59 | 36 | 30 | 13 | 76 | 147 | 0.68 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 60 | 35 | 31 | 12 | 102 | 159 | 0.74 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 61 | 39 | 32 | 12 | 102 | 161 | 0.72 | 0.76 | 1 | да | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 62 | 22 | 19 | 11 | 96 | 144 | 0.81 | 0.83 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 63 | 37 | 31 | 13 | 77 | 151 | 0.68 | 0.71 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 64 | 72 | 68 | 41 | 348 | 486 | 0.83 | 0.84 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 65 | 11 | 11 | 8 | 28 | 42 | 0.72 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 66 | 23 | 19 | 6 | 72 | 114 | 0.76 | 0.79 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 67 | 35 | 32 | 15 | 88 | 142 | 0.72 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 68 | 33 | 27 | 14 | 77 | 143 | 0.70 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 69 | 57 | 44 | 22 | 128 | 225 | 0.69 | 0.74 | 4 | да | 1 | 1 | 1 | 0.25 | 1 | 1 | 3 | 2.75 | 1 | 0.75 | 1 | 0.75 |
| 70 | 39 | 32 | 12 | 104 | 163 | 0.73 | 0.76 | 1 | да | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |
| 71 | 24 | 20 | 10 | 68 | 109 | 0.74 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 72 | 21 | 18 | 8 | 67 | 99 | 0.76 | 0.79 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 73 | 22 | 19 | 11 | 96 | 144 | 0.81 | 0.83 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 74 | 35 | 29 | 14 | 81 | 151 | 0.70 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 75 | 69 | 61 | 35 | 184 | 280 | 0.73 | 0.75 | 1 | да | 2 | 2 | 0 | 0 | 2 | 2 | 7 | 7 | 3 | 3 | 2 | 2 |
| 76 | 64 | 40 | 19 | 41 | 125 | 0.39 | 0.51 | 5 | да | 2 | 1.6 | 4 | 1.6 | 1 | 1 | 13 | 9.4 | 7 | 5 | 5 | 3.8 |
| 77 | 100 | 67 | 47 | 36 | 168 | 0.26 | 0.35 | 8 | да | 5 | 2.38 | 1 | 0.63 | 1 | 1 | 22 | 10.8 | 13 | 6.75 | 11 | 5.88 |
| 78 | 66 | 40 | 25 | 29 | 124 | 0.31 | 0.42 | 10 | да | 2 | 1.2 | 0 | 0 | 1 | 1 | 11 | 5.1 | 7 | 2.7 | 6 | 2.5 |
| 79 | 161 | 106 | 73 | 46 | 268 | 0.22 | 0.30 | 6 | да | 9 | 4.17 | 2 | 1 | 1 | 1 | 60 | 24.5 | 42 | 15.8 | 31 | 12.2 |
| 80 | 93 | 64 | 38 | 32 | 157 | 0.26 | 0.33 | 6 | да | 9 | 3 | 5 | 1.5 | 1 | 1 | 40 | 13.2 | 30 | 8.5 | 19 | 6.33 |
| 81 | 98 | 70 | 40 | 29 | 159 | 0.23 | 0.29 | 8 | да | 6 | 1.63 | 6 | 1.13 | 5 | 1.5 | 35 | 9 | 23 | 5.75 | 18 | 4.75 |
| 82 | 70 | 48 | 29 | 29 | 120 | 0.29 | 0.38 | 5 | да | 3 | 2 | 6 | 1.6 | 1 | 1 | 20 | 10.8 | 14 | 6.8 | 12 | 5.8 |
| 83 | 89 | 57 | 38 | 27 | 152 | 0.23 | 0.32 | 9 | да | 2 | 1.67 | 3 | 1.22 | 1 | 1 | 11 | 8.22 | 7 | 4.89 | 6 | 4.22 |
| 84 | 76 | 48 | 32 | 29 | 140 | 0.28 | 0.38 | 10 | да | 2 | 1.3 | 1 | 0.2 | 1 | 1 | 11 | 6.1 | 7 | 3.5 | 6 | 3.2 |
| 85 | 254 | 178 | 104 | 81 | 423 | 0.24 | 0.31 | 1 | не | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 86 | 93 | 64 | 38 | 32 | 157 | 0.26 | 0.33 | 6 | да | 9 | 3 | 5 | 1.5 | 1 | 1 | 40 | 13.2 | 30 | 8.5 | 19 | 6.33 |
| 87 | 89 | 57 | 38 | 27 | 152 | 0.23 | 0.32 | 9 | да | 2 | 1.67 | 3 | 1.22 | 1 | 1 | 11 | 8.22 | 7 | 4.89 | 6 | 4.22 |
| 88 | 76 | 48 | 32 | 29 | 140 | 0.28 | 0.38 | 10 | да | 2 | 1.3 | 1 | 0.2 | 1 | 1 | 11 | 6.1 | 7 | 3.5 | 6 | 3.2 |
| 89 | 60 | 40 | 25 | 29 | 124 | 0.33 | 0.42 | 10 | да | 2 | 1.2 | 0 | 0 | 1 | 1 | 11 | 5.1 | 7 | 2.7 | 6 | 2.5 |
| 90 | 64 | 40 | 19 | 41 | 125 | 0.39 | 0.51 | 5 | да | 2 | 1.6 | 4 | 1.6 | 1 | 1 | 13 | 9.4 | 7 | 5 | 5 | 3.8 |
| 91 | 101 | 68 | 48 | 37 | 169 | 0.27 | 0.35 | 8 | да | 5 | 2.38 | 1 | 0.63 | 1 | 1 | 22 | 10.9 | 14 | 6.88 | 12 | 6 |
| 92 | 86 | 58 | 38 | 32 | 148 | 0.27 | 0.36 | 6 | да | 4 | 2 | 2 | 1 | 1 | 1 | 28 | 11.8 | 18 | 7.5 | 14 | 6.33 |
| 93 | 94 | 62 | 37 | 42 | 177 | 0.31 | 0.40 | 8 | да | 3 | 1.88 | 3 | 0.5 | 1 | 1 | 19 | 9.25 | 11 | 5.5 | 8 | 4.63 |
| 94 | 39 | 32 | 12 | 104 | 163 | 0.73 | 0.76 | 1 | да | 1 | 1 | 2 | 2 | 1 | 1 | 3 | 3 | 1 | 1 | 1 | 1 |

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 95 | 22 | 19 | 11 | 96 | 144 | 0.81 | 0.83 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 96 | 9 | 9 | 6 | 30 | 42 | 0.77 | 0.77 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 97 | 35 | 29 | 14 | 81 | 152 | 0.70 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 98 | 34 | 28 | 14 | 77 | 145 | 0.69 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 99 | 21 | 17 | 6 | 72 | 111 | 0.77 | 0.81 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 100 | 36 | 33 | 15 | 88 | 144 | 0.71 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 101 | 30 | 26 | 11 | 141 | 218 | 0.82 | 0.84 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 102 | 51 | 43 | 24 | 177 | 258 | 0.78 | 0.80 | 1 | да | 2 | 2 | 0 | 0 | 2 | 2 | 7 | 7 | 3 | 3 | 2 | 2 |
| 103 | 226 | 159 | 103 | 30 | 322 | 0.12 | 0.16 | 12 | да | 12 | 3.17 | 3 | 0.83 | 1 | 1 | 47 | 15.8 | 33 | 10.3 | 26 | 8.5 |
| 104 | 98 | 63 | 39 | 30 | 159 | 0.23 | 0.32 | 8 | да | 7 | 2.25 | 3 | 0.5 | 1 | 1 | 29 | 10.4 | 17 | 6.13 | 12 | 4.88 |
| 105 | 123 | 82 | 51 | 30 | 197 | 0.20 | 0.27 | 11 | да | 7 | 2 | 3 | 0.73 | 1 | 1 | 30 | 9.27 | 20 | 5.64 | 15 | 4.64 |
| 106 | 226 | 159 | 96 | 28 | 321 | 0.11 | 0.15 | 16 | да | 10 | 2.63 | 3 | 0.69 | 1 | 1 | 29 | 11.9 | 19 | 7.75 | 14 | 6 |
| 107 | 129 | 88 | 62 | 31 | 206 | 0.19 | 0.26 | 8 | да | 9 | 3 | 3 | 0.5 | 1 | 1 | 34 | 14.3 | 24 | 9.25 | 22 | 7.75 |
| 108 | 131 | 90 | 64 | 31 | 208 | 0.19 | 0.26 | 8 | да | 9 | 3 | 3 | 0.5 | 1 | 1 | 34 | 14.5 | 24 | 9.5 | 22 | 8 |
| 109 | 44 | 31 | 16 | 26 | 84 | 0.37 | 0.46 | 3 | да | 4 | 2 | 1 | 0.33 | 1 | 1 | 24 | 10.3 | 16 | 6.33 | 13 | 5.33 |
| 110 | 44 | 31 | 16 | 26 | 84 | 0.37 | 0.46 | 3 | да | 4 | 2 | 1 | 0.33 | 1 | 1 | 24 | 10.3 | 16 | 6.33 | 13 | 5.33 |
| 111 | 251 | 164 | 105 | 29 | 348 | 0.10 | 0.15 | 12 | да | 16 | 4.67 | 6 | 1.5 | 1 | 1 | 59 | 18.7 | 37 | 11.5 | 28 | 8.75 |
| 112 | 219 | 154 | 78 | 29 | 312 | 0.12 | 0.16 | 12 | да | 12 | 3.17 | 3 | 0.83 | 1 | 1 | 69 | 16 | 49 | 10.7 | 23 | 6.5 |
| 113 | 153 | 98 | 62 | 33 | 237 | 0.18 | 0.25 | 12 | да | 11 | 2.5 | 3 | 0.75 | 1 | 1 | 47 | 10.9 | 23 | 6.42 | 20 | 5.17 |
| 114 | 145 | 94 | 59 | 30 | 224 | 0.17 | 0.24 | 12 | да | 8 | 2.33 | 3 | 0.75 | 1 | 1 | 28 | 10.3 | 18 | 6.08 | 13 | 4.92 |
| 115 | 265 | 166 | 107 | 31 | 360 | 0.10 | 0.16 | 12 | да | 23 | 4.5 | 5 | 1.25 | 1 | 1 | 117 | 20.5 | 71 | 12.3 | 47 | 8.92 |
| 116 | 163 | 120 | 75 | 32 | 249 | 0.16 | 0.21 | 11 | да | 7 | 2.18 | 3 | 0.73 | 1 | 1 | 32 | 12.5 | 25 | 8.64 | 18 | 6.82 |
| 117 | 151 | 104 | 62 | 28 | 230 | 0.16 | 0.21 | 11 | да | 8 | 2.36 | 3 | 0.73 | 1 | 1 | 37 | 11.5 | 25 | 7.27 | 16 | 5.64 |

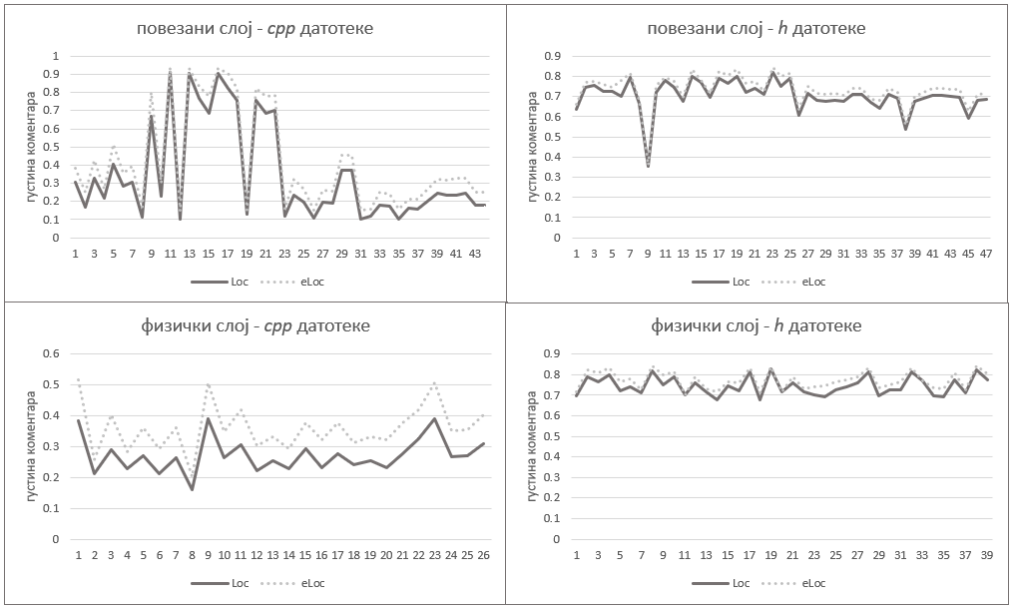
| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 118 | 59 | 53 | 26 | 92 | 148 | 0.61 | 0.63 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 119 | 19 | 16 | 9 | 48 | 81 | 0.72 | 0.75 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 120 | 32 | 27 | 15 | 68 | 118 | 0.68 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 121 | 59 | 50 | 32 | 123 | 196 | 0.68 | 0.71 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 122 | 21 | 18 | 11 | 45 | 78 | 0.68 | 0.71 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 123 | 23 | 20 | 13 | 48 | 82 | 0.68 | 0.71 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 124 | 21 | 18 | 6 | 52 | 84 | 0.71 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 125 | 21 | 18 | 6 | 52 | 84 | 0.71 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 126 | 48 | 45 | 19 | 99 | 152 | 0.67 | 0.69 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 127 | 62 | 52 | 30 | 110 | 186 | 0.64 | 0.68 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 128 | 36 | 31 | 19 | 88 | 145 | 0.71 | 0.74 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 129 | 36 | 31 | 18 | 81 | 136 | 0.69 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 130 | 55 | 51 | 19 | 64 | 109 | 0.54 | 0.56 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 131 | 42 | 37 | 24 | 87 | 136 | 0.67 | 0.70 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 132 | 36 | 31 | 20 | 80 | 129 | 0.69 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 133 | 115 | 80 | 52 | 29 | 181 | 0.20 | 0.27 | 8 | да | 4 | 2.25 | 3 | 0.5 | 1 | 1 | 30 | 11.8 | 22 | 7.5 | 21 | 6.5 |
| 134 | 117 | 80 | 61 | 38 | 195 | 0.25 | 0.32 | 8 | да | 6 | 2.25 | 3 | 0.5 | 1 | 1 | 45 | 13 | 33 | 8.5 | 32 | 7.63 |
| 135 | 98 | 65 | 40 | 30 | 158 | 0.23 | 0.32 | 10 | да | 2 | 1.6 | 3 | 0.8 | 1 | 1 | 12 | 7.9 | 8 | 4.7 | 6 | 4 |
| 136 | 97 | 62 | 38 | 30 | 157 | 0.24 | 0.33 | 8 | да | 7 | 2.25 | 3 | 0.5 | 1 | 1 | 29 | 10.3 | 17 | 6 | 12 | 4.75 |
| 137 | 98 | 65 | 41 | 32 | 162 | 0.25 | 0.33 | 8 | да | 7 | 2.25 | 3 | 0.5 | 1 | 1 | 29 | 10.3 | 19 | 6.25 | 15 | 5.13 |
| 138 | 136 | 89 | 57 | 30 | 221 | 0.18 | 0.25 | 13 | да | 7 | 2 | 3 | 0.77 | 1 | 1 | 23 | 8.77 | 15 | 5.23 | 11 | 4.38 |
| 139 | 136 | 89 | 57 | 30 | 221 | 0.18 | 0.25 | 13 | да | 7 | 2 | 3 | 0.77 | 1 | 1 | 23 | 8.77 | 15 | 5.23 | 11 | 4.38 |
| 140 | 20 | 17 | 11 | 48 | 80 | 0.71 | 0.74 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Редни број датотеке | *LOC* | *eLOC* | *lLOC* | *Comments* | *Lines* | *CD* | *eCD* | *ffc* | валидно? | *maxCC* | *avgCC* | *maxParam* | *avgParam* | *maxRet* | *avgRet* | *maxLOC* | *avgLOC* | *maxeLOC* | *avgeLOC* | *maxlLOC* | *avgeLOC* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 141 | 20 | 17 | 11 | 48 | 80 | 0.71 | 0.74 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 142 | 31 | 26 | 16 | 72 | 118 | 0.70 | 0.73 | 1 | да | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 143 | 19 | 16 | 9 | 44 | 76 | 0.70 | 0.73 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 144 | 31 | 27 | 12 | 45 | 93 | 0.59 | 0.63 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 145 | 37 | 32 | 19 | 79 | 137 | 0.68 | 0.71 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |
| 146 | 37 | 32 | 19 | 81 | 139 | 0.69 | 0.72 | N/A | не | - | - | - | - | - | - | - | - | - | - | - | - |

*Табела В.2 Вредности метрика добијене PCM алатом*

### Густина коментара

Слика В.1 приказује густину коментара по датотекама у односу на број линија кода и у односу на ефективни број линија кода. Густина коментара у односу на ефективни број линија кода је очекивано већа или једнака густини коментара у односу на број линија кода. Просечна вредност густине коментара за цели софтверски пакет је 0.51 у односу на број линија кода и 0.58 у односу на ефективни број линија кода.



