УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У ЧАЧКУ

Предраг Р. Столић

РАЗВОЈ САМОАДАПТИРАЈУЋИХ ВИРТУЕЛНИХ МАШИНА У ЦИЉУ ПОБОЉШАЊА ИСХОДА УЧЕЊА У ХЕТЕРОГЕНОМ РАЧУНАРСКОМ ОКРУЖЕЊУ

докторска дисертација

Чачак, 2023.

UNIVERSITY OF KRAGUJEVAC
FACULTY OF TECHNICAL SCIENCES ČAČAK

Predrag R. Stolić

DEVELOPMENT OF SELF-ADAPTATING VIRTUAL MACHINES TO IMPROVE LEARNING OUTCOMES IN HETEROGENEOUS COMPUTING ENVIRONMENT

Doctoral Dissertation

Čačak, 2023.

ИДЕНТИФИКАЦИОНА СТРАНИЦА ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Аутор
Име и презиме: Предраг Р. Столић
Датум и место рођења: 1. јул 1980., Бор, Република Србија
Садашње запослење: Универзитет у Београду - Технички факултет у Бору
Докторска дисертација
Наслов: Развој самоадаптирајућих виртуелних машина у циљу побољшања исхода учења у хетерогеном окружењу
Број страница: 185
Број слика: 81
Број табела: 32
Број библиографских података: 206
Установа и место где је рад израђен: Универзитет у Крагујевцу - Факултет техничких наука у Чачку, Чачак
Научна област (УДК): Техничко-технолошке науке, Електротехничко и рачунарско инжењерство, Информационе технологије и системи
Ментор: др Данијела Милошевић, редовни професор, Факултет техничких наука у Чачку, Универзитет у Крагујевцу
Оцена и обрана
Датум пријаве теме: 10. март 2023. године
Број одлуке и датум прихватања теме докторске/уметничке дисертације: IV-04-503/7 од 13. јула 2023. године
Комисија за оцену научне заснованости теме и испуњености услова кандидата:
1. др Марија Благојевић, ванредни професор, Факултет техничких наука у Чачку, Универзитет у Крагујевцу 2. др Данијела Милошевић, редовни професор, Факултет техничких наука у Чачку, Универзитет у Крагујевцу 3. др Јелица Протић, редовни професор, Електротехнички факултет, Универзитет у Београду
Комисија за оцену и одбрану докторске/уметничке дисертације: 1. др Марија Благојевић, ванредни професор, Факултет техничких наука у Чачку, Универзитет у Крагујевцу 2. др Јелица Протић, редовни професор, Електротехнички факултет, Универзитет у Београду 3. др Марјан Милошевић, ванредни професор, Факултет техничких наука у Чачку, Универзитет у Крагујевцу
Датум одбране дисертације:

ЗАХВАЛНИЦА

Ову докторску дисертацију посвећујем свом покојном оцу, др Радисаву Столићу, ванредном универзитетском професору који ми је пружао несебичну подршку у животу и који ми је усадио љубав према научно-истраживачком раду и показао да кад се нешто воли и жели границе су само оне које лично поставиш.

Хвала мојој породици, супрузи Снежани и деци Душану и Вањи на несебичној подршци коју су ми пружали током целокупног трајања докторских студија и израде докторске дисертације. Ова докторска дисертација је и њихов понос, победа и заслуга. Своју захвалност дугујем и сестри Александри и мајци Јованки.

Посебну захвалност дугујем проф. др Данијели Милошевић, ментору ове докторске дисертације и проф. др Зорану Стевићу на свом доприносу и уложеном времену приликом израде докторске дисертације.

Захвалност дугујем и свим професорима код којих сам имао част и задовољство да слушам предмете у оквиру својих докторских студија, као и члановима комисија који су узимали учешће у процесу пријаве и одбране докторске дисертације. Захваљујем се и члановима колектива Факултета техничких наука у Чачку на свим лепим речима, уложеном труду и срдачности током целокупног трајања докторских студија.

Предраг Р. Столић

РЕЗИМЕ

Савремено извођење наставних процеса у оквиру високошколског образовања ослања се на употребу принципа учења од куће који, поред предности, испољавају и неке недостатке. Један од уочених недостатака јесте хетерогеност рачунарског окружења, као резултат употребе личних рачунарских ресурса студената будући да ти рачунарски ресурси међусобно испољавају велику разноликост.

Начин превазилажења настале хетерогености јесте превођење у хомогени облик употребом виртуелних машина. Оваквим приступом врши се приближавање постизању униформности наставног процеса посматрано са становишта употребе рачунарске технике. Виртуелизација уноси нову комплексност јер се све активности виртуелизације спроводе од стране студената и наставног особља. То подразумева да ће се актери наставног процеса привремено удаљити од примарних циљева везаних за постизање позитивних исхода учења. Потребно је развити механизме који ће омогућити аутоматизацију процеса везаних за примену виртуелне машине у наставном процесу у циљу свођења учешћа актера наставног процеса у виртуелизацији на прихватљив минимум.

Ово је постигнуто развојем самоадаптирајуће виртуелне машине која, у односу на класичну виртуелну машину, поседује адаптивна својства у хардверском и софтверском смислу. Адаптивност у хардверском смислу огледа се у аутоматском прилагођавању тренутним условима окружења у коме виртуелна машина остварује рад, анализом параметара окружења, статистичком обрадом истих и предикцијом будућих стања, чиме се дефинишу параметри рада виртуелне машине. У софтверском смислу, адаптивност се постиже коришћењем принципа онтолошког инжењерства чиме се сви инсталациони процеси у оквиру виртуелне машине обављају без директног учешћа корисника.

Све функционалности реализованог решења потврђене су одговарајућим тестирањима.

Кључне речи: адаптивност, виртуелизација, виртуелна машина, виртуелни хардвер, високошколско образовање, онтолошко инжењерство, предиктивна аналитика, софтвер, хардвер

ABSTRACT

The modern implementation of teaching processes within higher education relies on the use of the principles of learning from home, which have some disadvantages. One of the observed disadvantages is the heterogeneity of the computer environment, as a result of the use of personal computer resources of students, since these computer resources exhibit great diversity among themselves.

The way to overcome the resulting heterogeneity is to translate it into a homogeneous form using virtual machines. Virtualization introduces new complexity because all virtualization activities are performed by students and teaching personnel. This means that the actors of the teaching process will temporarily move away from the primary goals related to achieving positive learning outcomes. It is necessary to develop mechanisms that will enable the automation of processes related to the application of a virtual machine in the teaching process in order to reduce the participation of actors of the teaching process in virtualization to an acceptable minimum.

This was achieved by developing a self-adaptive virtual machine that, compared to a classic virtual machine, has hardware and software adaptive properties. Hardware adaptability is reflected in automatic adaptation to the current conditions of the environment in which the virtual machine works, by analyzing the parameters of the environment, statistical processing of them and prediction of future states, which defines the operating parameters of the virtual machine. Software adaptability is achieved by using the principles of ontological engineering, whereby all installation processes within the virtual machine are performed without direct user participation.

All functionalities of the implemented solution were confirmed by appropriate tests.

Key words: adaptivity, hardware, higher education, ontology engineering, predictive analytics, software, virtualization, virtual machine, virtual hardware