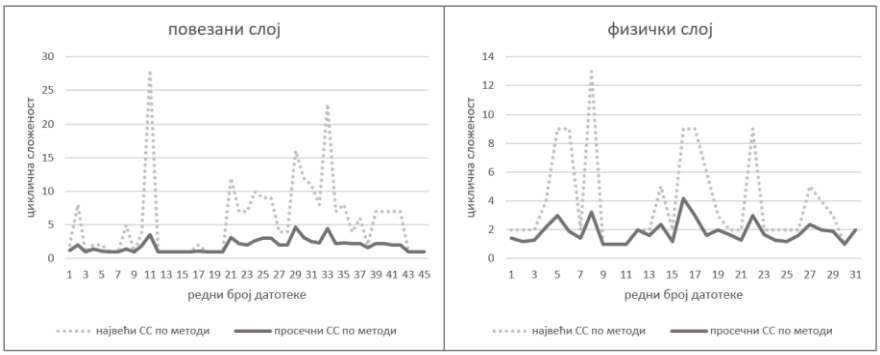

*Слика В.1 Густина коментара – PCM алат*

### Циклична сложеност

Слика В.2 приказује просечне и максималне вредности сложености по функцији у датотеци. У односу на графике густине коментара приметно је да је број датотека мањи. То је због чињенице да неке од датотека нису приказане на графику из неког од следећих разлога:

- у датотеци није детектована ниједна функција (нпр. *.h* датотеке).
- у *.h* датотеци детектована једна функција, са вредношћу цикличне сложености један (нису од интереса, јер посматрамо горње границе сложености)

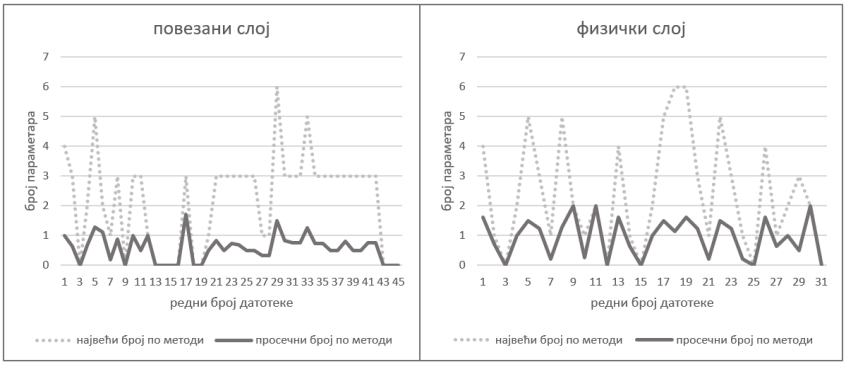
Поред просечне вредности исцртана је и максимална вредност по функцији за дату датотеку.




*Слика В.2 Циклична сложеност – просечна и највећа вредност по методи у датотеци*

### Број параметара функција

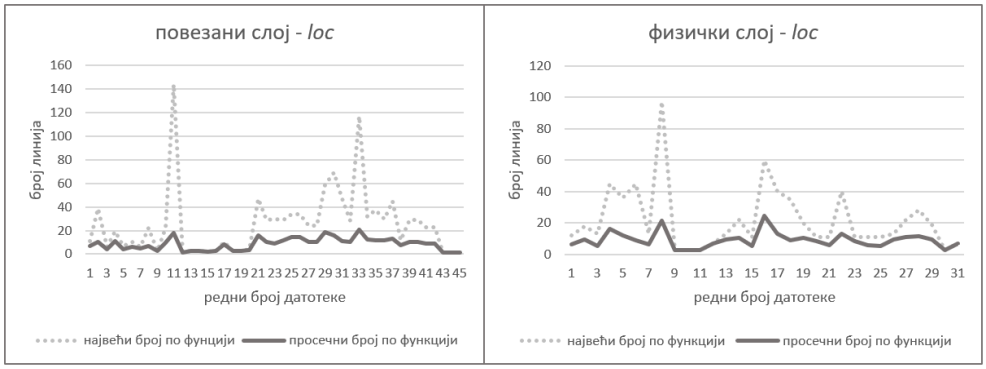
Слика В.3 приказује просечан и максимални број параметара по функцији за одређену датотеку.



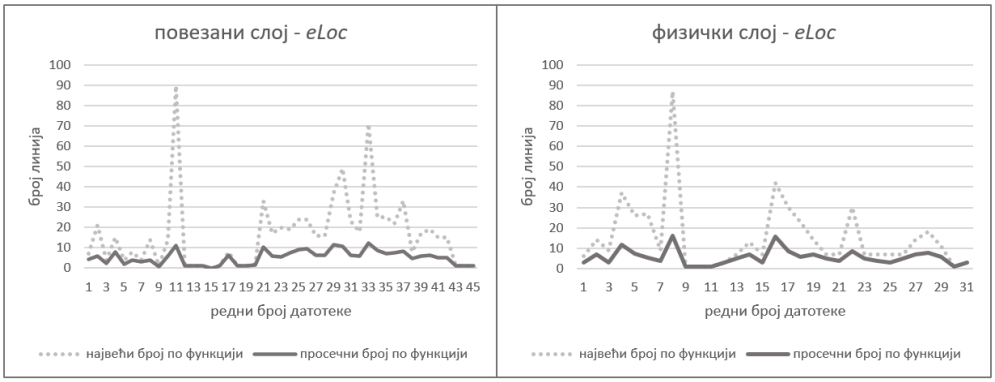

*Слика В.3 Број параметара функције – просечни и максимални по датотеци*

### Број инструкција по функцији

Слика В.4 и Слика В.5 приказују просечну и највећу вредност инструкција по функцијама по датотеци (број линија и ефективни број линија).




*Слика В.4 Просечна и највећа вредност LOC по функцијама за датотеку*




*Слика В.5 Просечна и највећа вредност eLOC по функцијама за датотеку*

### В.1.3 Израчунате метрике на нивоу пројекта

Укупна густина коментара за све датотеке се може израчунати на следећи начин:

$$CDTotal = \frac{\sum_{i=1}^{N} Comments_i}{\sum_{i=1}^{N} Comments_i + \sum_{i=1}^{N} LOC_i}$$

$$eCDTotal = \frac{\sum_{i=1}^{N} Comments_i}{\sum_{i=1}^{N} Comments_i + \sum_{i=1}^{N} eLOC_i}$$

где *N* представља број доступних датотека у пројекту.

Просечна циклична сложеност за све функције у датом пројекту може се израчунати на следећи начин:

$$avgCCTotal = \frac{\sum_{i=1}^{N}(avgCC_i * ffc_i)}{\sum_{i=1}^{N} ffc_i}$$

где *N* представља број исправних датотека које су се квалификовале за рачунање – датотеке означене као валидне.

Просечни број параметара за све функције у датом пројекту може се израчунати на следећи начин:

$$avgParamTotal = \frac{\sum_{i=1}^{N}(avgParam_i * ffc_i)}{\sum_{i=1}^{N} ffc_i}$$

где *N* представља број исправних датотека које су се квалификовале за рачунање – датотеке означене као валидне.

Просечни број излазних тачака за све функције у датом пројекту може се израчунати на следећи начин:

$$avgRetTotal = \frac{\sum_{i=1}^{N}(avgRet_i * ffc_i)}{\sum_{i=1}^{N} ffc_i}$$

где N представља број исправних датотека које су се квалификовале за рачунање – датотеке означене као валидне.

Вредности које се добијају за анализиране датотеке су приказане у табели испод.

| **Метрика** | *CDTotal* | *eCDTotal* | *avgCCTotal* | *avgParamTotal* | *avgRetTotal* |
|---|---|---|---|---|---|
| **Вредност** | 0.51 | 0.58 | 1.88 | 0.81 | 1.01 |

*Табела В.3 Вредности метрика на нивоу целокупног софтверског пројекта*

## В.2 РЕЗУЛТАТИ МЕРЕЊА – CM АЛАТ

CM алат *(*енгл. *SourceMeter)* је алат за анализу изворног кода, који између осталих подржава и *C++* програмски језик. За разлику од PCM алата, где је довољно проследити изворне датотеке, за анализу кода овај алат захтева део који се компајлира, и то са *gcc* компатибилним алатима. Самим тим све датотеке нису укључене у анализу. Анализа је вршена на порту који се извршава на Линукс оперативном систему. За потребе овог рада коришћена је бесплатна верзија 9.2 у

којој нису све опције доступне кориснику. У наставку је дат преглед метрика и резултати истих.

### В.2.1 Преглед метрика доступних у алату

Алат анализира изворни код на више нивоа: класа, метода, датотека, компонента итд. За сваки од ових нивоа нуди велики број метрика груписаних у шест група. У табели испод дат је преглед метрика по групама и нивоима за које су добијени резултати. Много већи број метрика је подржан од стране алата али није доступан у бесплатној верзији.

| Група | Метрика | *Класа* | *Компонента* | *Енумерација* | *Датотека* | *Функција* | *Интерфејс* | *Метода* | *Простор имена* | *Структура* | *Унија* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Кохезија* | *LCOM5* | x | | | | | | | | x | x |
| *Комплексност* | *HPV* | | | | | x | | x | | | |
| | *HPL* | | | | | x | | x | | | |
| | *McCC* | | | | x | x | | x | | | |
| | *WMC* | x | | | | | x | | | x | |
| *Спрегнутост* | *CBO* | x | | | | | | | | x | x |
| | *NOI* | x | | | | x | | x | | x | x |
| | *RFC* | x | | | | | | | | x | x |
| *Документација* | *CLOC* | x | | | x | x | x | x | x | x | x |
| | *TCLOC* | x | x | | | x | x | x | x | x | x |
| *Наслеђивање* | *DIT* | x | | | | | x | | | x | |
| | *NOC* | x | | | | | x | | | x | |
| *Величина* | *LLOC* | x | | x | x | x | x | x | x | x | x |
| | *LOC* | x | | x | x | x | x | x | x | x | x |
| | *NCL* | | | | | | | | x | | |
| | *NEN* | | | | | | | | x | | |
| | *TLLOC* | x | x | x | | x | x | x | x | x | x |
| | *TLOC* | x | x | x | | x | x | x | x | x | x |
| | *TNCL* | | x | | | | | | x | | |
| | *TNEN* | | x | | | | | | x | | |

*Табела В.4 Метрике подржане у СМ алату*

**Повезаност**

Представља меру у којој су елементи изворног кода међусобно повезани.

*LCOM5 - Недостатак кохезије у методама (енгл. Lack of Cohesion in Methods)*

Број функционалности класе. Један од основних принципа објектно оријентисаног програмирања је енкапсулација, што значи да атрибути који припадају заједно и операције које их користе требају бити организоване у једну класу, а једна класа ће реализовати само једну функционалност, односно њени атрибути и методе треба да буду кохерентни. Ова метрика мери недостатак кохезије и израчунава у колико би кохерентних класа, класа могла да се подели. Израчунава се узимањем неусмереног графикона, где су чворови реализоване локалне методе класе и постоји ивица између два чвора ако и само ако се користи заједнички (локални или наслеђени) атрибут или апстрактна метода или метода позива другу. Вредност метрике је број повезаних компоненти у графикону не рачунајући оне које садрже само конструкторе, деструкторе, методе за постављање и добијање одређеног атрибута.

**Комплексност**

Представља меру комплексности елемената изворног кода.

*HPV – Халстедов програмски речник (енгл. Halstead Program Vocabulary)*

Представља суму укупног броја различитих оператора и операнада у програму:

$$HPV = n_1 + n_2$$

где је:

- $n_1$ – укупан број различитих оператора (семантичка значења резервисаних кључних речи, тачка-запете, блокова и идентификатора осим у својим декларацијама)
- $n_2$ – укупан број различитих операнада (литерали, нумерички литерали и идентификатори у својим декларацијама)

*HPL - Халстедова дужина програма (енгл. Halstead Program Length)*

Представља суму укупног броја оператора и операнада у програму:

$$HPL = N_1 + N_2$$

где је:

- $N_1$ – укупан број оператора (семантичка значења резервисаних кључних речи, тачка-запете, блокова и идентификатора осим у својим декларацијама)
- $N_2$ – укупан број операнада (литерали, нумерички литерали и идентификатори у својим декларацијама)

*McCC – Циклична сложеност (енгл. McCabe's Cyclomatic Complexity)*

Представља сложеност изражену као број независних путања у изворном коду. Представља доњу границу за број могућих путања извршавања у изворном коду, а истовремено је горња граница за минимални број тестних случајева који су потребни за постизање потпуне покривености тестирања гранања. Вредност метрике се израчунава као број конструката укључених у рачунање из табеле испод + 1.

| **Конструкт** | **Укључен у рачунање *McCC*-а** |
|---|---|
| *'while' или 'do while' петља* | да |
| *'for' петља* | да |
| *'switch' конструкт* | не |
| '*case' конструкт* | да |
| *'catch' конструкт* | да |
| *'if' конструкт* | да |
| *'else' конструкт* | не |
| *'?' конструкт – угњеждено 'if'* | да |
| *'try конструкт* | не |
| *'\|\|' или 'or' конструкт* | да |
| *'&&' или 'and' конструкт* | да |

*Табела В.5 Конструкти укључени у рачунање цикломатичне сложености*

*WMC – Сложеност пондерисаних метода (енгл. Weighted Methods per Class)*

Представља сложеност класе изражену као број независних путања у њој. Израчунава се као збир вредности цикличне сложености (*McCC*) локалних метода.

**Спрегнутост**

Представља меру количине међузависности елемената изворног кода

*CBO – Повезаност објеката (енгл. Coupling Between Object classes)*

Представља број других класа који се директно користе из дате класе. Класе које интензивно користе многе друге класе имају велики утицај на понашање система, па их треба врло пажљиво модификовати и интензивно тестирати.

*NOI – Број одлазних позива (енгл. Number of Outgoing Invocations)*

На нивоу метода и функције представља број директно позваних метода. Ако се метода позове неколико пута, броји се само једном.

На нивоу класе, структуре и уније представља број директно позваних метода других класа, укључујући позивање метода за иницијализацију атрибута. Ако се метода позове неколико пута, броји се само једном.

*RFC – Број одговора класе (енгл. Response set For Class)*

Представља суму броја локалних (тј. ненаслеђених) метода у класи и броја директно позваних других метода помоћу својих метода или иницијализација атрибута.

**Документација**

Представља меру количине коментара и документације елемената изворног кода у систему.

*CLOC – Број коментарисаних линија (енгл. Comment Lines of Code)*

Представља број линија коментара и документације кода. У рачунање нису укључени:

- на нивоу метода и функција :анонимне и локалне класе,
- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена.

*TCLOC – Укупна број коментарисаних линија (енгл. Total Comment Lines of Code)*

Представља број линија коментара и документације кода. У рачунање су укључени:

- на нивоу метода и функција :анонимне и локалне класе,

- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена,
- на нивоу компоненте: подкомпоненте.

**Наслеђивање**

Представља меру различитих аспеката хијерархије наслеђивања у систему

*DIT – Дубина стабла наслеђивања (енгл. Depth of Inheritance Tree)*

Представља дужину пута која води од класе до њеног најудаљенијег претка у стаблу наслеђивања.

*NOC – Број подкласа (енгл. Number of Children)*

Представља број класа, спрега, енумерација и напомена (анотација) које су директно изведене из класе.

**Величина**

Представља мере основних особина система у смислу различитих кардиналности (броја елемената) (нпр. број линија кода, броја класа, број метода).

*LLOC – Логичке линије кода (енгл. Logical Lines of Code)*

Представља број линија кода које нису празне и које нису коментари. У рачунање нису укључени:

- на нивоу методе и функција: анонимне и локалне класе,
- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена.

*LOC – Линије кода (енгл. Lines of Code)*

Представља број линија кода укључујући и празне линије и линије које су коментари. У рачунање нису укључени:

- на нивоу методе и функција: анонимне и локалне класе,

- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена.

*NCL – Број класа (енгл. Number of Classes)*

Представља број класа у простору имена. Класе у потпросторима имена нису укључене у рачунање.

*NEN – Број енумерација (енгл. Number of Enums)*

Представља број енумерација у простору имена. Класе у потпросторима имена нису укључене у рачунање.

*TLLOC – Укупне логичке линије кода (енгл. Total Logical Lines of Code)*

Представља број линија кода које нису празне и које нису коментари. У рачунање су укључени:

- на нивоу методе и функција: анонимне и локалне класе,
- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена,
- на нивоу компоненте: подкомпоненте

*TLOC – Укупне линије кода (енгл. Total Lines of Code)*

Представља број линија кода укључујући и празне линије и линије које су коментари. У рачунање су укључени:

- на нивоу методе и функција: анонимне и локалне класе,
- на нивоу класе, структуре, уније и спреге: угнеждене, анонимне и локалне класе,
- на нивоу простора имена: потпростори имена,
- на нивоу компоненте: подкомпоненте

*TNCL – Укупни број класа (енгл. Total Number of Classes)*

Представља број класа у простору имена. У рачунање су укључени:

- на нивоу простора имена: потпростори имена,
- на нивоу компоненте: подкомпоненте

*TNEN – Укупни број енумерација (енгл. Total Number of Enums)*

Представља број енумерација у простору имена. У рачунање су укључени:

- на нивоу простора имена: потпростори имена,
- на нивоу компоненте: подкомпоненте

### В.2.2 Вредности добијене из алата

**Ниво класе**

СМ алат је препознао 66 класа од чега нису све валидне класе. Увидом у резултате за класе за које алат тврди да садрже једну линију кода, утврђено је да представљају дупликат већ постојеће класе и дати резултати су означени као неисправни. Колона ‘валидно’ означава на које вредности из табеле се ово односи.

| Редни број класе | LCOM5 | WMC | CBO | NOI | RFC | CLOC | TCLOC | DIT | NOC | LLOC | LOC | TLLOC | TLOC | исправно? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | 2 | 111 | 7 | 5 | 5 | 1 | 1 | 0 | 0 | 254 | 514 | 254 | 514 | да |
| *2* | 1 | 23 | 7 | 13 | 13 | 0 | 0 | 0 | 0 | 89 | 177 | 89 | 177 | да |
| *3* | 1 | 8 | 5 | 7 | 7 | 1 | 1 | 0 | 0 | 54 | 124 | 54 | 124 | да |
| *4* | 3 | 22 | 4 | 2 | 2 | 1 | 1 | 1 | 0 | 64 | 124 | 64 | 124 | да |
| *5* | 0 | 4 | 3 | 2 | 2 | 10 | 10 | 0 | 1 | 27 | 95 | 27 | 95 | да |
| *6* | 1 | 41 | 9 | 2 | 2 | 3 | 3 | 0 | 5 | 124 | 247 | 124 | 247 | да |
| *7* | 0 | 3 | 2 | 0 | 0 | 1 | 1 | 0 | 0 | 18 | 50 | 18 | 50 | да |
| *8* | 2 | 152 | 20 | 26 | 26 | 30 | 30 | 0 | 6 | 449 | 1021 | 449 | 1021 | да |
| *9* | 1 | 69 | 9 | 18 | 18 | 9 | 9 | 0 | 0 | 223 | 369 | 223 | 369 | да |
| *10* | 1 | 31 | 9 | 11 | 11 | 37 | 37 | 1 | 0 | 127 | 303 | 127 | 303 | да |
| *11* | 5 | 34 | 7 | 8 | 8 | 5 | 5 | 1 | 0 | 95 | 186 | 95 | 186 | да |
| *12* | 1 | 12 | 5 | 7 | 7 | 4 | 4 | 0 | 0 | 64 | 153 | 64 | 153 | да |
| *13* | 1 | 9 | 5 | 7 | 7 | 0 | 0 | 0 | 0 | 43 | 74 | 43 | 74 | да |
| *14* | 1 | 29 | 8 | 14 | 14 | 1 | 1 | 0 | 0 | 135 | 233 | 135 | 233 | да |
| *15* | 1 | 9 | 5 | 7 | 7 | 0 | 0 | 0 | 0 | 43 | 74 | 43 | 74 | да |
| *16* | 1 | 11 | 6 | 7 | 7 | 0 | 0 | 0 | 0 | 46 | 78 | 46 | 78 | да |
| *17* | 1 | 11 | 5 | 7 | 7 | 0 | 0 | 0 | 0 | 47 | 80 | 47 | 80 | да |
| *18* | 10 | 36 | 33 | 13 | 13 | 109 | 109 | 0 | 0 | 116 | 355 | 116 | 355 | да |
| *19* | 1 | 35 | 9 | 11 | 11 | 9 | 9 | 1 | 0 | 120 | 208 | 120 | 208 | да |
| *20* | 1 | 15 | 4 | 3 | 3 | 0 | 0 | 1 | 0 | 54 | 86 | 54 | 86 | да |
| *21* | 1 | 9 | 4 | 3 | 3 | 0 | 0 | 1 | 0 | 37 | 68 | 37 | 68 | да |
| *22* | 1 | 20 | 5 | 4 | 4 | 0 | 0 | 1 | 0 | 66 | 99 | 66 | 99 | да |
| *23* | 1 | 11 | 4 | 3 | 3 | 0 | 0 | 1 | 0 | 44 | 74 | 44 | 74 | да |
| *24* | 1 | 42 | 5 | 3 | 3 | 0 | 0 | 1 | 0 | 111 | 200 | 111 | 200 | да |

| Редни број класе | LCOM5 | WMC | CBO | NOI | RFC | CLOC | TCLOC | DIT | NOC | LLOC | LOC | TLLOC | TLOC | исправно? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *25* | 1 | 18 | 4 | 2 | 2 | 1 | 1 | 1 | 0 | 107 | 198 | 107 | 198 | да |
| *26* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *27* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *28* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *29* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *30* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *31* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *32* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *33* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *34* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *35* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *36* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *37* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *38* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *39* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *40* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *41* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *42* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *43* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *44* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *45* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *46* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *47* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *48* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *49* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *50* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *51* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *52* | 1 | 17 | 6 | 4 | 4 | 0 | 0 | 1 | 2 | 57 | 76 | 57 | 76 | да |
| *53* | 1 | 17 | 6 | 4 | 4 | 0 | 0 | 1 | 2 | 59 | 76 | 59 | 76 | да |
| *54* | 0 | 14 | 6 | 8 | 8 | 0 | 0 | 1 | 0 | 40 | 77 | 40 | 77 | да |
| *55* | 4 | 61 | 11 | 25 | 25 | 3 | 3 | 2 | 0 | 173 | 193 | 173 | 193 | да |
| *56* | 0 | 14 | 6 | 8 | 8 | 0 | 0 | 1 | 0 | 40 | 77 | 40 | 77 | да |
| *57* | 4 | 57 | 10 | 27 | 27 | 0 | 0 | 2 | 0 | 166 | 173 | 166 | 173 | да |
| *58* | 4 | 57 | 9 | 12 | 12 | 2 | 2 | 1 | 0 | 175 | 189 | 175 | 189 | да |
| *59* | 4 | 57 | 9 | 12 | 12 | 2 | 2 | 1 | 0 | 181 | 193 | 181 | 193 | да |
| *60* | 0 | 5 | 6 | 4 | 4 | 0 | 0 | 1 | 0 | 38 | 60 | 38 | 60 | да |
| *61* | 5 | 73 | 13 | 14 | 14 | 0 | 0 | 1 | 0 | 253 | 283 | 253 | 283 | да |
| *62* | 0 | 3 | 5 | 4 | 4 | 0 | 0 | 1 | 0 | 41 | 64 | 41 | 64 | да |
| *63* | 3 | 54 | 11 | 18 | 18 | 4 | 4 | 2 | 0 | 194 | 214 | 194 | 214 | да |
| *64* | 0 | 3 | 5 | 4 | 4 | 0 | 0 | 1 | 0 | 34 | 56 | 34 | 56 | да |
| *65* | 4 | 56 | 10 | 14 | 14 | 0 | 0 | 2 | 0 | 182 | 198 | 182 | 198 | да |

| Редни број класе | LCOM5 | WMC | CBO | NOI | RFC | CLOC | TCLOC | DIT | NOC | LLOC | LOC | TLLOC | TLOC | исправно? |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *66* | 5 | 75 | 9 | 12 | 12 | 4 | 4 | 1 | 0 | 323 | 336 | 323 | 336 | да |

*Табела В.6 Вредности добијених метрика на нивоу класе*

**Метрике на нивоу енумерација**

Метрике на нивоу енумерација су приказане у табели испод. Тичу се броја линија кода и броја линија коментара.

| Редни број енумерације | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***LLOC*** | 5 | 19 | 4 | 6 | 8 | 4 | 4 | 5 | 7 | 5 | 5 | 5 |
| ***LOC*** | 6 | 21 | 5 | 21 | 15 | 5 | 5 | 8 | 8 | 9 | 6 | 6 |
| ***TLLOC*** | 5 | 19 | 4 | 5 | 8 | 4 | 4 | 5 | 7 | 5 | 5 | 5 |
| ***TLOC*** | 6 | 21 | 5 | 15 | 15 | 5 | 5 | 8 | 8 | 9 | 6 | 6 |

*Табела В.7 Вредности добијених метрика на нивоу енумерација*

**Метрике на нивоу датотеке**

Табела испод приказује резултате метрика на нивоу датотека.

| Редни број датотеке | McCC | CLOC | LLOC | LOC |
|---|---|---|---|---|
| *1* | 1 | 32 | 6 | 59 |
| *2* | 1 | 241 | 57 | 356 |
| *3* | 1 | 91 | 14 | 141 |
| *4* | 1 | 86 | 14 | 145 |
| *5* | 1 | 51 | 21 | 67 |
| *6* | 1 | 146 | 50 | 252 |
| *7* | 1 | 278 | 69 | 463 |
| *8* | 1 | 54 | 9 | 85 |
| *9* | 1 | 130 | 32 | 206 |
| *10* | 1 | 78 | 18 | 142 |
| *11* | 1 | 83 | 17 | 154 |
| *12* | 1 | 98 | 23 | 154 |
| *13* | 1 | 80 | 18 | 142 |
| *14* | 1 | 55 | 7 | 79 |
| *15* | 1 | 28 | 2 | 137 |
| *16* | 1 | 30 | 2 | 38 |
| *17* | 128 | 54 | 392 | 771 |
| *18* | 33 | 30 | 94 | 165 |
| *19* | 1 | 28 | 13 | 48 |
| *20* | 87 | 37 | 198 | 332 |
| *21* | 21 | 31 | 100 | 194 |
| *22* | 54 | 39 | 178 | 276 |
| *23* | 7 | 33 | 47 | 146 |
| *24* | 2 | 31 | 43 | 94 |
| *25* | 9 | 29 | 56 | 113 |
| *26* | 13 | 29 | 62 | 124 |
| *27* | 15 | 28 | 66 | 132 |
| *28* | 1 | 112 | 17 | 158 |
| *29* | 1 | 87 | 20 | 138 |
| *30* | 1 | 60 | 11 | 126 |
| *31* | 1 | 86 | 13 | 130 |
| *32* | 1 | 79 | 12 | 115 |
| *33* | 1 | 78 | 14 | 118 |
| *34* | 1 | 152 | 51 | 263 |
| *35* | 1 | 70 | 14 | 118 |
| *36* | 1 | 74 | 17 | 124 |

| | | | | |
|---|---|---|---|---|
| *37* | 1 | 30 | 6 | 42 |
| *38* | 1 | 28 | 2 | 33 |
| *39* | 3 | 33 | 11 | 52 |
| *40* | 24 | 30 | 86 | 296 |
| *41* | 1 | 28 | 2 | 32 |
| *42* | 1 | 28 | 10 | 45 |
| *43* | 1 | 28 | 10 | 45 |
| *44* | 1 | 28 | 6 | 40 |
| *45* | 1 | 28 | 5 | 37 |
| *46* | 1 | 51 | 9 | 82 |
| *47* | 1 | 51 | 9 | 82 |
| *48* | 1 | 87 | 25 | 143 |
| *49* | 1 | 80 | 24 | 134 |
| *50* | 1 | 44 | 14 | 76 |
| *51* | 1 | 47 | 16 | 80 |
| *52* | 1 | 107 | 42 | 184 |
| *53* | 1 | 86 | 29 | 134 |
| *54* | 1 | 79 | 25 | 127 |
| *55* | 1 | 44 | 26 | 105 |
| *56* | 15 | 26 | 36 | 84 |
| *57* | 15 | 26 | 36 | 84 |
| *58* | 64 | 33 | 151 | 237 |
| *59* | 60 | 30 | 143 | 224 |

| | | | | |
|---|---|---|---|---|
| *60* | 50 | 31 | 126 | 206 |
| *61* | 50 | 31 | 128 | 208 |
| *62* | 67 | 29 | 204 | 312 |
| *63* | 47 | 32 | 157 | 250 |
| *64* | 49 | 28 | 145 | 231 |
| *65* | 64 | 31 | 260 | 360 |
| *66* | 1 | 71 | 10 | 109 |
| *67* | 2 | 174 | 34 | 256 |
| *68* | 1 | 160 | 14 | 216 |
| *69* | 1 | 88 | 18 | 142 |
| *70* | 1 | 76 | 18 | 144 |
| *71* | 1 | 80 | 18 | 150 |
| *72* | 1 | 95 | 14 | 142 |
| *73* | 1 | 102 | 20 | 159 |
| *74* | 1 | 29 | 6 | 40 |
| *75* | 18 | 41 | 54 | 125 |
| *76* | 22 | 42 | 85 | 177 |
| *77* | 29 | 32 | 78 | 148 |
| *78* | 28 | 37 | 96 | 169 |
| *79* | 15 | 29 | 63 | 124 |
| *80* | 22 | 29 | 73 | 140 |
| *81* | 34 | 27 | 85 | 152 |
| *82* | 13 | 32 | 90 | 157 |