САДРЖАЈ

	Стр.
1. УВОДНА РАЗМАТРАЊА	**1**
1.1. Предмет и циљ истраживања	**1**
1.2. Основне хипотезе	**4**
1.3. Методе и методологије истраживања	**4**
1.4. Преглед стања у подручју истраживања	**5**
1.4.1. Хетерогеност рачунарског окружења	**5**
1.4.2. Преглед сличних приступа коришћених за реализацију софтверских основа предложеног решења	**6**
1.4.3. Преглед сличних приступа коришћених за реализацију хардверских основа предложеног решења	**8**
1.5. Преглед излагања	**12**
2. ХЕТЕРОГЕНОСТ РАЧУНАРСКОГ ОКРУЖЕЊА У ДОМЕНУ ВИСОКОШКОЛСКОГ ОБРАЗОВАЊА	**14**
2.1. Појам хетерогености	**14**
2.2. Значај правилне анализе хетерогености рачунарског окружења	**16**
2.3. Поступак анализе хетерогености рачунарског окружења	**23**
2.4. Резултати анализе хетерогености рачунарског окружења	**32**
3. СОФТВЕРСКЕ ОСНОВЕ ПРЕДЛОЖЕНОГ РЕШЕЊА	**43**
3.1. Уводна разматрања	**43**
3.2. Коришћење виртуелних машина у домену високог образовања	**48**
3.3. Развој онтологије	**53**
3.3.1. Класе	**55**
3.3.2. Везе (својства објеката)	**59**
3.3.3. Атрибути	**63**
3.3.4. Унапред дефинисани ентитети	**65**
3.3.5. Расуђивање	**66**
3.4. Развој пратећег софтвера за онтологију	**67**
4. ХАРДВЕРСКЕ ОСНОВЕ ПРЕДЛОЖЕНОГ РЕШЕЊА	**74**
4.1. Идентификација узрочно-последичних веза	**74**
4.2. Иницијализација виртуелног хардвера	**84**
4.3. Прикупљање података у реалном времену	**90**
4.3.1. Партиције	**90**
4.3.2. Меморија	**91**
4.4. Анализа постојећих и предикција будућих стања	**101**
4.5. Прорачун вредности количине виртуелне радне меморије	**110**
5. РЕАЛИЗАЦИЈА САМОАДАПТИРАЈУЋЕ ВИРТУЕЛНЕ МАШИНЕ	**113**
5.1. Припрема основне виртуелне машине	**113**
5.2. Софтвер за управљање виртуелном машином	**124**
5.3. Инсталационе процедуре	**133**

6. РЕЗУЛТАТИ ТЕСТИРАЊА **138**
6.1. Уводна разматрања **138**
6.2. Функционално тестирање **139**
6.3. Тестирање оптерећења **148**
6.4. Тестирање конфигурације **153**
6.5. Тестирање корисничког интерфејса **158**
7. ПРИМЕНЉИВОСТ У НАСТАВНОМ ПРОЦЕСУ **160**
8. УОЧЕНА ОГРАНИЧЕЊА И МОГУЋИ БУДУЋИ ПРАВЦИ РАЗВОЈА **165**
8.1. Уочена ограничења **165**
8.2. Могући будући правци развоја **167**
9. ЗАКЉУЧАК **170**
ЛИТЕРАТУРА **173**

1. УВОДНА РАЗМАТРАЊА

1.1. Предмет и циљ истраживања

Реализација наставног процеса одавно је превазишла традиционалне оквире у којима је само извођење наставног процеса смештено искључиво у оквире учионица и лабораторија, односно искључиво у простор институције у оквиру које се сам наставни процес реализује. Наставни процес није постао имун на константни напредак које испољавају технике и технологије и у складу са тим превазилази своје традиционалне оквире у којима доминира учионица. Посебно је уочљив утицај дигиталних технологија на модернизацију наставног процеса па тако савремену реализацију наставног процеса не можемо данас замислити без неке врсте повезаности са употребом рачунара. Та повезаност испољава се директно или индиректно, па у складу са тим наставни предмети се у потпуности ослањају на употребу рачунарске технике у извођењу наставе, или су макар делимично подржани употребом рачунарске технике.

Интернет и интернет технологије су реализацију наставног процеса у последње две деценије подигли на један нови ниво. Фокус наставног процеса се трансформисао и у извођењу целокупног наставног процеса постају доминантни принципи тзв. „обрнуте учионице“, односно тежи се ка томе да кандидати савладавају предвиђено градиво радом код куће, док се капацитети образовних институција више стављају у један консултативни план, односно у оквиру рада у учионицама и лабораторијама се више пажње посвећује спровођењу одговарајућих дискусија, анализа и сличних поступака које кандидати не могу самостално савладати код куће употребом личних ресурса. У ту сврху су данас реализоване разне методе, методологије, технике, реализовани су одговарајући системи за електронско учење, за рад на даљину, туторски системи, убачена је одговарајућа интелигенција у системе како би се савладали различити стилови учења. Овакав приступ дао је изузетне резултате посебно у условима отежаног извођења наставног процеса какав је, на пример, био у периоду од 2020. до 2022. године где су, услед пандемијских услова, капацитети образовних институција постали делом или у целости недоступни конзументима образовног процеса.

Иако напред наведени приступ испољава многе квалитете, он, такође, испољава и одређене недостатке који су уочени током различитих нивоа реализације учења на даљину („онлајн учења“). Један од основних проблема које треба превазићи јесте изразита хетерегеност рачунарског окружења која се испољава приликом реализације наставног процеса на горе поменути начин. Приликом извођења наставног процеса у оквиру капацитета образовне институције, рачунарско окружење се може сматрати хомогеним и предавачи и кандидати су јасно и недвосмислено упознати са капацитетима, карактеристикама и особеностима целокупне рачунарске опреме која се користи у реализацији наставног процеса па су могућности појаве грешака, погрешних интерпретација или неких непредвиђених околности сведене на минимум, а уколико до њих ипак дође може се реговати правовремено, недвосмислено и на одговарајући начин. Трансфером учења на ресурсе који нису везани за домен образовних институција, односно коришћењем личних ресурса у виду рачунарске опреме која је у власништву самих кандидата, овај проблем постаје врло сложен и изразит будући да постоји изузетна хетерогеност рачунарског окружења. Сваки кандидат поседује рачунарске ресурсе различитих капацитета, карактеристика, стања, начина коришћења и осталог и врло је тешко постићи униформност. Употреба различитих рачунарских система од стране кандидата манифестује се у томе да ће се задаци који се представљају пред кандидате реализовати у различитим окружењима, различити ће бити поступци инсталација

софтвера, различите ће бити перформансе у извршавању, различите ће бити грешке које се могу испољити и које се испољавају приликом инсталације софтвера, стартовањем истог, коришћења и сличних операција.

Један од начина превазилажења постојећих ограничења јесте увођење концепата виртуелизације у сам наставни процес чиме се може постићи боље приближавање постизању једне врсте униформности самог наставног процеса са становишта употребе рачунарске технике. Реализацијом виртуелних машина обезбеђује се одговарајућа конзистентност окружења које ће се користити за потребне спровођења разних активности које се подразумевају приликом релизације наставних активности. Такође, виртуелне машине омогућавају реализацију једне врсте изолованог окружења у коме се све активности које се спроводе ограничавају на домен виртуелне машине чиме се спречава настајање таквих грешака које могу нарушити стабилност или довести до таквих грешака које могу у потпуности онемогућити систем који користе кандидати и што може у извесној мери бити ствар и одговорности предавача који спроводи сам наставни процес, па чак и саме образовне институције. Виртуелне машине нуде могућност једне врсте превођења хетерогености рачунарског окружења у хомогену средину.

У горе поменутој трансформацији, у којој се тежи одређеном прелазу из хетерогености у хомогеност, идентификују се два кључна проблема, па се сходно томе и у проблему истраживања идентификују два потпроблема који ће бити посебно размотрени.

Први је дат у софтверском домену и подразумева да ће се све активности везане за инсталацију и конфигурисање неопходног софтвера за коришћење у оквиру виртуелне машине спровести од стране крајњег корисника. Будући да се ради о индивидуалном приступу, очекивано је да ће се јавити мања или већа разноликост у тумачењу процедура, у реализацији самих поступака, одабиру софтвера и сличном, па не можемо говорити о потпуној униформности и биће значајнијих одступања од начела хомогености. Такође, овде треба истаћи да не би било адекватно решење ни обезбеђивање виртуелне машине од стране техничког особља образовне институције јер би постојала два могућа сценарија. У првом сценарију би била реализована једна виртуелна машина за целокупну образовну институцију, односно за све нивое студија, све студијске програме и све предмете. Таква виртуелна машина би била преобимна, захтевна у погледу ресурса, на истој би се налазио целокупан софтвер од кога велика већина за крајњег корисника и није неопходна (не слуша све предмете, други студијски програм и слично) и на крају таква виртуелна машина би била врло збуњујућа за употребу од стране крајњих корисника. У другом сценарију посматра се реализација појединачних виртуелних машина за сваки од предмета. То практично значи стварање велике количине виртуелних машина (на десетине виртуелних машина) где се сада јавља додатна оптерећеност техничког особља и ресурса саме образовне институције што такође није препоручљиви концепт. Овде би се вероватно испољили и још неки додатни пропратни ефекти попут могућег предимензионисања система у појединим моментима, повећање степена комплексности инфраструктуре, додатни економски напори и слично.