*Табела В.8 Вредности добијених метрика на нивоу датотека*

### Метрике на нивоу функција

Алат није препознао ниједну функцију по својим критеријумима

### Метрике на нивоу спрега (енгл. *interface*)

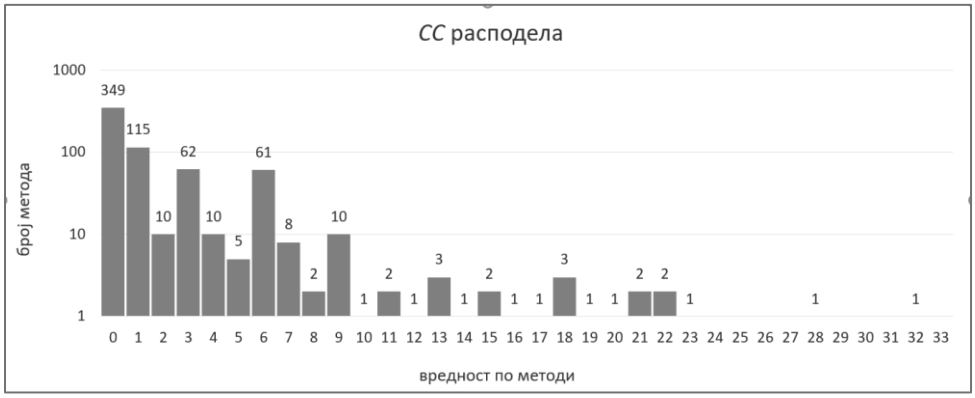
СМ алат је препознао 23 спреге од чега нису све валидне спреге. Увидом у резултате за спреге за које се алат тврди да садрже једну линију кода, утврђено је да представљају дупликат већ постојеће спреге и дати резултати су означени као неисправни у колони ‘исправно’.

| Редни број интерфејса | *WMC* | *CBO* | *CLOC* | *TCLOC* | *DIT* | *NOC* | *LLOC* | *LOC* | *TLLOC* | *TLOC* | *исправно?* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | 0 | 2 | 3 | 3 | 0 | 1 | 17 | 103 | 17 | 103 | да |
| *2* | 0 | 0 | 7 | 7 | 0 | 1 | 11 | 76 | 11 | 76 | да |
| *3* | 0 | 0 | 2 | 2 | 0 | 1 | 4 | 59 | 4 | 59 | да |
| *4* | 0 | 0 | 0 | 0 | 0 | 1 | 7 | 62 | 7 | 62 | да |
| *5* | 0 | 2 | 3 | 3 | 0 | 1 | 16 | 70 | 16 | 70 | да |
| *6* | 0 | 2 | 3 | 3 | 0 | 1 | 12 | 35 | 12 | 35 | да |
| *7* | 0 | 2 | 3 | 3 | 0 | 1 | 13 | 35 | 13 | 35 | да |
| *8* | 0 | 2 | 3 | 3 | 0 | 1 | 13 | 37 | 13 | 37 | да |
| *9* | 0 | 2 | 3 | 3 | 0 | 1 | 13 | 37 | 13 | 37 | да |
| *10* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *11* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *12* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *13* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *14* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *15* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *16* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *17* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *18* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *19* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *20* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *21* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *22* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | не |
| *23* | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 29 | 4 | 29 | да |

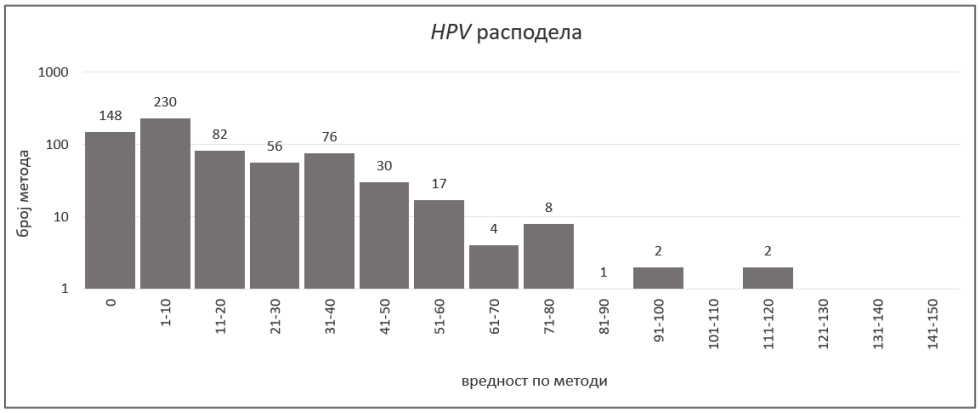
*Табела В.9 Вредности добијених метрика на нивоу спрега*

**Метрике на нивоу метода**

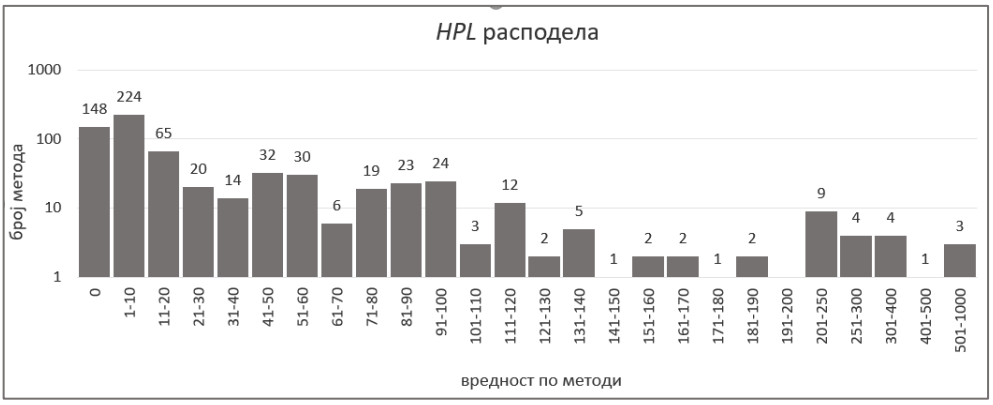
Након анализе алат је избацио резултате метрика за 646 методе. Може се приметити да велики број метода за одређене метрике има вредност нула, то је због чињенице да алат креира као нову методу и додељује вредност нула виртуелним методама, декларацијама, дефиницијама (између осталог правећи копије једне исте методе). У наставку су приказани резултати метрика од интереса у облику хистограма. Нулте вредности нису избачене, нити су посебно анализиране да ли су валидне или не. Оне се не узимају у разматрање код тумачења - од интереса за посматране метрике су горње граничне вредности.



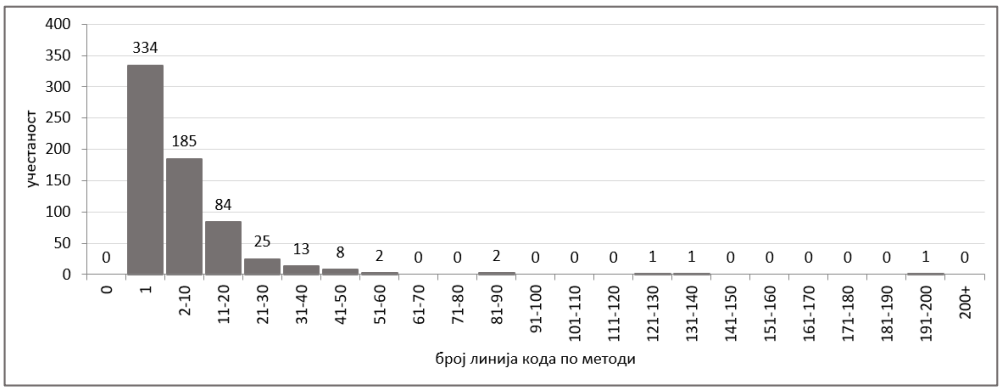

*Слика В.6 Расподела метода у односу на вредност цикличне сложености*




*Слика В.7 Расподела метода у односу на HPV вредност*




*Слика В.8 Расподела метода у односу на HPL вредност*




*Слика В.9 Расподела метода у односу на број линија кода*

CM алат рачуна декларацију у .h датотеци као једну методу са једном логичком линијом кода. Ово не утиче значајно на анализу јер се посматрају горње вредности.

**Метрике на нивоу структура**

Метрике на нивоу структура су приказане у табели испод.

| **Редни број структуре** | ***LCOM5*** | ***WMC*** | ***CBO*** | ***NOI*** | ***RFC*** | ***CLOC*** | ***TCLOC*** | ***DIT*** | ***NOC*** | ***LLOC*** | ***LOC*** | ***TLLOC*** | ***TLOC*** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 9 | 6 | 9 |
| *2* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 4 | 3 | 4 |
| *3* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 4 | 3 | 4 |
| *4* | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 3 | 5 | 3 | 5 |
| *5* | 0 | 0 | 0 | 0 | 0 | 2 | 2 | 0 | 0 | 9 | 12 | 9 | 12 |
| *6* | 0 | 0 | 3 | 0 | 0 | 2 | 2 | 0 | 0 | 7 | 15 | 7 | 15 |
| *7* | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 10 | 7 | 10 |

*Табела В.14 Вредности добијених метрика на нивоу структуре*

**Метрике на нивоу унија**

Алат није препознао ниједну унију по својим критеријумима.

### В.2.3 Вредности израчунатих метрика

Користећи вредности добијене као резултат рада алата, додатно су израчунате следеће метрике, везане за сложеност на нивоу метода.

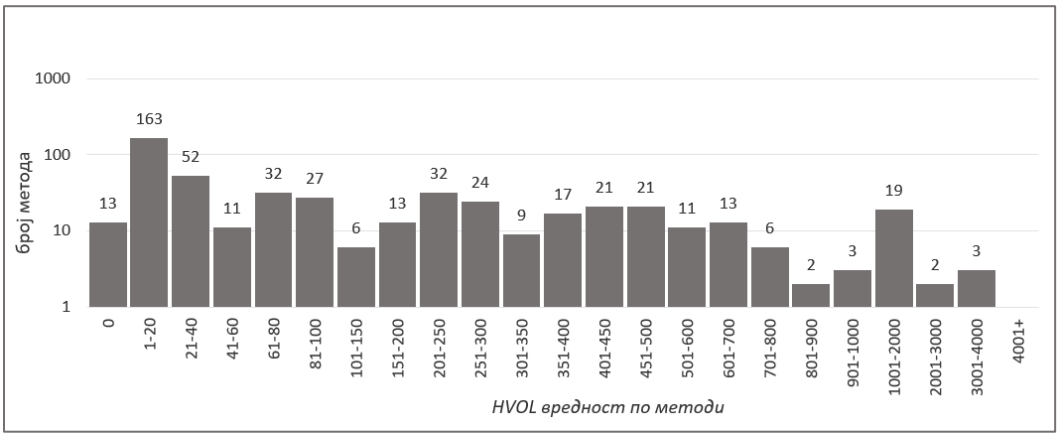
*HVOL - Халстедов обим програма (енгл. Halstead Volume)*

Рачуна се следећом формулом:

$$HVOL = N * \log_2 n$$

где је:

- $N$ – Халстедова дужина програма
- $n$ – Халстедов речник програма




*Слика В.5 Расподела метода у односу на HVOL вредност*

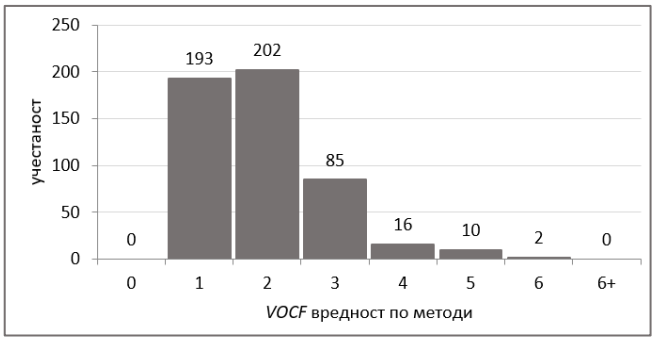
*VOCF – Учестаност речника (енгл. Vocabulary Frequency)*

Рачуна се следећом формулом:

$$VOCF = \frac{N}{n}$$

где је:

- $N$ – Халстедова дужина програма
- $n$ – Халстедов речник програма




*Слика В.10 Расподела метода у односу на VOCF вредност*

## В.3 РЕЗУЛТАТИ МЕРЕЊА – АТРИБУТИ КВАЛИТЕТА СОФТВЕРА

Постоје разни радови и мишљења аутора везаних за рачунање атрибута квалитета софтвера. У наставку су приказане изабране метрике и начини на који су се рачунале у овом раду.

### В.3.1 Преглед метрика

#### Степен сложености одржавања софтвера

Степен сложености одржавања софтвера је иницијално дефинисан у [210] и за његово рачунање се користе традиционалне метрике:

$$MI = 171 - 5.2 * \ln(HVOL) - 0.23 * McCC - 16.2 * \ln(LLOC)$$

где је:

- *MI* – Степен сложености одржавања софтвера (енгл. *Maintainability Index*)
- *HVOL* – Халстедов обим програма,
- *McCC* – циклична сложеност,
- *LLOC* – број логичких линија кода.

Већа вредност метрике указује на бољи степен одржавања софтвера.

Постоји више модификација рачунања овог степена сложености. Модификована формула из [211] користи скалу од 0 до 100 и рачуна се на следећи начин:

$$MIMS = \max(0, (171 - 5.2 * \ln(HVOL) - 0.23 * McCC - 16.2 * \ln(LLOC)) * 100/171)$$

где је:

- *MIMS* – Степен сложености одржавања софтвера – *Microsoft* верзија (енгл. *Maintainability Index -Microsoft version*)
- *HVOL* – Халстедов обим програма,
- *McCC* – циклична сложеност,
- *LLOC* – број логичких линија кода.

Према овој формули, ако је вредност мања од 10, степен сложености одржавања софтвера је низак, а ако је вредност већа од 20, степен сложености одржавања софтвера је прихватљив.

Модификована формула представљена у [212] узима у обзир додатно и густину коментара:

$$MICD = 171 - 5.2 * \ln HVOL - 0.23 * McCC - 16.2 * \ln LLOC + 50 * \sin(\sqrt{2.46 * CD})$$

где је:

- *MICD* – Степен сложености одржавања софтвера – верзија са густином коментара (енгл. *Maintainability Index – Code Density*)
- *HVOL* – Халстедов обим програма,
- *McCC* – циклична сложеност,
- *LLOC* – број логичких линија кода,
- *CD* – густина коментара.

Према ауторима све компоненте чија је вредност већа од 85 имају висок степен одржавања софтвера, а компоненте чија је вредност испод 65 су тешке за одржавање.

**Портабилност**

Према формули коју су предложили аутори [143], портабилност зависи и може се израчунати на основу својих податрибута: модуларности, програмског језика, сложености и процене системске архитектуре. За сваки од атрибута се дефинише одговарајућа тежинска вредност са којом тај атрибут учествује у атрибуту

портабилности. Атрибути се вреднују на скали од 1 до 5 (1 – *веома добро*, 2 – *добро*, 3 – *прихватљиво*, 4 – *потребна побољшања*, 5 - *лоше*). Атрибути и податрибути се даље мапирају и/или разлажу на скуп атрибута и/или одговарајућих метрика који их описују. У односу на вредност тих метрика, аутори су дефинисали табеле које мапирају вредности метрика на поменуту скалу од 1 до 5. Неке од метрика које се налазе у овим таблицама нису прецизно дефинисане, тј. на шта се тачно односе вредности датих метрика (на класу, методу, модул, просечну вредност и сл.) и како се рачунају. Управо због тога, за потребе процене решења у овом раду ће се посматрати најбоља и најгора добијена вредност атрибута у односу на присутне варијације.

Пре самог израчунавања вредности портабилности, аутори представљају два упита која морају да буду задовољена, у супротном портабилност се рангира са најгорим резултатом:

- *Да ли постоји критична зависност од хардвера, а да подршка не постоји на циљној платформи?*
- *Да ли се у софтверу користи језик специфичан за платформу?*

Након успешног проласка ових упита, портабилност се израчунава следећом формулом:

$$P = Oa(0.2) + Mo(0.3) + Pl\ (0.2) + Co\ (0.3)$$

где је:

() – вредност у загради представља тежински фактор за дати атрибут

*Оа* – процена архитектуре система, чија вредност се добија одговарањем на низ упита:

- *Да ли се користи виртуална машина или неки од примарно коришћених интерпретерских језика (Java, Python, Smalltak)*?
- Ако је одговор на први упит не, постављају се следећи упити од којих сваки једнако доприноси коначном резултату:
    - *Колико добро је део изворног кода специфичан за платформу одвојен од заједничког дела за све платформе*?

- *Коришћење стандардизованих и широко доступних АПИ-ја (нпр. OpenGL), обраћање пажње на руковање меморијом (редослед бајтова, паковање структура и слично)*.
- *Коришћење библиотека које имају подршку на више платформи*.

*Pl* – вредност програмског језика. У односу на програмски језик који се користи добија се одговарајућа вредност на скали од 1 до 5.

*Mo* – вредност модуларности се добија следећом формулом:

$$Mo = Mn(0.5) + Ms(0.5)$$

где *Mn* представља број модула, а *Ms* величину модула.

*Co* – вредност комплексности система се добија на следећи начин:

$$Co = Vg(0.5) + Cp(0.5)$$

где *Vg* представља цикличну сложеност, а *Cp* повезаност (енгл. *coupling*).

За потребе мапирања вредности метрика на поменуту скалу, а у сврху израчунавања атрибута портабилности, користи се табела из [143], која је приказана испод.

| Оцена | *Pl* | *Oa* | *Модули* | | | *Vg* |
|---|---|---|---|---|---|---|
| | | | *Mn* | *Cp* | *Ms* | |
| 1 | *Java, C, C++, Python, Perl* | =1 | >20 | <50% | 0-200 | <3 |
| 2 | | 1-2 | 15-20 | 51-60% | 201-300 | 3-6 |
| 3 | други језици високог нивоа | 2-3 | 11-15 | 61-75% | 301-400 | 7-10 |
| 4 | | 3-4 | 5-10 | 76-90% | 401-500 | 11-14 |
| 5 | језици специфични за платформу | 4-5 | 1-5 | >90% | >500 | 15-20 |

*Табела В.10 Портабилност – мапирање метрика на циљну скалу*

**Проширивост**

За израчунавање проширивости, користи се исти принцип као и за портабилност - предложен од стране аутора [143]. Проширивост се дефинише као зависност од следећих атрибута: модуларност, сложеност и процена системске архитектуре. Израчунава се по формули:

$$E = Mo(0.34) + Co(0.33) + Oa(0.33)$$

где *Mo*, *Co* и *Oa* представљају податрибуте који су дефинисани на исти начин као и у случају портабилности.

За потребе мапирања вредности метрика на поменуту скалу, а у сврху израчунавања атрибута проширивости, користи се табела из [143], која је приказана испод.

| Оцена | *Mn* | *Ms* | *WMC* | *Vg* | *LOCM* | *CBO* | *Oa* |
|---|---|---|---|---|---|---|---|
| 1 | >20 | 0-200 | 1-2 | <3 | <90% | <2 | 1 |
| 2 | 15-20 | 201-300 | 3-6 | 3-6 | 60-89% | 2-4 | 2 |
| 3 | 11-15 | 301-400 | <14 | 7-10 | 40-59% | 4-6 | 3 |
| 4 | 5-10 | 401-500 | 14-20 | 11-14 | 25-39% | 6-8 | 4 |
| 5 | 1-5 | >500 | >20 | 15-20 | <25% | >8 | 5 |

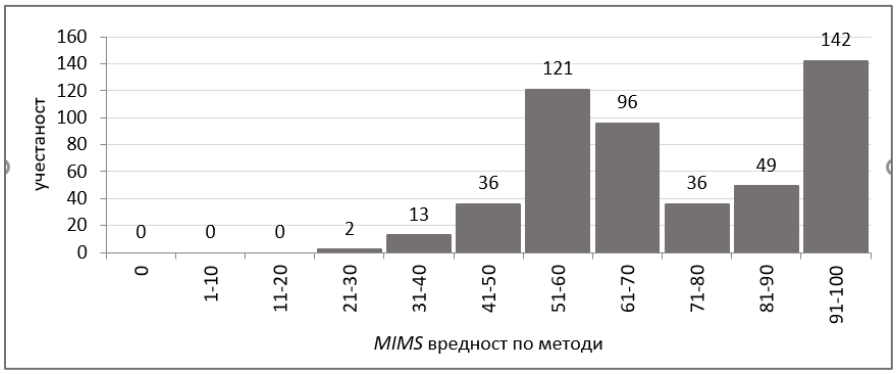
*Табела В.11 Проширивост – мапирање метрика на циљну скалу*

### В.3.2 Вредности израчунатих метрика

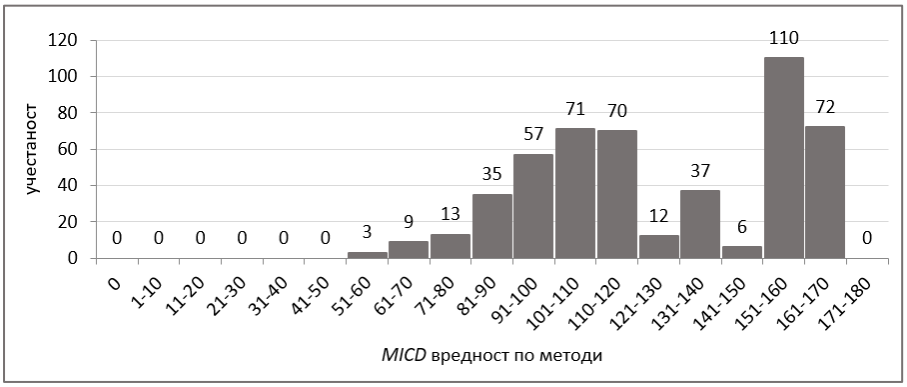
#### Степен сложености одржавања софтвера

На графицима испод приказане се вредности добијене за сваку од три формуле за израчунавање степена сложености одржавања софтвера. Улазни параметри у формуле су вредности добијене CM алатом.


*Слика В.11 Расподела метода у односу на MI вредност*




*Слика В.12 Расподела метода у односу на MIMS вредност*




*Слика В.13 Расподела метода у односу на MICD вредност*

**Портабилност**

*Процена архитектуре система - Oa*

Степен архитектуре система се одређује на основу упита и одговора на саме упите. Узевши у обзир да не користимо виртуалну машину, вредност се формира на основу вредности упита 2, 3 и 4. Сваки од упита учествује са истим тежинским фактором и на основу одговора добијамо да је *Oa* = 1.66 .

| Упит | Коментар | Одговор |
|---|---|---|
| ***У1:*** *Коришћење виртуалне машине / интерпретерских језика* | Као део решења не користи се виртуална машина, иако је решење предвиђено да може да се извршава и над хипервизорима. | Не |
| ***У2:*** *Одвојеност дела специфичног за платформу* | Предвиђен је слој за апстракцију платформе, у коме се реализују ствари које су специфичне за платформу (оперативни систем, хардвер). Главни део користи интерфејсе који се реализују у слоју за апстракцију. | 1 – врло добро |

| Упит | Коментар | Одговор |
|---|---|---|
| ***У3:*** *Коришћење стандардизованих АПИ-ја, руковање меморијом* | Користе се, а и предвиђено је коришћење стандардизованих АПИ-ја (нпр. *OpenGL* у јединицама обраде) који се може користити на више платформи. Руковање меморијом је апстраховано у самом слоју апстракције (уместо коришћења неког доступног АПИ-ја), тако да горњи део средњег слоја не води рачуна о томе. | 2 – добро |
| ***У4:*** *Коришћење библиотека доступних на више платформи* | Коришћење широко доступних библиотека и спрега у јединицама обраде. Додатно, у слоју апстракције платформе се такође користе широко доступне библиотеке и спреге (нпр. pthread) како би се могао искористити на више платформи. | 2 - добро |

*Табела В.12 Упити за процену архитектуре система*

*Вредност програмског језика - Pl*

У раду се користи *C++* програмски језик, што значи да је *Pl* = 1.

*Модуларност* - Mo

За потребе израчунавања, под модулом ће се посматрати класа, тј. класе које је препознао CM алат, искључујући декларације класа. Укупан број класа које алат пријављује са искљученим сувишним класама је 40 (означићемо га као скуп А), а ако искључимо и класе које припадају реализацији слоја за апстракцију платформе, број класа је 29 (скуп Б). Увидом у табелу мапирања, добијамо да је *Mn* = 1.

Просечну величину модула рачунамо као број линија кода по модулу или алтернативно као број логичких линија кода по модулу. Табела испод приказује величину модула у односу на посматране модуле и врсте линија кода. Увидом у табелу мапирања, за све варијанте добијамо да је *Ms* = 1.

| Посматрани модули | *LLOC* по модулу | *LOC* по модулу |
|---|---|---|
| *Скуп А* | 113 | 186 |
| *Скуп Б* | 123 | 196 |

*Табела В.13 Величина модула*

За степен модуларности добија се вредност 1.

*Комплексност - Co*

Степен комплексности зависи од цикличне сложености и од повезаности. Аутори нису најјасније дефинисали у односу на шта се гледају вредности у табели мапирања.

У случају цикличне сложености помиње се да се односи на секцију изворног кода. За секцију у нашем случају узели смо методу. Просечна вредност цикличне сложености за методе је 4.3, а медијан вредност је 3. Увидом у таблицу мапирања, добијамо да је *Vg* = 2 (добро).

За спрегнутост аутори истичу да је једна од софтверских метрика којима се може израчунати спрегнутост и број одговора класе. У таблици мапирања, за спрегнутост имамо процентуалне вредности, док нам CM алат даје број одговора класе, спрегнутост између класа објеката и број одлазних позива. Резултати метрика из CM алата указују да су вредности броја одговора на класу, броја одлазних позива и спрегнутости између класа објеката на нижим нивоима вредности у односу на границе које постављају други аутори. За спрегнутост ћемо узети најбољу и најгору вредност – за најбољу вредност узимамо 1 (метрике указују да је спрегнутост ниска, а критеријум је мање од 50% за овај степен). За најгору вредност узимамо вредност 5.

Вредност за степен комплексности који се добија је 1.5 у најбољем случају, тј. 3.5 у најгорем случају.

*Табела B.14* приказује вредности степена ових атрибута, укључујући и портабилност у најбољем и најгорем случају.

| | *Oa* | *Pl* | *Mo* | *Co* | ***Po*** |
|---|---|---|---|---|---|
| *Најбоља вредност* | 1.66 | 1 | 1 | 1.5 | **1.28** |
| *Најгора вредност* | 2 | 1 | 1 | 3.5 | **1.95** |

*Табела B.14 Вредност степена портабилности*

**Проширивост**

Степен проширивости зависи од модуларности, степена архитектуре система и од комплексности. Атрибути за модуларност и степен архитектуре система се рачунају и мапирају на исти начин као што је био случај код портабилности. Што

се тиче комплексности, овај атрибут такође зависи од цикличне сложености и од спрегнутости. Вредности цикличне сложености за мапирање су исте као и код портабилности. У таблици за мапирање појављују се и следеће метрике: спрегнутост између класа објеката, сложеност пондерисаних метода.

Ако за меру спрегнутости узмемо спрегнутост између класа објеката, просечна вредност за спрегнутост по класи је 7.4 (рачунајући резултате добијене CM алатом). На основу мапирања у табели добијамо да је *Cp* = 4. Ако посматрамо сложеност пондерисаних метода, просечна вредност по класи је већа од 20, што нам даје лошу вредност након мапирања, тј. 5, док је за цикличну сложеност вредност мапирања 2. На основу ових вредности добијамо најбољу и најгору вредност за степен комплексности и за проширивост. Добијене вредности су приказане у табели испод. Закључак је да је степен проширивости на добром нивоу – у опсегу између прихватљивог и веома доброг.

| | *Oa* | *Mo* | *Co* | ***E*** |
|---|---|---|---|---|
| **Најбоља вредност** | 1.66 | 1 | 3 | **1.88** |
| **Најгора вредност** | 2 | 1 | 4.5 | **2.5** |

*Табела В.15 Вредност степена проширивости*

# ЛИТЕРАТУРА

[1] „ISO 11898 Part 1-4, Road vehicles — Controller area network (CAN),“ ISO IEC, 2016.

[2] „ISO 17458 Part 1-5, Road vehicles — FlexRay communications system,“ ISO, 2013.

[3] „ISO 17987 Part 1-7, Road vehicles — Local Interconnect Network (LIN),“ ISO, 2016.

[4] S. Fürst, „Challenges in the design of automotive software,“ у *Design, Automation and Test in Europe, DATE,* Dresden, 2010.

[5] M. Milosevic, M. Z. Bjelica, T. Maruna и N. Teslic, „Software Platform for Heterogeneous In-Vehicle Environments,“ *IEEE Transactions on Consumer Electronics,* т. 64, бр. 2, pp. 213-221, Maj 2018.

[6] M. Milosevic, N. Kolarovic, N. Zmukic, O. Djekic и M. Kovacevic, „AMV - Automotive Machine Vision Middleware,“ Техничко решење развијено у оквиру пројекта технолошког развоја ТР 32031, циклус истраживања у периоду 2016, 2016.

[7] M. Milosevic, M. Pranjkic, M. Kovacevic, Z. Lukac и O. Djekic, „AMV workload balancing,“ Техничко решење развијено у оквиру пројеката технолошког развоја ТР 32031, циклус истраживања у периоду 2011-2017, 2017.

[8] R. K. Jurgen, Autonomous Vehicles for Safer Driving, SAE International, 2013.

[9] H.-U. Michel, D. Kaule и M. Salfer, „Virtualisierung: Vision einer intelligenten Vernetzung.,“ *Elektronik automotive,* т. 04/2012, pp. 28-32, 2012 .

[10] „Audi A8 – Central driver assistance controller (zFAS),“ Audi Technology Portal, Jul 2017. [На мрежи]. Доступно: https://www.audi-technology-

portal.de/en/electrics-electronics/driver-assistant-systems/audi-a8-central-driver-assistance-controller-zfas. [Последњи приступ Септембар 2022].

[11] V. M. Navale, K. Williams, A. Lagospiris, M. Schaffert и M.-A. Schweiker, „(R)evolution of E/E Architectures," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.,* т. 8, pp. 282-288, 2015.

[12] „ISO 21806 Part 1-15, Road vehicles — Media Oriented Systems Transport (MOST)," ISO, 2020.

[13] A. Patel, M. Daftedar, M. Shalan и W. M. El-Kharashi, „Embedded Hypervisor Xvisor: A Comparative Analysis," у *23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Turku, 2015.

[14] „32-bit TriCore™ AURIX™– TC3xx," Infineon Technologies AG, [На мрежи]. Доступно: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc3xx/. [Последњи приступ Септембар 2022].

[15] Texas Instruments Incorporated, „Hercules™ Arm® Cortex®-R MCUs for functional safety," [На мрежи]. Доступно: https://www.ti.com/microcontrollers/hercules-safety-mcus/overview.html. [Последњи приступ Септембар 2022].

[16] Freescale Semiconductor, „MPC5643L Microcontroller Reference Manual," [На мрежи]. Доступно: https://www.nxp.com/files-static/32bit/doc/ref_manual/MPC5643LRM.pdf. [Последњи приступ Септембар 2022].

[17] Renesas Electronics Corporation, „RH850/P1x-C Application Note," [На мрежи]. Доступно: https://www.renesas.com/eu/en/doc/products/mpumcu/apn/rh850/001/R01AN4327ED0300_RH850_P1x-C.pdf. [Последњи приступ Septembar 2020].

[18] „SPC5 32-bit Automotive MCUs," STMicroelectronics, [На мрежи]. Доступно: https://www.st.com/en/automotive-microcontrollers/spc5-32-bit-automotive-mcus.html. [Последњи приступ Septembar 2022].

[19] Intel, „Cyclone V Device Overview,“ [На мрежи]. Доступно: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-v/cv_51001.pdf. [Последњи приступ Септембар 2022].