Други проблем је више оријентисан на хардверску компоненту употребе виртуелне машине. Иако је речено да постоји изолованост виртуелне машине, овде се ипак мора узети у обзир чињеница да се виртуелне машине неће извршавати у потпуности самостално, односно да ипак постоји одређена повезаност између гостујућег система (виртуелна машина) и примарног система (host) који у суштини представља рачунарски систем крајњег корисника са свим особеностима и разноликостима који се могу јавити. У складу са тим постоји директна узрочно-последична веза између стварних ресурса коју поседује рачунарски систем корисника и виртуелних ресурса које ће црпети виртуелна

машина. Свакако три кључна ресурса који се издвајају по повезаности стварног система и виртуелне машине јесу централна процесорска јединица (CPU), меморија и дискови (било да се ради о HDD или SSD). Ове три компоненте су кључне за правилно функционисање виртуелних машина и виртуелне машине заузимају значајније капацитете истих, тако да правилна алокација и управљање овим ресурсима постају ствари од суштинској значаја. Међутим, већ је напоменуто постојање изразите хетерогености рачунарског окружења, што значи да ће између рачунара крајњих корисника постојати значајне варијације у погледу ових ресурса. Постојаће различити процесори по типу виртуелизације коју они омогућавају, по броју физичких и логичких језгара, по сету инструкција и слично, постојаће различите количине реално расположиве меморије, постојаће различита партиционисаност дискова и различито заузеће по партицијама. На све ово треба додати и временску променљивост доступности ових ресурса, будући да се заузеће процесора мења током рада, као и количина расположиве меморије, а постоји и мања или већа променљивост у заузећу на партицијама. Овде треба напоменути да свакако постоји утицај и графичког подсистема (интегрисане и дискретне графичке карте), мрежних контролера (жичних и бежичних), улазно-излазних (I/O) јединица, али њихов утицај је мање доминантан у односу на процесор, меморију и дискове па се због обимности анализе неће узети у разматрање. Као што се из реченог види због различитости и саме променљивости окружења у коме се виртуелна машина реализује и у коме ће виртуелна машина остваривати своје задатке и овде ће се јавити проблем налажења једног свеобухватног и универзалног решења.

У складу са претходно изложеним, предмет истраживања докторске дисертације се може дефинисати као налажење одређеног скупа метода које ће омогућити правилну реализацију и рад виртуелних машина у поменутом изразито хетерогеном рачунарском окружењу и адекватну адаптацију виртуелних машина на променљивост окружења у коме остварују своје задатке. Овде се намерно користи одредница „скуп метода“ како би се указало да се морају обухватити оба аспекта адаптације виртуелних машина, како у хардверском, тако и у софтверском смислу. На тај начин реализује се један свеобухватан приступ, уместо парцијалних приступа усмерених само на софтвер, или само на хардвер, чиме би се покрила само једна од особености које су разматране претходно и чиме би се изгубио један део функционалности решења за којим се у дисертацији трага.

Примарни циљ истраживања представља примена претходно поменутих добијених методолошких принципа на реализацију самоадаптирајуће виртуелне машине, односно крајњи циљ се може формулисати као развој самоадаптивности виртуелне машине на изразито хетерогено и променљиво рачунарско окружење кроз одговарајуће поступке и решења. Тиме би се остварило једно аутоматизовано, паметно и интелигентно решење чиме би се различити корисници (студенти, предавачи, техничко особље и остали заинтересовани) боље фокусирали на остварење исхода учења у целокупном наставном процесу уместо честе праксе да се део фокуса губи услед разних активности везано за различите софтверске и хардверске аспекте рада виртуелне машине (праћење перформанси, конфигурисање, инсталације, ажурирања, оптимизације и остало). Реализацијом самоадаптирајућих виртуелних машина стиче се један велики простор за коришћење овако реализоване виртуелизације у различитим врстама едукације јер се смањује неопходна количина техничког знања из различитих домена рачунарске технике (оперативних система, рачунарских система, виртуелизације итд.) будући да се често јавља ситуација да корисници немају нека основна предзнања из домена рачунарске технике, или та знања поседују у недовољној количини, а чест је случај и погрешних тумачења, као и недовољне информисаности приликом доношења значајнијих одлука и спровођења кључних акција.

1.2. Основне хипотезе

Општа хипотеза, којом се може окарактерисати реализовано истраживање у оквиру докторске дисертације, формулисана је на следећи начин:

X: Може се развити скуп одговарајућих метода које ће омогућити реализацију самоадаптирајућих виртуелних машина, како са хардверског, тако и са софтверског аспекта, уз помоћ којих се проблем хетерогености рачунарског окружења може адекватно савладати и тиме омогућити боље остварење предвиђених исхода учења у наставном процесу.

Посебне хипотезе дате су у следећим облицима:

X_1: Постоји изражено хетерогено рачунарско окружење у софтверском и хардверском смислу у коме се одвија наставни процес у предметима где је доминантна употреба рачунара када се тај процес спроводи приликом активности које се спроводе ван образовних институција („учење код куће“).

X_2: Проблеми које изазива хетерогеност рачунарског окружења у софтверском и хардверском смислу може се превазићи употребом метода виртуелизације и виртуелних машина.

X_3: Може се развити самоадаптирајуће виртуелно окружење у софтверском смислу за употребу у наставном процесу где је доминантна употреба рачунара.

X_4: Може се развити самоадаптирајуће виртуелно окружење у хардверском смислу за употребу у наставном процесу где је доминантна употреба рачунара.

X_5: Употребом самоадаптирајућег виртуелног окружења врши се унапређење фокуса конзумената наставног процеса на сам наставни процес, чиме се остварују додатни бенефити у погледу реализације исхода учења.

X_6: Самоадаптирујуће виртуелно окружење омогућава стварање таквог окружења за реализацију наставног процеса које је мање захтевно у инфраструктурном, економском и енергетском погледу.

1.3. Методе и методологије истраживања

1. За савладавање сложености система, односно за адекватну анализу система и захтева које систем треба да испуни, коришћене су методе Структурне системске анализе (CCA).
2. У целокупном развоју система коришћен је итеративно-инкрементални приступ.
3. За анализу хетерогености рачунарског окружења развијен је одговарајући софтвер за идентификацију основних хардверских и софтверских параметара рачунара. Софтвер је развијен за Windows, Linux и macOS платформу у Python програмском језику. За прикупљање добијених података извршено је моделовање одговарајуће релационе базе података, а потом је на основу одговарајућег инжењеринга над креираним моделом реализована физичка база података у одговарајућем систему за управљање релационим базама података (MySQL). Након извршеног прикупљања података, добијени подаци су адекватно анализирани одговарајућим статистичким методама.
4. Развој самоадаптирајућег виртуелног окружења извршено је на основу Linux guest оперативног система, док је развој у односу на host оперативни систем извршен за Windows, Linux и macOS платформу. Коришћен је VirtualBox хипервизор типа 2 који постоји у одговарајућим верзијама за Windows, Linux и macOS платформе. Целокупан

развој извршен је употребом Python програмског језика и његових одговарајућих библиотека.

5. Снимање параметара неопходног софтвера и подешавање софтвера на виртуелној машини извршен је употребом принципа онтолошког инжењерства, при чему је у развоју онтологија коришћен Stanford-ов Protégé софтвер. Релевантни подаци добијени су употребом скриптовања у Linux shell-у, као и коришћењем одговарајућег софтвера развијеног у програмском језику Python. Самоадаптација виртуелне машине са становишта софтвера реализована је употребом програмског језика Python и релеватних библиотека.
6. Сервиси за снимање хардверских параметара и одговарајућих хардверских стања host рачунара реализовани су употребом програмског језика Python, одговарајућег скриптовања у shell-у на Linux и macOS платформама, као и коришћењем WMI (Windows Management Instrumentation) код Windows платформе.
7. Сервис за снимање параметара и стања дефинисаног виртуелног хардвера на Linux виртуелној машини реализован је употребом Python програмског језика и његових одговарајућих библиотека, као и употребом скриптовања у Linux shell-у.
8. Снимање података вршено је употребом релационих база података реализованих у оквиру MySQL система за управљање релационим базама података на основу претходно развијених модела. У појединим моментима, у сврху снимања података, паралелно су употребљаване и текстуалне датотеке реализоване у CSV (Comma Separated Value) формату.
9. Анализа добијених података извршена је применом одговарајућих математичких статистичких метода везаних за анализу временских серија (time series analysis). У циљу добијања одговарајућих увида и предикција коришћени су елементи науке о подацима и одговарајући алгоритми вештачке интелигенције намењени предвиђању везаних за временске серије (time series forecasting). Одговарајући програмски кодови развијани су коришћењем програмског језика Python и његових одговарајућих библиотека, као и GNU Octave програмског језика и његових одговарајућих пакета.
10. Целокупно управљање хипервизором у циљу адаптације виртуелног хардвера извршено је реализовањем наменског софтвера у програмском језику Python за Windows, Linux и macOS платформе.
11. Наменски сервери коришћени приликом развоја, тестирања и продукције крајњег решења реализовани су употребом LNMP стека (Linux – Nginx – MySQL – PHP stack).

1.4. Преглед стања у подручју истраживања

1.4.1. Хетерогеност рачунарског окружења

Савремено рачунарство реализује различите задатке у различитим областима који, у зависности од комплексности решења за којим се трага, захтевају употребу разноврсних ресурса. У [1] аутори истичу постојање хетерогеног окружења које је карактеристика савременог рачунарства будући да на различитим нивоима постоји изражена разноликост ресурса који се користе, односно постоје значајне разлике у карактеристикама појединачних рачунарских компоненти, као и рачунара и рачунарских система у целости. Ако се посматрају већи организовани скупови рачунарских ресурса који су организовани по принципима модерних fog, edge, cloud и сличних окружења, опет се посебна пажња мора обратити на сваки локални чвор услед изузетне хетерогености која се испољава на том најнижем нивоу као последица конструкције сваког од чворова посебно, а та хетерогеност се потом преноси и испољава и на вишим нивоима [2]. Зато се, данас, поред поузданости, скалабилности, безбедности и приватности, хетерогеност

често намеће као један од изазова које треба савладати приликом пројектовања и имплементације решења заснованих на савременим рачунарским концептима [3], односно, између осталог, један од фокуса треба да буде и остваривање таквог приступа који омогућава адекватно повезивање хетерогених уређаја у широком спектру окружења [4]. Начини адаптације на изразито хетерогено окружење остварују се примењивањем неколико кључних концепата. Да би се правилно руковало хетерогеношћу ресурса мора се обезбедити правилно управљање истим узимајући у обзир одређену непредвидивост, променљивост и динамичност коју испољавају [5], потом треба обезбедити адекватну повезаност међу хетерогеним уређајима [6] и на крају и софтверска решења треба адаптирати тако да се могу успешно примењивати у условима хетерогености [7], [8].

Када се говори о претходним разматрањима по питању третирања хетерогености али посматрано у системима намењеним употреби у домену високошколског образовања, поред поменутог, отварају се и још неки додатни изазови које је потребно превазићи како би се адекватно савладали проблеми хетерогеног окружења. Ови нови изазови појавили су се као последица дигиталне трансформације високошколског образовања, а посебно су се испољили у великој мери у периоду од 2020. године надаље услед светске пандемијске ситуације узроковане појавом COVID-19 болести. Управо у оквиру тог контекста, аутори у [9] су уочили да се услед изражене хетерогености окружења, студенти суочавају са појавом одређених техничких проблема приликом коришћења личних рачунарских ресурса за учење. Ови аутори наводе, такође, да је у складу са тенденцијом појаве оваквих околности приликом учења, између осталог, пожељно увести у сам процес виртуелна окружења. Употреба личних рачунарских ресурса повећава опсег проблема који се могу испољити у једном таквом изразито хетерогеном рачунарском окружењу, увећавају се безбедоносни изазови, јавља се проблем употребе и заштите личних података и слично. Међутим, у [10] препознат је још један озбиљан проблем који се често налази на маргинама осталих изражених, а то је питање застарелости опреме и софтвера који постојеће ризиле значајно увећава, а такође уводи и неке нове па је неопходно наћи начин да се и ови ризици сведу на прихватљиви минимум. Путем виртуализације се може наћи адекватно решење на наведене изазове будући да се свака виртуелна машина без увођења комплексних операција и додатних напора по саме студенте може врло лако и брзо инсталирати и реинсталирати, подесити, поправити и софтверски прилагодити [11]. Треба напоменути и да адекватно савладавање хетерогености рачунарског окружења не мора да буде скуп процес, већ се оно може савладавати и употребом бесплатних решења каква су на пример поједина решења заснована на употреби софтвера отвореног кода (eng. open-source software) [12]. Поред финансијске карактеристике, већина ових решења је доступна и на више платформи (Windows, Linux, macOS) што их чини адекватним кандидатима за превазилажење проблема и изазова који се налазе у оквирима хетерогених рачунарских окружења.

1.4.2. Преглед сличних приступа коришћених за реализацију софтверских основа предложеног решења

У складу са претходним разматрањима, везано за проблеме које испољава хетерогеност рачунарског окружења, можемо рећи да се са софтверског становишта уже идентификују неки специфични изазови које треба савладати. У [13] аутори уочавају да се, када се у разматрање узме софтверско окружење, испољава неколико кључних питања на које треба адекватно одговорити, а то су тешко дељење и поновна употреба стечених знања, сваки софтвер поседује неке своје одређене особености што подиже ниво комплексности међусобне комуникације, јавља се проблем сувишности будући да

постоји велики део заједничког знања међу различитим софтверима и стицање знања се увек врши од почетка за сваки нови софтвер који се узима у разматрање. Употребом онтологија може се одговорити на претходно постављене задатке будући да оне дефинишу одговарајући модел састављен од одговарајућих класа, атрибута и међусобних релација и који кроз своју одговарајућу спецификацију, концептуализацију и имплементацију у одговарајућем окружењу пружа заједничку семантику уз могућност адекватне размене знања и њихове једноставне поновне употребе [14]-[17]. Принципи засновани на онтологијама омогућили су превазилажење широког спектра проблема који су се јављали у домену софтверског инжењерства и реализације информационих система развојем нових приступа, а постоје и случајеви допуњавања већ постојећих методологија онтологијама и стварања одговарајућих интегрисаних приступа [18], [19].

Онтологије су већ доказано средство у сфери системског инжењерства, како у сфери управљања информацијама, тако и у сфери управљања конфигурацијама [20], [21]. Свој пуни допринос онтологије дају у случајевима када треба одговорити на изазове променљивости софтверског окружења јер обезбеђују неопходне технике идентификације насталих промена у софтверском окружењу, као и технике неопходне за креирање адекватног одговора на настале промене у софтверском окружењу [22], [23] и омогућавају синтетисање добијених знања у оквиру организованих репозиторијума [24] који омогућавају принцип поновне употребе (reuse).

Напред наведено указује на применљивост онтолошких принципа у савладавању дефинисаних проблема у софтверском домену подручја којим се предметна докторска дисертација бави. У литератури постоји више приступа одређивању типа онтологије у зависности од предмета класификације, као и више приступа при дефинисању методологија за развој онтологија. Без улажења у дубљу анализу претходног тврђења, онтолошки приступ који ће бити коришћен у дисертацији би најподесније било сврстати у групу приступа директно зависних од апликације (application-dependent methodologies), односно саму онтологију по типу у онтологију апликације (application ontology) [25]. У домену онтолошког инжењерства, развијено је више језика који се могу користити успешно приликом развоја различитих врста онтологија. Приликом развоја онтологије за потребе предметне докторске дисертације искористиће се предности које нуди OWL 2 Web Ontology Language [26] због своје широке распрострањености и доказане употребљивости, будући да се највећи број онтолошких решења у софтверском домену реализује управо употребом наведеног језика [27], [28]. Такође, коришћењем OWL/XML формата постиже се приказ онтологије у једном разумљивијем формату [29] који је погоднији за даљу употребу у одговарајућим програмским језицима и апликацијама.

Комплетан процес развоја онтологије може се извршити употребом различитих окружења и софтверских алата. За развој онтологије овом приликом коришћен је бесплатан софтвер отвореног кода Protégé [30], [31] који, у сагласности са напред наведеним, у потпуности подржава развој онтологија употребом OWL језика [32] и који представља најзаступљеније и најпопуларније окружење за развој онтологија [33]. Такође, у оквиру поменутог окружења може се извршити адекватна визуелизација одговарајућих класа и међусобних релација коришћењем одговарајућих приказа генерисаних употребом OWLViz [34] и OntoGraph [35] додатака. Током развоја онтологија, од изузетне важности је провера конзистентности самих онтологија која се врши употребом специјализованих програмских решења за резоновање (eng. Reasoners) [36]. Постоје различити програми ове намене, а у случају развоја онтологије у оквиру предметне докторске дисертације коришћени су FaCT++ [37] и HermiT [38] у оквиру Protégé развојног окружења.