[20] Xilinx, „Zynq UltraScale+ Heterogeneous MPSoC,“ Xilinx, [На мрежи]. Доступно: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html. [Последњи приступ Септембар 2022].

[21] NVIDIA , „NVIDIA Tegra 4 Family GPU Architecture,“ 2013. [На мрежи]. Доступно: https://www.nvidia.com/docs/IO/116757/Tegra_4_GPU_Whitepaper_FINALv2.pdf. [Последњи приступ Септембар 2022].

[22] Mobileye, „The Evolution of EyeQ,“ Mobileye, [На мрежи]. Доступно: https://www.mobileye.com/our-technology/evolution-eyeq-chip/. [Последњи приступ Септембар 2022].

[23] Qualcomm Technologies, Inc., „Snapdragon 820 Automotive Platform,“ Qualcomm Technologies, Inc., [На мрежи]. Доступно: https://www.qualcomm.com/products/snapdragon-820-automotive-platform. [Последњи приступ Септембар 2022].

[24] SAMSUNG, „The Samsung Exynos Auto solutions,“ SAMSUNG, [На мрежи]. Доступно: https://www.samsung.com/semiconductor/minisite/exynos/application/automotive/. [Последњи приступ Септембар 2020].

[25] Texas Instruments, „TDAx ADAS SoCs,“ [На мрежи]. Доступно: https://www.ti.com/processors/automotive-processors/tdax-adas-socs/overview.html. [Последњи приступ Октобар 2020].

[26] Ambarella, Inc., „Ambarella Introduces CV2 4K Computer Vision SoC with CVflow™ Architecture and Stereovision,“ 2018. [На мрежи]. Доступно: http://investor.ambarella.com/news-releases/news-release-details/ambarella-introduces-cv2-4k-computer-vision-soc-cvflowtm. [Последњи приступ Септембар 2022].

[27] CEVA, Inc., „CEVA-XM6 Product Note,“ [На мрежи]. Доступно: https://www.ceva-dsp.com/resource/xm6-product-note/. [Последњи приступ Септембар 2022].

[28] NXP Semiconductors, „i.MX 8 Series Applications Processors,“ [На мрежи]. Доступно: https://www.nxp.com/products/processors-and-microcontrollers/arm-processors/i-mx-applications-processors/i-mx-8-processors:IMX8-SERIES. [Последњи приступ Септембар 2022].

[29] Renesas Electronics Corporation, „R-Car H3,“ Renesas Electronics Corporation, [На мрежи]. Доступно: https://www.renesas.com/in/en/solutions/automotive/soc/r-car-h3.html. [Последњи приступ Септембар 2022].

[30] „Next-gen Audi A8 to feature MIB2+, series debut of zFAS domain controller, Mobileye image recognition with deep learning; Traffic Jam Pilot,“ 2017. [На мрежи]. Доступно: https://www.greencarcongress.com/2017/01/20170105-audia8.html. [Последњи приступ Септембар 2022].

[31] Intel Corporation, „Intel®GO™ automated driving solutions,“ 2017. [На мрежи]. Доступно: https://newsroom.intel.com/newsroom/wp-content/uploads/sites/11/2017/01/intel-go-product-brief.pdf. [Последњи приступ Септембар 2022].

[32] ZF Friedrichshafen AG, „Autonomous Driving: The Formula for Success,“ 2019. [На мрежи]. Доступно: https://www.zf.com/site/magazine/en/articles_13440.html. [Последњи приступ Септембар 2020].

[33] TTTech Auto, „RazorMotion - The next level of development and evaluation is here,“ [На мрежи]. Доступно: https://www.tttech-auto.com/wp-content/uploads/TTTech-Automotive_RazorMotion-1.pdf. [Последњи приступ Septembar 2022].

[34] TTTech Computertechnik AG, „TTA Drive: Autonomous Driving with TTTech's ADAS ECU,“ 2016. [На мрежи]. Доступно: https://www.youtube.com/watch?v=zAteVcOzpyc. [Последњи приступ Septembar 2022].

[35] S. Wright, „Autonomous cars generate more than 300 TB of data per year,“ Tuxera, 2021. [На мрежи]. Доступно: https://www.tuxera.com/blog/autonomous-cars-300-tb-of-data-per-year/. [Последњи приступ Септембар 2022].

[36] „ISO 11519-3: Road vehicles — Low-speed serial data communication — Part 3: Vehicle area network (VAN),“ ISO, 1994.

[37] „K-Line - A Component of the Automobile Industry,“ [На мрежи]. Доступно: https://www.kunbus.com/k-line.html. [Последњи приступ Септембар 2020].

[38] „Inter Equipment Bus,“ [На мрежи]. Доступно: http://www.interfacebus.com/Design_Connector_IEbus.html. [Последњи приступ Септембар 2022].

[39] C. Ciocan, „The Domestic Digital Bus system (D2B)-a maximum of control convenience in audio video,“ *IEEE Transactions on Consumer Electronics,* pp. 619-622, 1990.

[40] Broadcom Corporation, „OPEN Alliance BroadR-Reach® (OABR) Physical Layer Transceiver Specification For Automotive Applications,“ 2014. [На мрежи]. Доступно: http://www.ieee802.org/3/1TPCESG/public/BroadR_Reach_Automotive_Spec_V3.2_w_o.pdf. [Последњи приступ Септембар 2022].

[41] A. Festag, R. Baldessari, W. Zhang, L. Le, A. Sarma и M. Fukukawa, „CAR-2-X Communication for Safety and Infotainment in Europe,“ *NEC TECHNICAL JOURNAL,* т. 3, pp. 21-26, 2008.

[42] VMware, „Workstation Pro,“ [На мрежи]. Доступно: https://www.vmware.com/products/workstation-pro.html. [Последњи приступ Септембар 2022].

[43] VMware, „VMware Workstation Player,“ [На мрежи]. Доступно: https://www.vmware.com/products/workstation-player.html. [Последњи приступ Септембар 2022].

[44] Oracle, „VirtualBox,“ [На мрежи]. Доступно: https://www.virtualbox.org/. [Последњи приступ Септембар 2022].

[45] „Xvisor: eXtensible Versatile hypervISOR,“ [На мрежи]. Доступно: http://xhypervisor.org/. [Последњи приступ Септембар 2022].

[46] VMware, „VMware ESXi: The Purpose-Built Bare Metal Hypervisor,“ [На мрежи]. Доступно: https://www.vmware.com/products/esxi-and-esx.html. [Последњи приступ Септембар 2022].

[47] „Linux KVM - Kernel Virtual Machine,“ [На мрежи]. Доступно: https://www.linux-kvm.org/page/Main_Page. [Последњи приступ Септембар 2022].

[48] The Linux Foundation, „Xen Project,“ [На мрежи]. Доступно: https://xenproject.org/. [Последњи приступ Септембар 2022].

[49] Microsoft, „Hyper-V Technology Overview,“ [На мрежи]. Доступно: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-technology-overview. [Последњи приступ Септембар 2022].

[50] G. Heiser и B. Leslie, „The OKL4 Microvisor: Convergence Point of Microkernels and Hypervisors,“ у *Proceedings of the 1st ACM SIGCOMM Asia-Pacific Workshop on Systems, ApSys 2010,* New Delhi, India, 2010.

[51] Green Hills Software , „INTEGRITY Multivisor Secure Virtualization,“ [На мрежи]. Доступно: https://www.ghs.com/products/rtos/integrity_virtualization.html. [Последњи приступ Септембар 2022].

[52] Arm Limited, „The Virtualization Extensions,“ [На мрежи]. Доступно: https://developer.arm.com/documentation/ddi0406/c/System-Level-Architecture/The-System-Level-Programmers--Model/The-Virtualization-Extensions. [Последњи приступ Септембар 2022].

[53] Intel Corporation, „Intel Virtualization Technology FlexMigration Application Note,“ 2012.

[54] Intel Corporation, „Intel Virtualization Technology for Directed I/O - Architecture Specification,“ 2021.

[55] IBM, „IBM z/VM virtualization software,“ [На мрежи]. Доступно: https://www.ibm.com/it-infrastructure/z/zvm. [Последњи приступ Септембар 2022].

[56] VirtualLogix VLX Inc, „VirtualLogix VLX Virtualization Software,“ [На мрежи]. Доступно: https://www.design-reuse.com/news/24371/arm-cortex-a15-virtualization-software.html. [Последњи приступ Септембар 2022].

[57] VDA QMC, „Automotive SPICE®,“ 2018. [На мрежи]. Доступно: http://www.automotivespice.com/. [Последњи приступ Септембар 2022].

[58] ISO, „ISO/IEC 15504 Parts 1-5:2012 Information technology - Process assessment,“ ISO, 2012.

[59] „ISO 26262, Road vehicles — Functional safety,“ ISO, 2018.

[60] „MISRA-C: Guidelines for the use of the C language in critical systems,“ Motor Industry Software Reliability Association, Warwickshire, UK, 2004.

[61] „ISO/PAS 21448 Road vehicles — Safety of the intended functionality,“ ИСО, 2019.

[62] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham и S. Savage, „Experimental Security Analysis of a Modern Automobile,“ у *2010 IEEE Symposium on Security and Privacy*, Berkeley/Oakland, CA,, 2010.

[63] T. Hoppe, S. Kiltz и J. Dittmann, „Security Threats to Automotive CAN Networks - Practical Examples and Selected Short-Term Countermeasures,“ у *SAFECOMP*, 2008.

[64] D. K. Oka, U. E. Larson, F. Picasso и E. Jonsson, „A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay,“ у *Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems*, Genova, Italy, 2008.

[65] B. Glas, J. Guajardo, H. Hacioglu, M. Ihle, K. Wehefritz и A. Yavuz, „Signal-based Automotive Communication Security and Its Interplay with Safety Requirements,“ у *Proceedings of the Embedded Security in Cars Conference*, 2012.

[66] M. Wolf, A. Weimerskirch и K. Lemke, „Secure In-Vehicle Communication,“ у *Embedded Security in Cars: Securing Current and Future Automotive IT Applications,* Springer Berlin Heidelberg, 2006, pp. 95-109.

[67] Q. Zou, W. K. Chan, K. C. Gui, Q. Chen, K. Scheibert, L. Heidt и E. Seow, „The Study of Secure CAN Communication for Automotive Applications,“ SAE International, 2017.

[68] H. Kobayashi, C. Konno, M. Kayashima и M. Nakano, „Approaches for Vehicle Information Security,“ IPA, 2013.

[69] „ISO/IEC J3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems,“ ISO, 2016.

[70] „ISO/SAE DIS 21434 Road vehicles — Cybersecurity engineering,“ ИСО, 2021.

[71] J. A. McCall, P. K. Richards и G. F. Walters, *Factors in Software Quality,* 1977.

[72] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. McLeod и M. Merritt, *Characteristics of Software Quality,* North Holland, 1978.

[73] R. G. Dromey, „A model for software product quality,“ *IEEE Transactions on Software Engineering,* бр. no. 2, pp. 146-163, 1995.

[74] R. B. Grady и D. Caswell, *Software metrics: Establishing a company-wide program,* Englewood Cliffs, NJ: Prentice-Hall, 1987.

[75] IEEE, *Standard for Software Maintenance,* Software Engineering Standards Subcommittee of the IEEE Computer Society., 1993.

[76] ISO/IEC., *ISO/IEC Standard 9126: Software product evaluation—quality characteristics,* Geneva: International Organization for Standardization and the International Electrotechnical Commission, 1991.

[77] ISO/IEC, *ISO/IEC Standard 25010: Systems and software engineering—System and software quality models,* Geneva: International Organization for Standardization and the International Electrotechnical Commission, 2011.

[78] K. M. Adams, „Introduction to Non-functional Requirements,“ у *Nonfunctional Requirements in Systems Analysis and Design,* Springer, 2015, pp. 45-72.

[79] D. Samadhiya, S.-H. Wang и D. Chen, „Quality models: Role and value in software engineering,“ у *2010 2nd International Conference on Software Technology and Engineering*, San Juan, PR, USA, 2010.

[80] iso25000.com, „ISO/IEC 25010,“ 2022. [На мрежи]. Доступно: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010. [Последњи приступ Септембар 2022].

[81] S. Bunzel, „AUTOSAR – the Standardized Software Architecture,“ *Informatik-Spektrum*, т. 34, pp. 79-83, 2010.

[82] S. Fürst и M. Bechter, „AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR Adaptive Platform,“ у *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, Toulouse, 2016.

[83] „Japan Automotive Software Platform and Architecture,“ [На мрежи]. Доступно: https://www.jaspar.jp/en. [Последњи приступ Септембар 2022].

[84] BlackBerry QNX, „QNX Platform for ADAS 2.0,“ 2017. [На мрежи]. Доступно: https://blackberry.qnx.com/content/dam/qnx/products/adas/adas-product-brief.pdf. [Последњи приступ Септембар 2022].

[85] Siemens, „Capital VSTAR Embedded Software,“ [На мрежи]. Доступно: https://www.plm.automation.siemens.com/global/en/products/electrical-electronics/ecu-software-integration.html. [Последњи приступ Септембар 2022].

[86] Elektrobit, „EB tresos product line,“ [На мрежи]. Доступно: https://www.elektrobit.com/products/ecu/eb-tresos/. [Последњи приступ Септембар 2022].

[87] Vector, „MICROSAR Product Information,“ [На мрежи]. Доступно: https://assets.vector.com/cms/content/products/microsar/Docs/MICROSAR_ProductInformation_EN.pdf. [Последњи приступ Септембар 2022].

[88] TTTech Auto AG, „MotionWise: The safety mastermind for automated driving and beyond,“ [На мрежи]. Доступно: https://www.tttech-auto.com/products/automated-driving/motionwise/. [Последњи приступ Септембар 2022].

[89] Elektrobit, „EB corbos product line,“ [На мрежи]. Доступно: https://www.elektrobit.com/products/ecu/eb-corbos/. [Последњи приступ Септембар 2022].

[90] BlackBerry QNX, „QNX Car Platform for Infotainment,“ 2017. [На мрежи]. Доступно: https://blackberry.qnx.com/content/dam/qnx/products/qnxcar/qnx-car-infotaintment-product-brief.pdf. [Последњи приступ Септембар 2022].

[91] Mentor, „Automotive OPTstack™ Middleware,“ [На мрежи]. Доступно: https://www.mentor.com/embedded-software/xse-automotive/optstack. [Последњи приступ Септембар 2020].

[92] Elektrobit, „Connected vehicle,“ [На мрежи]. Доступно: https://www.elektrobit.com/products/eb-connected-vehicle/. [Последњи приступ Септембар 2022].

[93] Real-Time Innovations., „RTI Connext Drive,“ Real-Time Innovations, [На мрежи]. Доступно: https://www.rti.com/drive. [Последњи приступ Септембар 2022].

[94] Green Hills Software, „INTEGRITY RTOS - The most reliable & secure operating system,“ [На мрежи]. Доступно: https://www.ghs.com/products/rtos/integrity.html. [Последњи приступ Септембар 2022].

[95] NVIDIA Corporation, „NVIDIA DRIVE - Software,“ [На мрежи]. Доступно: https://developer.nvidia.com/drive/drive-software. [Последњи приступ Септембар 2022].

[96] Texas Instruments Incorporated, „Processor SDK for TDAx ADAS SoCs - Linux and TI-RTOS Support,“ Texas Instruments Incorporated, [На мрежи]. Доступно: https://www.ti.com/tool/PROCESSOR-SDK-TDAX. [Последњи приступ Септембар 2022].

[97] Qualcomm Technologies, Inc., „Qualcomm Accelerates Autonomous Driving with New Platform – Qualcomm Snapdragon Ride,“ Qualcomm Technologies, Inc., 2020. [На мрежи]. Доступно: https://www.qualcomm.com/news/releases/2020/01/06/qualcomm-accelerates-

autonomous-driving-new-platform-qualcomm-snapdragon. [Последњи приступ Септембар 2022].