На крају претходних разматрања овде треба напоменути две битне чињенице. Као што се из претходних разматрања види, онтологије нуде изузетну базу у софтверском

домену будући да представљају зрели алат за превазилажење многих проблема за које не постоји адекватно решење коришћењем неких других доступних техника. Међутим, онтологије нису покриле многе аспекте модерне употребе рачунара и рачунарских окружења. Као што аутори у [39] наводе, многи софтверски аспекти у домену традиционалног рачунарства су покривени принципима онтолошког инжењерства и постоји обиље публиковане литературе, међутим многи аспекти, између којих су и аспекти употребе у виртуелизацији и виртуелним окружењима, на шта се односе и разматрања изнета у предметној докторској дисертацији нису адекватно покривена и разматрана и ту постоји огроман простор за даље деловање. Ако се сада узме у обзир ужи домен који се односи на системе намењене образовању, како се у [40] тврди, постоји проблем успостављања одговарајућих рачунарских окружења и информационих система услед убрзаног технолошког развоја и изузетног испољавања хетерогености, али управо на овом месту принципи онтолошког моделирања дају изузетно добру основу за превазилажење ових проблема.

1.4.3. Преглед сличних приступа коришћених за реализацију хардверских основа предложеног решења

Виртуелизација у домену образовања, а посебно у домену високошколског образовања, дуго времена је у великом броју случајева погрешно интерпретирана и поистовећивана са неким другим концептима попут концепта учења на даљину (eng. distance learning) иако се ради о једном засебном, комплексном концепту на који се треба обратити посебна пажња [41]. Поред тога, чак и у економски развијеним земљама са напреднијим образовним системима, концепт виретуелизације није свуда разматран подједнако, будући да је постојала велика бојазан да ће увођење концепата виртуелизације у наставни процес иницирати велике финансијске напоре за образовне инситуције, па је за инситуције са скромнијим буџетима виртуелизација остајала на одређеним маргинама модерног образовања [42]. Међутим, свест о виртуелизацији у високошколском образовању се значајно мења у скорије време, будући да рачунарска техника и технологија напредују огромном брзином и високошколске институције све теже могу држати корак у том погледу, односно опрема инсталирана у оквиру капацитета ових образовних институција све брже застарева и све теже може испратити брзе промене у савременом окружењу, па се виртуелизација у складу са напред наведеним намеће као неопходно деловање у правцу одржавања континуитета модерне наставе [43].

Други кључни моменат који је увођење концепта виртуелизације у високошколско образовање начинио приоритетом, јесте потреба да се омогући стварање таквог персонализованог окружења у коме ће се остварити измештање фокуса са наставника на студента [44] у коме ће студент моћи адекватна практична искуства и бити спреман за реалну примену стечених знања у стварном професионалном окружењу попут практичних знања које на пример професионални пилоти стичу спровођењем вежби у виртуелним окружењима у тренажним симулаторима [45]. Ако разматрамо извођење наставе у традиционалном окружењу у оквиру рачунарских лабораторија, претходно је врло тешко оствариво услед физичке ограничености ресурса којим образовне институције располажу будући да се приликом извођења наставе на једно рачунарско место углавном смешта више студената (уобичајена пракса је да то буде 2-3 студента), а поред тога постоји и проблем да се предавач не може адекватно фокусирати на сваког од студената, нарочито у већим наставним групама [46]. На пример чак и ако би предавач пришао сваком наставном месту, реално би само један студент активно користио рачунарско окружење, док би остали студенти морали бити пасивни посматрачи услед физичке ограничености наставног места [46]. Уколико би разматрали учење студената

код куће чиме би се могао остварити принцип један студент на једно наставно место, без виртуелизације постоји проблем да би студент морао у једном моменту да обавља и додатне улоге, на пример да буде систем администратор како би отклонио проблеме са инсталацијом софтвера, проблеме у конфигурацији и слично, чиме би фокус студента прешао са наставног процеса на неке друге активности што, такође, представља проблем [47].

Иако се виртуелизацијом постижу многе предности у домену високошколског образовања, овде се мора споменути да постоје и уочени одређени недостаци који се испољавају њиховом имплементацијом попут потребе за додатним лиценцирањем софтвера од стране високошколских институција, као и додатног оптерећења како наставника, тако и студената у погледу савладавања правилног коришћења и конфигурисања софтвера за виртуелизацију [48]. Међутим, треба нагласити да применом одговарајућих савремених решења и технологија и ови негативни аспекти виртуелизације могу се у неким случајевима избећи у потпуности, или се у већини случајева они своде на прихватљиви минимум. Тако данас постоје доказани поступци виртуелизације у домену високошколског образовања чију окосницу чини коришћење софтвера отвореног кода [49] чиме се значајно редукују трошкови лиценцирања софтвера од стране образовних институција, а почињу да се користе и решења која у многоме аутоматизују разне пратеће процесе [50] чиме се избегава понекад јако комплексна и нејасна администрација и конфигурисање система. Битно је истаћи да се у процесу примене савремених решења виртуелизације у образовном домену губи пристутност негативног става студената усмереног ка припреми личних окружења за употребу у сврху образовања и одвијања наставног процеса будући да студенти кроз концепт виртуелизације добијају окружење унапред припремљено за сам наставни процес [51].

Светска пандемијска криза која је наступила 2020. године посматрана са становишта високошколског образовања и примене виртуелизације у истом указала је на све предности које виртуелизација може понудити савременом образовању не само у регуларним околностима, већ и када се ради о драстичним променама окружења у коме треба остварити образовни процес. На неких начин услед пандемијских услова дат је одговор какво је тренутно стање виртуелизационих решења у образовању и који су будући правци које треба остварити. Поједина истраживања која су вршена указују на значајне доприносе које је виртуелизација остварила у побољшању услова у којима се одвија образовни процес [52], [53]. Међутим, добре резултате прати и чињеница да постоје моменти када постојећа инфраструктура није могла адекватно одговорити на постојеће захтеве па треба значајније радити на њеном даљем унапређењу и адекватном уподобљавању за виртуелизационе процесе [54]. Примећено је да је у појединим моментима не само за студенте, већ и за наставно особље спровођење активности у виртуелном домену представљао прави изазов услед недостатка одговарајућих смерница, процедура, протокола, неаутоматизованих процеса [55] и да се наставно особље услед бављења низом пратећих техничких активности везаних за наставни процес, уместо самим наставним процесом, излагало тзв. синдрому прегоревања (eng. burnout syndrome) [56]. Уочено доводи до закључка да је процес виртуелизације неопходан у високошколском образовању на свим нивоима, али да треба бити свестан да то није процес који се може остварити у кратком временском периоду и да се адекватна трансформација на виртуелизациона решења не могу остварити преко ноћи [57], већ да то мора бити један континуиран процес који се мора спровести и спроводити у годинама које следе како би имали један одржив систем високошколског образовања у савременом и врло променљивом окружењу.

Као што је већ напоменуто у претходним разматрањима, виртуелизација данас представља један изузетно широки појам који обухвата многе концепте који међусобно деле многе сличности, али испољавају и међусобне разлике. У разматрањима која су заступљена у предметној докторској дисертацији дефинисаће се виртуелизациона архитектура као вишеслојна архитектура у којој су доминантна три основна слоја: виртуелна машина (eng. virtual machine), софтвер за виртуелизацију (eng. virtualization software) и одговарајући физички хардвер (eng. underlying physical hardware) [58]. У складу са поменутим везано за платформску, односно хардверску виртуелизацију, каква ће се користити у разматрањима у дисертацији, треба споменути постојање три основна типа реализације овакве врсте виртуелизације, а то су пуна (нативна) виртуелизација, парцијална виртуелизација и паравиртуелизација [59]. Што се тиче софтвера који омогућава виртуелизацију, овде треба напоменути да се у литератури срећу два назива која дефинишу овакав вид софтвера: монитор, односно контролор виртуелне машине (eng. Virtual Machine Monitor – VMM) и хипервизор (hypervisor) [60] која су идентична и могу се равноправно користити. У предметној дисертацији усвојиће се да се за софтвер за виртуелизацију користи термин хипервизор. Хипервизор се среће у две основне верзије: као хипервизор типа 1 који не захтева оперативни систем за своје извршавање и хипервизор типа 2 који се извршава на одговарајућем оперативном систему [61]. У свим разматрањима у оквиру докторске дисертације користиће се Oracleов VirtualBox [62] који представља хипервизор типа 2 и који подржава пуну виртуелизацију [63] о чему ће бити речи у наредним поглављима.

Овде треба напоменути да се поред употребе хиперевизора могу користити и друге технике за омогућавање виртуелизације, попут виртуелизације употребом контејнера (eng. Container Based Virtualization) или уникернела (eng. UniKernel), међутим овде ће се ипак користити хипервизор будући да испољава неке кључне предности у односу на остале технике попут већег степена стандардизованости, веће преносивости, боље могућности имплементације у изражено хетерогеним рачунарским окружењима и слично [64], [65]. Такође, употребом решења заснованих на употреби хипервизора постиже се врло висока изолованост окружења које се реализује у оквиру виртуелне машине чиме се омогућавају додатне безбедоносне могућности приликом оваквог начина имплементације виртуелизације, осим у ретким случајевима када сам хипервизор претрпи неки вид оштећења и почне да испољава рад ван уобичајених норми [66].

Могло би се рећи да је један од основних циљева виртуелизације постизање једног таквог нивоа у коме ће се извршити реализација виртуелног хардвера над стварним хардвером уз минимизацију комплексности тако добијеног система кроз реализацију добијања високе поузданости уз једноставно управљање целокупном конфигурацијом система [67]-[69]. Међутим, минимизацију комплексности није лако остварити увек и у свим случајевима на једноставан начин, будући да се у оквиру виртуелног окружења уочавају три велике целине на које треба обратити пажњу, при чему свака поседује своје особености, а то су: гостујући систем (eng. guest), односно систем који се реализује у оквиру виртуелне машине, затим систем домаћин (eng. host) који у суштини представља оригинални, физички рачунарски систем на коме се инсталира хипервизор и трећу целину представља сам хипервизор [70]. Сходно томе, треба обратити пажњу на следеће могуће везе интеракције: гостујући систем-хипервизор, гостујући систем-систем домаћин, систем домаћин-хипервизор [71]. У случајевима када над једним хипервизором постоји више виртуелних машина постојаће и везе између виртуелних машина, односно везе гостујући систем 1-гостујући систем 2. Као што се види постојање ових узрочно-последичних веза може у појединим ситуацијама значајно отежати савладавање комплексности виртуелног окружења. На све претходно морају се додати и везе које

парцијални системи остварују са самим собом, односно, постојаће и везе систем домаћин-систем домаћин, као и гостујући систем-гостујући систем.

Иако се говори о изолованости виртуелних машина и постојању виртуелног хардвера на страни виртуелних машина, не сме се одбацивати утицај виртуелног хардвера на физички хардвер и обрнуто. Код неких делова хардвера овај утицај је мање доминантан и лакше га је контролисати, али постоје делови хардвера који имају јако специфично понашање у домену виртуелизације, попут рачунарске меморије, чија се фиксна вредност додељује виртуелној машини приликом покретања исте, а само коришћење како рачунара, тако и виртуелне машине, додељује коришћењу меморије у готово свим аспектима изразито динамичку карактеристику. Зато и аутори у [72] истичу да је од пресудне важности поседовање изузетно доброг знања о коришћењу меморијских ресурса за сваку од виртуелних машина како би се могле спровести адекватне технике управљања меморијом са становишта виртуелизације. Када говоримо о техникама управљања меморијом у домену виртуелних машина, постоје различити приступи и различите имплементације. Неки од приступа се заснивају на употреби паравиртуелних решења [73], а чест је случај да се решивост меморијских проблема у неком специфичном домену виртуелизације не може остварити коришћењем само једне технике, већ у решењу коезистира више техника истовремено [74]. Такође, мора се споменути да је у појединим случајевима имплементације одређених техника управљања меморијом у домену виртуелних машина приметно постојање одређених бочних ефеката које доводе до смањења сложености управљања меморијом, али у исто време појачавају сложеност неких других компонената виртуелног система [75] што може довести до јављања и решавања неких нових изазова. Приликом управљања меморијским ресурсима код виртуелних машина данас су често заступљене тзв. технике балонирања меморије (eng. Memory Balooning Techniques) које дају могућност да, под одређеним условима, једна виртуелна машина може користити одређени део меморијских ресурса друге виртуелне машине и обратно да, под одређеним условима, једна виртуелна машина може уступити одређени део меморијских ресурса другој виртуелној машини [76]-[78]. Без обзира на то која се техника користила у управљању меморијским ресурсима, од пресудне важности је да она адекватно буде имплементирана у одговарајућем домену виртуелизационог решења, будући да су чести безбедоносни и разни други ризици управо везани за различите апсекте управљања меморијским ресурсима, од чега највећи број случајева се односи на неки од проблема који је повезан са потрошњом меморије [79].

Овде треба споменути да ће, у основи разматрања датих у предметној докторској дисертацији, бити изражен још један аспект рада са виртуелним машинама, а то је питање њихове миграције. Оно што је потребно истаћи, као доминантан проблем приликом миграције виртуелне машине, јесте остваривање правилне подешености виртуелне машине на одредишту на основу адекватно спроведених анализа доступних ресурса како на одредишту, тако и на изворној локацији виртуелне машине [80]. Како би се умањила комплексност процеса миграције виртуелне машине искоришћено је виртуелно окружење засновано на потпуној виртуелизацији. Потпуна виртуелизација, за разлику од других могућих типова виртуелизације у хипервизорима, поред добрих перформанси, омогућава и велики степен компатибилности који није условљен модификацијама на гостујућем систему саме виртуелне машине [81]. Тиме је процес миграције значајније поједностављен јер су избегнуте неке сувишне додатне манипулације са виртуелним машинама чиме је смањена и вероватноћа појаве грешке приликом стављања виртуелне машине у рад на одредишној локацији.

Поред горе наведеног, аутори у [81] истичу и да сама виртуелизација представља концепт који комбинује различите софтверске и хардверске технологије како би се

постигла одговарајућа апстракција, па у складу са тим су и софтверски и хардверски део интегралне целине предметне докторске дисертације.

1.5. Преглед излагања

У поглављу 2 је показано постојање изразито хетерегеног рачунарског окружења приликом реализације наставног процеса када се користи учење на даљину. Представљено је софтверско решење намењено идентификацији софтверских и хардверских компоненти система који се користе од стране корисника у наставном процесу приликом реализације задатака „од куће“. Посебно су представљене особености решења за сваку од три анализиране платформе – Windows, Linux, macOS. Представљен је тестни узорак над којим је вршено испитивање, као и добијени подаци приликом испитивања хетерогености рачунарског окружења над одговарајућим тестним узорком. На крају је извршена компарација добијених резултата са резултатима снимљеним у рачунарским лабораторијама.

Основни концепти везани за софтвер које би корисник користио за извођење задатака у оквиру наставног процеса на виртуелној машини представљени су у поглављу 3. Представљена су ограничења која се односе на традиционалне поступке који се примењују у инсталацији и подешавању софтвера од стране идентификованих корисника: кандидата, предавача, као и осталих заинтересованих страна у спровођењу наставног процеса (на пример техничко особље образовне институције). Анализирани су кључни елементи неопходни за правилну имплементацију софтвера у оквиру окружења које се користи под окриљем виртуелне машине и извршена је њихова правилна идентификација како би се могли применити у даљем поступку реализације самоадаптирајуће виртуелне машине.

Слично претходном поглављу, у поглављу 4 су представљени основни концепти реалног (стварног) хардвера који чини систем домаћина (host систем) на коме се извршава одговарајући хипервизор у оквиру ког се покреће одговарајућа виртуелизација, а потом су представљени и основни концепти виртуелног хардвера који се користи од стране самог хипервизора и виртуелне машине. Потом је извршена основна идентификација узрочно-последичних веза између реалног и виртуелног хардвера уз паралелну идентификацију свих ограничења који се могу испољити на релацији реални хардвер – виртуелни хардвер и обрнуто, на релацији виртуелни хардвер – реални хардвер. Анализирани су кључни хардверски елементи које треба правилно спрегнути у циљу оптималног рада виртуелне машине, али и оптималног рада самог система домаћина, уз правилну идентификацију свих оптималних параметара које треба предвидети и адекватно реализовати током имплементације самоадаптирајуће виртуелне машине.