[98] Linux Foundation, „Automotive Grade Linux: Unified Code Base,“ 2016. [На мрежи]. Доступно: https://www.automotivelinux.org/software/unified-code-base/. [Последњи приступ Септембар 2022].

[99] „GENIVI: Beyond Linux IVI and Into the Connected Vehicle.,“ [На мрежи]. Доступно: https://www.covesa.global/. [Последњи приступ Септембар 2022].

[100] „Open robinos specification: architecture, mechanisms, interfaces, and testing,“ 2017. [На мрежи]. Доступно: https://twittertechnews.com/wp-content/uploads/2018/07/open_robinos_specification_1.1.0.pdf. [Последњи приступ Септембар 2020].

[101] Open Source Robotics Foundation, Inc., „ROS 2 Design,“ Open Source Robotics Foundation, Inc., [На мрежи]. Доступно: https://design.ros2.org/. [Последњи приступ Септембар 2022].

[102] The Autoware Foundation, „Autoware.Auto,“ [На мрежи]. Доступно: https://autowarefoundation.gitlab.io/autoware.auto/AutowareAuto/. [Последњи приступ Септембар 2022].

[103] A. Groll, J. Holle, C. Ruland, M. Wolf, T. Wollinger и F. Zweers, „OVERSEE a secure and open communication and runtime platform for innovative automotive applications,“ у *7th Embedded Security in Cars Conf. (ESCAR),* 2009.

[104] S. Liu, J. Tang, Z. Zhang и J.-L. Gaudiot, „CAAD: Computer Architecture for Autonomous Driving,“ arXiv, 2017.

[105] X. Zhou, K. Wang, L. Zhu, S. Huang, G. Han и J. Yu, „Development of Vehicle Domain Controller Based on Ethernet,“ *Journal of Physics: Conference Series,* т. 1802, бр. 2, p. 022065, 2021.

[106] M. Z. Bjelica и Z. Lukac, „Central Vehicle Computer Design: Software Taking Over,“ *IEEE Consumer Electronics Magazine,* т. 8, бр. 6, pp. 84-90, 2019.

[107] M. Dendaluce Jahnke, „Real-time multi-domain optimization controller for multi-motor electric vehicles using automotive-suitable methods and

heterogeneous embedded platforms,“ Докторска теза, Faculty of Engineering, Bilbao, 2019.

[108] P. Mundhenk, G. Tibba, L. Zhang, F. Reimann, D. Roy и S. Chakraborty, „Dynamic Platforms for Uncertainty Management in Future Automotive E/E Architectures: Invited,“ у *Proceedings of the 54th Annual Design Automation Conference 2017*, Austin, TX, USA, 2017.

[109] K. Becker, J. Frtunikj, M. Felser, L. Fiege и C. Buckl, „RACE RTE: A Runtime Environment for Robust Fault-Tolerant Vehicle Functions,“ у *CARS 2015 - Critical Automotive applications: Robustness & Safety*, Paris, France, 2015.

[110] C. Buckl, M. Geisinger, G. Dhiraj, F. J. Ruiz-Bertol и A. Knoll, „CHROMOSOME: A Run-Time Environment for Plug & Play-Capable Embedded Real-Time Systems,“ *SIGBED Rev.*, т. 11, бр. 3, p. 36–39, 2014.

[111] K. Chaaban, „A Distributed Real-Time Architecture For Advanced Vehicles,“ Докторска теза, Université de Technologie de Compiègne, France, 2006.

[112] T. Woopen, B. Lampe, T. Boddeker, A. Kampmann, B. Alrifaee, T. Stolte, I. Jatzkowski, M. Maurer, M. Mostl, S. Ackermann, C. Amersbach, D. Pullen, S. Leinen, F. Diermeyer, D. Keilhoff и M. Buchholz, „UNICARagil - Disruptive Modular Architectures for Agile, Automated Vehicle Concepts,“ 27th Aachen Colloquium Automobile and Engine Technology 2018, Aachen , Germany, 2018.

[113] A. Kampmann, B. Alrifaee, M. Kohout, A. Wüstenberg, T. Woopen, M. Nolte, L. Eckstein и S. Kowalewski, „A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles,“ у *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 2019.

[114] P. Mundhenk, E. Parodi и R. Schabenberger, „Fusion: A Safe and Secure Software Platform for Autonomous Driving,“ у *2nd International Workshop on Autonomous Systems Design (ASD 2020)*, 2020.

[115] C. Berger, B. Nguyen и O. Benderius, „Containerized Development and Microservices for Self-Driving Vehicles: Experiences & Best Practices,“ у *IEEE International Conference on Software Architecture Workshops (ICSAW)*, 2017.

[116] J.-W. Yoo, Y. Lee, D. Kim и K. Park, „An Android-based automotive middleware architecture for plug-and-play of applications,“ у *IEEE Conference on Open Systems*, 2012.

[117] E. Cochlovius, D. Dodge и S. Acharya, „The Multimedia Engine MME - a Flexible Middleware for Automotive Infotainment Systems,“ у *Digest of Technical Papers - International Conference on Consumer Electronics*, 2008.

[118] C. Menard, A. Goens, M. Lohstroh и J. Castrillon, „Achieving Determinism in Adaptive AUTOSAR,“ у *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020.

[119] H. Askaripoor, . M. H. Farzaneh и A. Knoll, „A Platform to Configure and Monitor Safety-Critical Applications for Automotive Central Computers,“ у *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA )*, 2021.

[120] M. Iorio, A. Buttiglieri, M. Reineri, F. Risso, R. Sisto и F. Valenza, „Protecting In-Vehicle Services: Security-Enabled SOME/IP Middleware,“ *IEEE Vehicular Technology Magazine*, т. 15, бр. 3, pp. 77-85, 2020.

[121] L. A. Marina, B. Trasnea и S. M. Grigorescu, „A Multi-Platform Framework for Artificial Intelligence Engines in Automotive Systems,“ у *22nd International Conference on System Theory, Control and Computing (ICSTCC)*, 2018.

[122] F. Giaimo и C. Berger, „Design Criteria to Architect Continuous Experimentation for Self-Driving Vehicles,“ у *IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, Sweden, 2017.

[123] J. Park, S. Kim, W. Yoo и S. Hong, „Designing Real-Time and Fault-Tolerant Middleware for Automotive Software,“ у *SICE-ICASE International Joint Conference*, Busan, Korea (South), 2006.

[124] S. Saidi, S. Steinhorst, A. Hamann, D. Ziegenbein и M. Wolf, „Special Session: Future Automotive Systems Design: Research Challenges and Opportunities,“ у *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Turin, Italy, 2018.

[125] H. Vdovic, J. Babic и V. Podobnik, „Automotive Software in Connected and Autonomous Electric Vehicles: A Review,“ *IEEE Access,* т. 7, pp. 166365-166379, 2019.

[126] H. Askaripoor, M. Hashemi Farzaneh и A. Knoll, „E/E Architecture Synthesis: Challenges and Technologies,“ *Electronics,* т. 11, бр. 4, 2022.

[127] H. S. Andrade, J. Schroeder и I. Crnkovic, „Software Deployment on Heterogeneous Platforms: A Systematic Mapping Study,“ *IEEE Transactions on Software Engineering,* т. 47, бр. 8, pp. 1683-1707, 2021.

[128] S. Kugele, P. Obergfell, M. Broy, O. Creighton, M. Traub и W. Hopfensitz, „On Service-Orientation for Automotive Software,“ у *IEEE International Conference on Software Architecture (ICSA)*, Gothenburg, Sweden, 2017.

[129] BMW AG, „The vsomeip stack,“ [На мрежи]. Доступно: https://github.com/COVESA/vsomeip. [Последњи приступ Септембар 2022].

[130] M. Vogel, P. Knapik, M. Cohrs, B. Szyperrek, W. Pueschel, H. Etzel, D. Fiebig, A. Rausch и M. Kuhrmann, „Metrics in automotive software development: A systematic literature review,“ *Journal of Software: Evolution and Process*, 2020.

[131] F. N. Colakoglu, A. Yazici и A. Mishra, „Software Product Quality Metrics: A Systematic Mapping Study,“ *IEEE Access,* т. 9, pp. 44647-44670, 2021.

[132] T. J. McCabe, „A Complexity Measure,“ *IEEE Transactions on Software Engineering*, pp. 308-320, 1976.

[133] S. R. Chidamber и C. F. Kemerer, „A Metric Suite for Object Oriented Design,“ *IEEE Transations on Software Engineering,* т. 20, бр. 6, pp. 476-493, 1994.

[134] F. Brito e Abreu и R. Carapuca, „Candidate Metrics for Object-Oriented Software within a Taxonomy Framework,“ у *Proceedings of AQUIS'93 (Achieving Quality In Software)*, Italy, 1993.

[135] F. Abreu, „The MOOD Metrics Set,“ у *Proceedings of ECOOP'95, Workshop on Metrics*, 1995.

[136] V. L. Basili, L. Briand и W. L. Melo, „A Validation of Object-Oriented Metrics as Quality Indicators,“ *IEEE Transactions Software Engineering,* т. 22, бр. 10, 1996.

[137] L. C. Briand, J. Wust, J. W. Daly и D. V. Porter, „Exploring the Relationship Between design Measures and Software Quality in Object-Oriented Systems,“ *Journal Systems and Software,* т. 51, бр. 3, 2000.

[138] R. Shatnawi, „An Investigation of CK Metrics Thresholds,“ у *ISSRE Supplementary Conference Proceedings*, 2006.

[139] W. Li, „Another Metric Suite For Object-Oriented Programming,“ *Journal of Systems and Software*, т. 44, бр. 2, 1998.

[140] B. Henderson-Sellers, L. Constantine и I. Graham, „Coupling and Cohesion (towards a Valid Metrics Suite for Object-Oriented Analysis and Design),“ *Object Oriented Systems,* т. 3, pp. 143-158, 1996.

[141] M. Hitz и B. Montazeri, „Measuring Coupling and Cohesion In Object-Oriented Systems,“ у *Proc. Int. Symposium on Applied Corporate Computing*, Monterrey, Mexico, 1995.

[142] L. H. Rosenberg, „Applying and Interpreting Object Oriented Metrics,“ у *Software Technology Conference*, Utah, 1998.

[143] C. E. Johnson, R. Patel, D. Radcliffe, P. Lee и J. Nguyen, „Establishing Qualitative Software Metrics in Department of the Navy Programs,“ SPAWAR System Center Pacific, San Diego, California, 2015.

[144] Google, „GoogleTest Framework,“ [На мрежи]. Доступно: https://github.com/google/googletest. [Последњи приступ Септембар 2022].

[145] B. Kovacevic, M. Kovacevic, T. Maruna и I. Papp, „A java application programming interface for in-vehicle infotainment devices,“ *IEEE Trans. Consum. Electron.,* т. 63, pp. 68-76, Feb. 2017.

[146] A. Simic, O. Kocic, M. Z. Bjelica и M. Milosevic, „Driver monitoring algorithm for advanced driver assistance systems,“ у *Proc. 24th Telecommun. Forum,* 2016.

[147] V. Gojak, J. Janjatovic, N. Vukota, M. Milosevic и M. Z. Bjelica, „Informational bird’s eye view system for parking assistance,“ у *Proc. IEEE 7th Int. Conf. Consum. Electron,* Berlin, 2017.

[148] RT-RK Institute for Computer Based Systems, „Automotive Machine Vision Alpha reference board on Texas Instruments SoCs,“ [На мрежи]. Доступно:

https://www.rt-rk.com/services/automotive. [Последњи приступ Септембар 2022].

[149] D. K. Mandal, J. Sankaran, A. Gupta, K. Castille, S. Gondkar, S. Kamath, P. Sundar и A. Phipps, „An Embedded Vision Engine (EVE) for automotive vision processing,“ у *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014.

[150] O. Djekic, „Софтверска магистрала за међупроцесорску везу у системима за аутономну вожњу,“ Нови Сад, 2018.

[151] F. Lu, S. Lee, R. K. Satzoda и M. Trivedi, „Embedded Computing Framework for Vision-based Real-time Surround Threat Analysis and Driver Assistance,“ у *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Las Vegas, NV, USA, 2016.

[152] R. K. Satzoda, S. Lee, F. Lu и M. Trivedi, „Snap-DAS: A vision-based driver assistance system on a SnapdragonTM embedded platform,“ у *Proc. IEEE Intell.Veh. Symp.*, 2015.

[153] C. Caraffi, T. Vojir, J. Trefny, J. Sochman и Matas Jiri, „A system for real-time detection and tracking of vehicles from a single car-mounted camera,“ у *15th International IEEE Conference on Intelligent Transportation Systems*, Anchorage, AK, USA, 2012.

[154] R. Steffen, R. Bogenberger, J. Hillebrand, W. Hintermaier, A. Winckler и M. Rahmani, „Design and Realization of an IP-based In-car Network Architecture,“ у *1st International ICST Symposium on Vehicular Computing Systems*, 2010.

[155] M Squared Technologies LLC, „Resource Standard Metrics (RSM) - source code metrics and quality analysis tool,“ M Squared Technologies LLC, 2022. [На мрежи]. Доступно: https://msquaredtechnologies.com/Resource-Standard-Metrics.html. [Последњи приступ Септембар 2022].

[156] FrontEndART Ltd, „SourceMeter - static source code analysis,“ 2022. [На мрежи]. Доступно: https://www.sourcemeter.com/. [Последњи приступ Септембар 2022].

[157] S. Benlarbi, K. El-Emam, N. Goel и S. N. Rai, „Thresholds for Object-Oriented Metrics,“ у *ISSRE ’00: Proceedings of the 11th International Symposium on Software Reliability Engineering*, Washington, DC, USA, 2000.

[158] M. R. Qureshi и W. Qureshi, „Evaluation of the Design Metric to Reduce the Number of Defects in Software Development,“ *International Journal of Information Technology and Computer Science* , т. 4, 2012.

[159] M. M. Trivedi, T. Gandhi и J. McCall, „Looking-in and looking-out of a vehicle: Computer-vision-based enhanced vehicle safety,“ *IEEE Trans. Intell. Transp. Syst.*, т. 8, бр. 1, pp. 108-120, 2007.

[160] F. Lu, S. Lee, R. K. Satzoda и M. Trivedi, „Embedded computing framework for vision-based real-time surround threat analysis and driver assistance,“ у *Proc. IEEE Conf. Comput. Vis. Pattern Recogn. Workshop*, 2016.

[161] B. Morris и M. Trivedi, „Vehicle iconic surround observer: Visualization platform for intelligent driver support applications,“ у *Proc. IEEE Intell. Veh. Symp.*, 2010.

[162] K. Omerovic, J. Janjatovic, M. Milosevic и T. Maruna, „Supporting sensor fusion in next generation android in-vehicle infotainment units,“ у *Proc. IEEE 6th Int. Conf. Consum. Electron.*, Berlin, 2016.

[163] M. Hammond, G. Qu и O. A. Rawashdeh, „Deploying and scheduling vision based advanced driver assistance systems (ADAS) on heterogeneous multicore embedded platform,“ у *Proc. 9th Int. Conf. Frontier Comput. Sci. Technol.*, 2015.

[164] A. Jindal и W. Ruan, „Symphony: Task scheduling and memory management in heterogeneous computing,“ 5th Int. Workshop OpenCL, 2017.

[165] Qualcomm Technologies, Inc., „Snapdragon Ride SDK: a premium platform for developing customizable ADAS applications,“ [На мрежи]. Доступно: https://www.qualcomm.com/news/onq/2022/01/05/snapdragon-ride-sdk-premium-solution-developing-customizable-adas-and-autonomous. [Последњи приступ Септембар 2022].

[166] Matt Spencer, ARM, „How the SOAFEE Architecture Brings A Cloud-Native Approach To Mixed Critical Automotive Systems,“ ARM, 2021.

[167] Google Developers, „Android Automotive OS,“ [На мрежи]. Доступно: https://developers.google.com/cars/design/automotive-os . [Последњи приступ Септембар 2022].