На основу идентификованих софтверских и хардверских чинилаца у претходним поглављима, поглавље 5 представља опис предложеног решења за реализацију самоадаптирајуће виртуелне машине. Извршен је приказ развијених онтологија и развијеног софтвера на бази поменутих онтологија за самоадаптирање виртуелне машине са становишта софтвера који се инсталира на самој виртуелној машини. Представљени су развијени модели током развоја различитих аспеката рада самоадаптирајуће виртуелне машине. Извршен је приказ софтверског решења намењеног реализацији одговарајућих сервиса за праћење и документовање потребних стања реалног хардвера на систему домаћину (host машина) за Windows, Linux и macOS платформу. Идентично поменутом, извршен је и приказ софтверског решења намењеног реализацији одговарајућих сервиса за праћење и документовање потребних стања виртуелног хардвера на виртуелној машини под одговарајућом Linux дистрибуцијом.

Представљено је и одговарајуће софтверско решење за аутоматско управљање одговарајућим кључним елементима рада хипервизора. Приказана је софтверска имплементација решења која омогућава успостављање одговарајућих узрочно-последичних веза између реалног и виртуелног света коришћењем одговарајућих статистичких метода у анализи одговарајућих временских серија (time series analysis), као и предиктивне аналитике, елемената науке о подацима, вештачке интелигенције у предвиђању временских серија (time series forecasting). Представљена је интеграција претходно реализованих решења у хардверску самоадаптивност виртуелне машине. Приказано је иницијално конфигурисање виртуелне машине која треба да остварује своју самоадаптивност.

Добијени резултати приликом тестирања самоадаптивне виртуелне машине на различитим платформама и у условима различитих хардверских и софтверских конфигурација представљени су у оквиру поглавља 6. Приказани су и добијени резултати реалног рада развијеног решења на различитим хардверским и софтверским платформама.

Кроз поглавље 7 приказана је конкретна применљивост самоадаптирајућих виртуелних машина у оквиру остваривања наставног процеса. Представљено је на који начин самоадаптирајуће виртуелне машине побољшавају квалитет наставног процеса, као и како самоадаптирајуће виртуелне машине могу допринети бољем остваривању исхода учења. Такође указано је и на један шири аспект применљивости самоадаптирајућих виртуелних машина у односу на циљани аспект примене у области академског образовања.

У оквиру поглавља 8 дискутовано је о одговарајућим ограничењима која се испољавају или се могу испољити приликом коришћења самоадаптирајућих виртуелних машина како са техничког становишта, тако и са економског становишта, одговарајуће законске регулативе и сличног. Дат је и осврт на потенцијална даља истраживања и унапређења која се могу остварити на основу изложеног решења у докторској дисертацији.

2. ХЕТЕРОГЕНОСТ РАЧУНАРСКОГ ОКРУЖЕЊА У ДОМЕНУ ВИСОКОШКОЛСКОГ ОБРАЗОВАЊА

2.1. Појам хетерогености

Хетеро представља префикс који у преводу са грчког језика означава присутност неке различитости. У складу са тим, илустровани Оксфорд енглески речник [82] придев хетероген дефинише кроз два могућа приступа, као испољавање одређених разлика у карактеру или као испољавање одређене разноврсности у садржају. У домену рачунарства појам хетерогености је присутан више деценија у различитим облицима везивајући се за означавање разноврсних разноликости које су се јављале и које се јављају како у самој реализацији рачунарских система, тако и у примени рачунарски заснованих решења у разним областима. Може се рећи да хетерогеност, кроз своје различите варијације појмова на који се односи, прати развој рачунарства, односно како је само рачунарство напредовало у различитим аспектима, тако су се јављале неке нове особине хетерогености и нови начини њеног испољавања.

Као што је већ указано претходно, појам хетерогености, у домену рачунарства и рачунарских система, није релативно нов појам, будући да се он испољио одмах на почетку експанзије рачунарства и тежње да се разноврсни рачунарски производи, хардверски и софтверски, обједине у целовита решења где сви они међусобно комуницирају и размењују одговарајуће скупове података и информација [83]. Када се спомиње хетерогеност у рачунарском свету, у већини случајева се мисли на неку од имплементација хетерогеног рачунарства, које је и данас врло доминантно када постоји, на пример, изражена употребу графичких процесора поред централних процесорских јединица у пословима везаним за рад са криптовалутама, или у задацима који подразумевају изразиту употребу решења заснованих на примени алгоритама вештачке интелигенције [84-87]. Хетерогеност се доста испољава и у домену рачунарских мрежа и хетерогене рачунарске мреже представљају, такође, један од појмова који се често сусреће у рачунарској литератури [88]. Експанзија архитектура заснованих на edge, cloud и fog принципима доводе до разматрања разних облика хетерогених испољавања и у овим областима [89]. Напретком рачунарских технологија остварују се разноврсни рачунарски системи за превазилажење различитих стварних проблема, од оних једноставних имплементација заснованих на врло уском унифицираном скупу хардвера и софтвера који се користи, па све до рачунарских система изразито високих нивоа комплексности са израженом разноликошћу употребљаваног хардвера и софтвера. На пример, задњих година приметна је велика експанзија решења заснованих на коришћењу Internet Of Things (IoT) концепата где је потребно објединити разноврсност хардвера и софтвера у целовита решења, па се и овде проблем хетерогености третира као једно од питања високе важности [90,91]. Проблем хетерогености присутан је у готово свим пољима савременог рачунарства и третира се на одговарајуће начине, међутим, нису сви проблеми решени на одговарајући начин, а постоје и случајеви када су реализована решења унела и одговарајућу додатну дозу збуњености у већ комплексна подручја испољавања хетерогености. Тако на пример постоје случајеви где се појам хетерогена мрежа може односити на два сасвим различита појма, један из хардверског домена при чему се односи на хетерогену рачунарску мрежу [92], док у другом случају означава појам из софтерског домена где се односи на реализацију софтверске хетерогене мреже намењене учењу [93]. Оваквих примера у третирању хетерогености има много чиме се продубљују непознанице које владају у хетерогеном рачунарском свету уместо да се сложеност тумачења умањује.

Уколико би се претходна дискусија ограничила искључиво на домен примене рачунарских система у образовању, може се наћи велики број радова који третирају проблем хетерогености у овом сегменту. Услед специфичности задатака који се остварују применом различитих система за подршку учењу, овде се у највећем броју случајева хетерогеност разматра са становишта самог образовног процеса, уместо конкретизације проблема хетерогености усмереног ка самом скупу хардвера и софтвера који се користи. У већини радова примарна усмереност је ка хетерогености која се јавља међу студентима који похађају одређене курикулуме и предмете у оквиру тих курикулума и њиховим различитим стиловима учења, начинима усвајања градива, адаптације на околину у којој се остварује учење и сличним аспектима [94,95]. Поред студената, анализира се и хетерогеност међу самим предавачима и њиховим стиловима предавања, начинима реализације и приступу одређеним тематским целинама [96]. Када се посматрају сами системи за реализацију електронског учења, овде се углавном проблем хетерогености усмерава у два правца, први третира проблем реализације хетерогених садржаја у оквиру једног хомогеног система за електронско учење [97], док се у другом правцу истиче изразита хетерогеност која се јавља међу студентима у начину прихватања одређеног система за електронско учење који им се ставља на располагање и који ће бити коришћен у остваривању образовног процеса [98]. Услед концепције самих система за електронско учење и задатака који се представљају за њих у циљу остваривања позитивних исхода учења, сами системи за електронско учење користе изузетно велике количине разноврсних података у свом раду, од одређених лог и мета података до врло комплексних мултимедијалних података. Овде треба узети у обзир да системи за електронско учење напретком технологија поред строго структуираних података данас користе и врло велике количине неструктуираних и полуструктуираних података. У складу са тим постоји изразита хетерогеност података који се користе у оквирима система за електронско учење, па се услед овакве појаве доста пажње посвећује и третирању проблема хетерогености у домену употребе података у оквиру поменутих система [99].