[168] L. Völker, „SOME/IP – Die Middleware für Ethernet-basierte Kommunikation,“ Hanser automotive networks, 2013.

[169] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii и T. Azumi, „Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems,“ у *ACM/IEEE 9th International Conference on Cyber-Physical Systems*, 2018.

[170] Elektrobit, „EB robinos - Automated driving software,“ [На мрежи]. Доступно: https://www.elektrobit.com/products/automated-driving/eb-robinos/. [Последњи приступ Септембар 2022].

[171] B. Ranft и C. Stiller, „The role of machine vision for intelligent vehicles,“ *IEEE Trans. Intell. Veh.*, т. 1, бр. 1, pp. 8-19, 2016.

[172] L. Gaikwad и S. Lokhande, „Lane departure identification for advanced driver assistance,“ *IEEE Trans. Intell. Transp. Syst.*, т. 16, бр. 2, pp. 910-918, 2015.

[173] R. O. Mbouna, S. G. Kong и M.-G. Chun, „“Visual analysis of eye state and head pose for driver alertness monitoring,“ *IEEE Trans. Intell. Transp. Syst.*, т. 14, бр. 3, pp. 1462-1469, 2013.

[174] C. Stiller, F. Puente Leon и M. Kruse, „Information fusion for automotive applications—An overview,“ *J. Inf. Fusion*, т. 12, бр. 4, pp. 244-252, 2011.

[175] Siemens , „Mentor Automotive Announces Connected OS™ Platform for Innovative Connectivity into the Car Network and to Consumer Electronic Devices,“ [На мрежи]. Доступно: https://www.plm.automation.siemens.com/global/en/our-story/newsroom/mentor-connected-os/91937. [Последњи приступ Септембар 2022].

[176] M. Segal и K. Akeley, „The OpenGL Graphics System: A Specification, Version 1.4,“ Khronos Group, Beaverton, OR, USA, 2002.

[177] B. Stockwell, „OpenGL SC: Safety-Critical Profile Specification, Version 1.0.1," Khronos Group, Beaverton, OR, USA, 2009..

[178] Khronos Group, „The OpenCL specification, version 1.1," Khronos Group, Beaverton, OR, USA, 2011..

[179] J. H. Hong, Y. H. Ahn, B. J. Kim и K. S. Chung, „Design of OpenCL framework for embedded multi-core processors," *IEEE Trans. Consum. Electron.*, т. 60, бр. 2, p. 33–241, 2014.

[180] G. Bradski и A. Kaehler, „Learning OpenCV: Computer Vision With the OpenCV Library," O'Reilly Media, Sebastopol, CA, USA, 2008.

[181] S. Gautham и E. Rainey, „The OpenVX Specification, Version 1.0," Khronos Group, Beaverton, OR, USA, 2014.

[182] 2021 OPEN Alliance SIG, „Automotive Ethernet Specifications," [На мрежи]. Доступно: http://www.opensig.org/Automotive-Ethernet-Specifications/. [Последњи приступ Октобар 2021].

[183] CE4A, „Consumer Electronic for Automotive," [На мрежи]. Доступно: https://ce4a.de/. [Последњи приступ Септембар 2022].

[184] Car Connectivity Consortium, „An organization driving global technologies for smartphone-centric car connectivity solutions," [На мрежи]. Доступно: https://carconnectivity.org/. [Последњи приступ Септембар 2022].

[185] ETAS, „ETAS AUTOSAR Solutions," [На мрежи]. Доступно: https://www.etas.com/en/applications/applications_autosar.php. [Последњи приступ Септембар 2022].

[186] Neusoft Corporation, „NeuSAR," [На мрежи]. Доступно: https://www.neusoft.com/Products/Automotive/2289/4415091330.html. [Последњи приступ Септембар 2022].

[187] Beijing Jingwei Hirain Technologies, „HiQuanten-Embedded Basic Software," [На мрежи]. Доступно: https://www.439by.com/sts/71/610/. [Последњи приступ Септембар 2022].

[188] KPIT, „KSAR Classic Platform for Safety-Critical ECUs,“ [На мрежи]. Доступно: https://www.kpit.com/workimpact/ksar-classic-platform-for-safety-critical-ecus/. [Последњи приступ Септембар 2022].

[189] NeuSoft Reach, „aCore - software based on the AUTOSAR Adaptive platform,“ [На мрежи]. Доступно: https://www.reachauto.com/en/news/show/14/. [Последњи приступ Октобар 2021].

[190] Vector Informatik GmbH, „The Solution for High-Performance ECUs according to AUTOSAR Adaptive,“ [На мрежи]. Доступно: https://www.vector.com/kr/en/products/products-a-z/embedded-components/microsar-adaptive/. [Последњи приступ Септембар 2022].

[191] Wind River Systems, „Wind River Automotive Solutions,“ [На мрежи]. Доступно: https://resources.windriver.com/automotive/automotive-solutions. [Последњи приступ Октобар 2021].

[192] ETAS, „RTA-VRTE – Adaptive AUTOSAR for Vehicle Computers,“ [На мрежи]. Доступно: https://www.etas.com/en/company/news-archive-2018-rta-vrte-adaptive-autosar-for-vehicle-computers.php. [Последњи приступ Септембар 2022].

[193] eSOL, „AUBIST Adaptive Platform,“ [На мрежи]. Доступно: https://www.esol.com/embedded/aubist_ap.html. [Последњи приступ Октобар 2021].

[194] TTTech Auto AG, „The safety mastermind for automated driving and beyond,“ [На мрежи]. Доступно: https://www.tttech-auto.com/products/safety-software-platform/motionwise/. [Последњи приступ Септембар 2022].

[195] OFP Projekt, „Interface Specification Open Fusion Platform,“ 2019.

[196] Elektrobit, „Connected vehicle software solutions,“ [На мрежи]. Доступно: https://www.elektrobit.com/products/eb-connected-vehicle. [Последњи приступ Септембар 2022].

[197] P. Burgio, M. Bertogna, I. S. Olmedo, P. Gai, A. Marongiu и M. Sojka, „A software stack for next-generation automotive systems on many-core heterogeneous platforms,“ у *2016 Euromicro Conference on Digital System Design*, Limassol, Cyprus, 2016.

[198] H. M. Hajj, W. El-Hajj, M. El-Dana, M. Dakroub и F. Fawaz, „An extensible software framework for building vehicle to vehicle applications,“ у *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference*, Caen, France, 2010.

[199] T. Lin, C. Lin и J. Lee, „Scheduling methods for OpenVX programs on heterogeneous multi-core systems,“ у *Parallel and Distributed Processing Techniques and Applications*, Las Vegas, 2105.

[200] E. Rainey, J. Villarreal, G. Dedeoglu, K. Pulli, T. Lepley и F. Brill, „Addressing System-Level Optimization with OpenVX Graphs,“ у *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Columbus, OH, 2014.

[201] „OpenMP Application Programming Interface,“ OpenMP Architecture Review Board, 2020.

[202] „The OpenACC Application Programming Interface,“ OpenACC-Standard.org, 2019.

[203] Tesla, „Artificial Intelligence & Autopilot,“ [На мрежи]. Доступно: https://www.tesla.com/AI. [Последњи приступ Септембар 2022].

[204] H. Diess, F. Witter и A. Antlitz, „Speech at the Annual Media Conference,“ Volkswagen, Wolfsburg, 2021.

[205] Daimler AG, „MB.OS is the "Next Big Thing",“ [На мрежи]. Доступно: https://www.daimler.com/career/about-us/mercedes-benz-operating-system/michael-hafner.html. [Последњи приступ Септембар 2022].

[206] BMW group, „The all-new BMW iDrive,“ 2021. [На мрежи]. Доступно: https://www.press.bmwgroup.com/global/article/detail/T0327315EN/the-all-new-bmw-idrive?language=en. [Последњи приступ Септембар 2022].

[207] Apex.AI, Inc., „Customer Success Story Toyota’s Woven Planet (Toyota creates a visionary vehicle OS, Arene),“ [На мрежи]. Доступно: https://www.apex.ai/toyota-woven-planet. [Последњи приступ Септембар 2022].

[208] SAIC Motor Corporation Limited, „SAIC Motor revealed latest Z-One Galaxy Full-stack Solution,“ [На мрежи]. Доступно:

https://www.saicmotor.com/english/latest_news/saic_motor/55392.shtml. [Последњи приступ Септембар 2022].

[209] Volvo Car Group, „Future Volvo cars to run on Volvo operating system as company takes software development in-house,“ 2021. [На мрежи]. Доступно: https://www.media.volvocars.com/global/en-gb/media/pressreleases/283545/future-volvo-cars-to-run-on-volvo-operating-system-as-company-takes-software-development-in-house. [Последњи приступ Септембар 2022].

[210] P. Oman, J. Hagemeister и D. Ash, „A Definition and Taxonomy for Software Maintainability,“ Software Engineering Test Laboratory, University of Idaho, Moscow, ID, 1991.

[211] Microsoft, „Code metrics - Maintainability index range and meaning,“ [На мрежи]. Доступно: https://docs.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2022. [Последњи приступ Септембар 2022].

[212] D. Coleman, D. Ash, B. Lowther и P. Oman, „Using metrics to evaluate software system maintainability,“ *Computer*, т. 27, бр. 8, pp. 44-49, 1994.

*Овај Образац чини саставни део докторске дисертације, односно докторског уметничког пројекта који се брани на Универзитету у Новом Саду. Попуњен Образац укоричити иза текста докторске дисертације, односно докторског уметничког пројекта.*

# План третмана података

| **Назив пројекта/истраживања** |
|---|
| Предлог архитектуре средњег слоја софтвера за рачунарски систем у возилима |
| **Назив институције/институција у оквиру којих се спроводи истраживање** |
| а) Универзитет у Новом Саду, Факултет техничких наука, Нови Сад<br>б) Истраживачко-развојни институт РТ-РК, Нови Сад |
| **Назив програма у оквиру ког се реализује истраживање** |
| Истраживање се врши у оквиру израде докторске дисертације на студијском програму Рачунарство и аутоматика |
| **1. Опис података** |
| *1.1* Врста студије<br><br>*Укратко описати тип студије у оквиру које се подаци прикупљају*<br><br>У овој студији нису прикупљани подаци. Истраживање је базирано на доступним подацима чији су извори наведени у дисертацији.<br><br>1.2 Врсте података<br><br>а) квантитативни<br><br>б) квалитативни<br><br>1.3. Начин прикупљања података<br><br>а) анкете, упитници, тестови<br><br>б) клиничке процене, медицински записи, електронски здравствени записи<br><br>в) генотипови: навести врсту _____<br><br>г) административни подаци: навести врсту _____<br><br>д) узорци ткива: навести врсту_____<br><br>ђ) снимци, фотографије: навести врсту_____ |

е) текст, навести врсту _____

ж) мапа, навести врсту _____

з) остало: описати _____

1.3 Формат података, употребљене скале, количина података

1.3.1 Употребљени софтвер и формат датотеке:

а) Excel фајл, датотека _____

b) SPSS фајл, датотека _____

c) PDF фајл, датотека _____

d) Текст фајл, датотека _____

e) JPG фајл, датотека _____

f) Остало, датотека _____

1.3.2. Број записа (код квантитативних података)

а) број варијабли _____

б) број мерења (испитаника, процена, снимака и сл.) _____

1.3.3. Поновљена мерења

а) да

б) не

Уколико је одговор да, одговорити на следећа питања:

а) временски размак између поновљених мера је _____

б) варијабле које се више пута мере односе се на _____

в) нове верзије фајлова који садрже поновљена мерења су именоване као _____

Напомене: _____

*Да ли формати и софтвер омогућавају дељење и дугорочну валидност података?*

*а) Да*

*б) Не*

*Ако је одговор не, образложити* _____

_____

## 2. Прикупљање података

2.1 Методологија за прикупљање/генерисање података

2.1.1. У оквиру ког истраживачког нацрта су подаци прикупљени?

а) експеримент, навести тип _____

б) корелационо истраживање, навести тип _____

ц) анализа текста, навести тип _____

д) остало, навести шта _____

*2.1.2 Навести врсте мерних инструмената или стандарде података специфичних за одређену научну дисциплину (ако постоје).*

_____

_____

2.2 Квалитет података и стандарди

2.2.1. Третман недостајућих података

а) Да ли матрица садржи недостајуће податке? Да Не

Ако је одговор да, одговорити на следећа питања:

а) Колики је број недостајућих података? _____

б) Да ли се кориснику матрице препоручује замена недостајућих података? Да Не

в) Ако је одговор да, навести сугестије за третман замене недостајућих података

_____

2.2.2. На који начин је контролисан квалитет података? Описати

_____

_____

2.2.3. На који начин је извршена контрола уноса података у матрицу?

_____

_____

## 3. Третман података и пратећа документација

3.1. Третман и чување података

*3.1.1. Подаци ће бити депоновани у _____ репозиторијум.*

*3.1.2. URL адреса _____*

*3.1.3. DOI _____*

*3.1.4. Да ли ће подаци бити у отвореном приступу?*

*а) Да*

*б) Да, али после ембарга који ће трајати до _____*

*в) Не*

*Ако је одговор не, навести разлог _____*

*3.1.5. Подаци неће бити депоновани у репозиторијум, али ће бити чувани.*

*Образложење*

_____

_____

3.2 Метаподаци и документација података

3.2.1. Који стандард за метаподатке ће бити примењен? _____

3.2.1. Навести метаподатке на основу којих су подаци депоновани у репозиторијум.

_____

_____

*Ако је потребно, навести методе које се користе за преузимање података, аналитичке и процедуралне информације, њихово кодирање, детаљне описе варијабли, записа итд.*

_____

_____

_____

_____

3.3 Стратегија и стандарди за чување података

3.3.1. До ког периода ће подаци бити чувани у репозиторијуму? _____

3.3.2. Да ли ће подаци бити депоновани под шифром? Да Не

3.3.3. Да ли ће шифра бити доступна одређеном кругу истраживача? Да Не

3.3.4. Да ли се подаци морају уклонити из отвореног приступа после извесног времена?

Да Не

Образложити

_____

_____

## 4. Безбедност података и заштита поверљивих информација

Овај одељак МОРА бити попуњен ако ваши подаци укључују личне податке који се односе на учеснике у истраживању. За друга истраживања треба такође размотрити заштиту и сигурност података.

4.1 Формални стандарди за сигурност информација/података

Истраживачи који спроводе испитивања с људима морају да се придржавају Закона о заштити података о личности (*https://www.paragraf.rs/propisi/zakon_o_zastiti_podataka_o_licnosti.html*) и одговарајућег институционалног кодекса о академском интегритету.

4.1.2. Да ли је истраживање одобрено од стране етичке комисије? Да Не

Ако је одговор Да, навести датум и назив етичке комисије која је одобрила истраживање

_____

4.1.2. Да ли подаци укључују личне податке учесника у истраживању? Да Не

Ако је одговор да, наведите на који начин сте осигурали поверљивост и сигурност информација везаних за испитанике:

а) Подаци нису у отвореном приступу

б) Подаци су анонимизирани

ц) Остало, навести шта

_____

_____

## 5. Доступност података

*5.1. Подаци ће бити*

*а) јавно доступни*

*б) доступни само уском кругу истраживача у одређеној научној области*

*ц) затворени*

*Ако су подаци доступни само уском кругу истраживача, навести под којим условима могу да их користе:*

_____

_____

*Ако су подаци доступни само уском кругу истраживача, навести на који начин могу приступити подацима:*

_____

_____

*5.4. Навести лиценцу под којом ће прикупљени подаци бити архивирани.*

_____

## 6. Улоге и одговорност

*6.1. Навести име и презиме и мејл адресу власника (аутора) података*

_____

*6.2. Навести име и презиме и мејл адресу особе која одржава матрицу с подацима*

_____

*6.3. Навести име и презиме и мејл адресу особе која омогућује приступ подацима другим истраживачима*

_____