Што се тиче аспекта хетерогености у системима за електронско учење, постоје и разматрања хетерогености која се тичу хардвера, али она нису уско усмерена на саме системе за електронско учење, већ су више ствар разматрања самих архитектура на којима се системи за електронско учење реализују, попут реализација система за електронско учење у облацима и слично. Претпоставља се да се не улази у разматрања хетерогености по питањима софтвера и хардвера јер се већина система за електронско учење реализују као веб оријентисани системи чиме се систем представља сваком кориснику на готово идентичан начин и чиме се стиче велики привид хомогености на највишем нивоу, па се не улази у дубља разматрања хетерогености која се остварује на нижим нивоима. Међутим, овде треба напоменути да се разматрања софтверске и хардверске хетерогености која се јавља у образовним процесима не треба тако лако одбацивати без обзира на велики напредак система за електронско учење који су значајно еволуирали последње деценије, будући да студенти у реализацији својих задатака приликом учења код куће неће користити само онлајн платформе за електронско учење, већ ће доста користити и своје личне рачунарске ресурсе који испољавају велики степен хетерогености и уносе нову комплексност у саму реализацију исхода учења када се образовни процес у целости или делимично спроводи по принципима учења од куће. У складу са тиме се управо и овде, у овој предметној докторској дисертацији, разматрања базирају на анализи проблема хетерогености који се испољава приликом коришћења личних рачунарских ресурса студената, у сврху реализације образовног процеса од куће, који испољавају огромну разноликост, а које треба адекванто разумети и уврстити у целокупни образовни процес у циљу добијања адекватних позитивних исхода учења.

За крај ових разматрања везаних за појам хетерогености, мора се споменути да иако хетерогеност није нов појам и одавно је присутан у рачунарском свету, то не значи да су новији аспекти рачунарства постали имуни на хетерогеност. Напротив, можда је појам хетерогености израженији и сложенији него икада и сусреће се и у актуелним концептима везаним за вештачку интелигенцију, неуронске мреже, машинско учење, безбедности података и сличном [100,101].

Пре уласка у даља разматрања која се тичу проблема хетерогености који ће бити анализирани и дискутовани у предметној докторској дисертацији, мора се напоменути да ће се разматрања вршити искључиво о хетерогености рачунарског окружења заснованом на рачунарима, преносним рачунарима, серверима и сличним рачунарским производима, док остали уређаји који испољавају рачунарске карактеристике у виду мобилних телефона, таблета, одређених паметних уређаја неће бити узети у разматрање. Овакво усмеравање истраживања и разматрања чини се из неколико разлога. Прво, одређена група производа искључена је из разматрања услед преобимности истраживања која би се десила уколико би ти уређаји били укључени у истраживање и уколико би се дискутовало о њима. Друго, иако се данас у извођењу савремене наставе и реализацији наставних активности у многоме користе мобилни и паметни уређаји различите природе, класични рачунарски производи у виду десктоп рачунара, преносних рачунара и сличних ипак остају доминантни будући да поседују одређене карактеристике у хардверском и софтверском смилу које већина мобилних уређаја још нема, или за сада нису у многоме прихваћени за једну ширу употребу. И на крају треба споменути, да, будући да се израда докторске дисертације у потпуности ослања на виртуелизационе технике, мобилни уређаји врло мало испољавају капацитете за реализацију виртуелизационих активности, па је у складу са тим примарни фокус, као што је наведено, у предметној докторској дисертацији на рачунарској опреми у традиционалном смислу, односно, на декстоп рачунарима, мини рачунарима, преносним рачунарима, серверима и сличној рачунарској опреми.

2.2. Значај правилне анализе хетерогености рачунарског окружења

Како би се разумела потреба за анализом хетерогености рачунарског окружења у коме се остварује образовни процес у високошколском образовању приликом реализације учења од куће, спроведена је одговарајућа анкета над реалним испитаницима који су подељени у две групе чиме су формирани одговарајући узорци. Прва група испитаника обухватила је студенте прве године основних академских студија Техничког факултета у Бору Универзитета у Београду, док је други узорак сачињен од студената четврте године основних академских студија студијског програма Рударско инжењерство који се реализује, такође, на Техничком факултету у Бору Универзитета у Београду. За узорковање су намерно одабрани студенти прве и четврте године основних академских студија како би се добио поглед на почетку и на крају првог степена студија.

Анкета је спроведена током школске 2022/2023. године и била је у потпуности добровољна и анонимна, без обухватања личних података било које врсте. Сви студенти одговарали су на идентичан скуп питања, при чему је првих седам питања из предметне анкете усмерено ка разумевању хетерогености која се јавља приликом употребе личних рачунарских ресурса студената. Тај скуп питања са понуђеним одговорима из предметне анкете приказан је у табели 1.

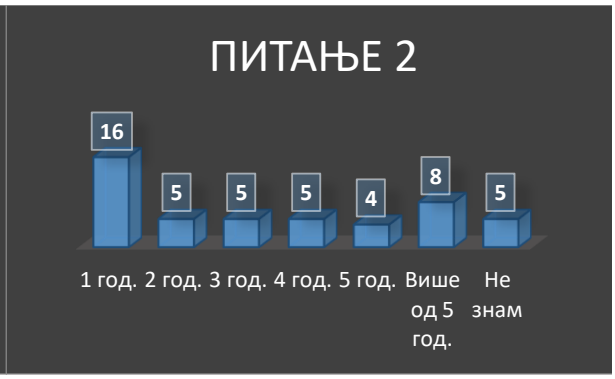
Табела 1. Садржај питања и понуђених одговора првих седам питања из реализоване анкете

Редни број питања	**Садржај питања и понуђених одговора**
1.	**ПИТАЊЕ:** Колико рачунара поседујете?
	ОДГОВОР: А) 1 ; Б) 2 ; В) 3 ; Г) 4 ; Д) 5 ; Ђ) Више од 5 ; Е) Не знам
2.	**ПИТАЊЕ:** Рачунар који користим за праћење наставе сам набавио/набавила пре:
	ОДГОВОР: А) 1 годину ; Б) 2 године ; В) 3 године ; Г) 4 године ; Д) 5 година ; Ђ) Више од 5 година ; Е) Не знам
3.	**ПИТАЊЕ:** На рачунару имам инсталиран оперативни систем:
	ОДГОВОР: А) Windows 7 ; Б) Windows 8/8.1 ; В) Windows 10 ; Г) Windows 11 ; Д) macOS ; Ђ) Linux ; Е) Није наведен на листи: _____ ; Ж) Не знам
4.	**ПИТАЊЕ:** Да ли сте од почетка школске године променили рачунар?
	ОДГОВОР: 1) Да ; 2) Не
5.	**ПИТАЊЕ:** Да ли сте од почетка школске године променили неки део у рачунару, или надоградили постојећу рачунарску конфигурацију неком новом компонентом?
	ОДГОВОР: 1) Да ; 2) Не
6.	**ПИТАЊЕ:** Да ли сте од почетка школске године имали неку од сервисних интервенција на свом рачунару (замена неисправне компоненте, реинсталација система и слично)?
	ОДГОВОР: 1) Да ; 2) Не
7.	**ПИТАЊЕ:** Да ли знате која вам је тренутна хардверска конфигурација рачунара? Ако не одговор на питање ДА укратко напишите хардверску конфигурацију.
	ОДГОВОР: 1) Да ; 2) Не _____ _____ _____

Као што се из претходне табеле види, питања су конципирана тако да дају одређени увид у неке од карактеристика личних рачунара које студенти користе током реализације задатака које захтева од њих реализација наставног процеса у оном сегменту који се реализује ван специјализованих рачунарских лабораторија високошколске институције. Оно што је значајно споменути, јесте да је концепција питања таква да додатно омогућава увид у то колико сами студенти заиста познају карактеристике рачунарске опреме која се налази у њиховом власништву и коју користе приликом реализације различитих врсти наставних задатака који се пред њих постављају.

Први узорак састављен је од одговора који су пружени од стране 48 анкетираних студената прве године основних академских студија Техничког факултета у Бору који су годину уписали у школској 2022/2023. години. Структура одговора на понуђена питања у оквиру поменуте анкете приказана је одговарајућим графиконима приказаних на сликама од 1 до 7.

Слика 1. Број забележених одговора у оквиру првог питања из анкете - први узорак.



Слика 2. Број забележених одговора у оквиру другог питања из анкете - први узорак.

Слика 3. Број забележених одговора у оквиру трећег питања из анкете - први узорак

Слика 4. Број забележених одговора у оквиру четвртог питања из анкете - први узорак

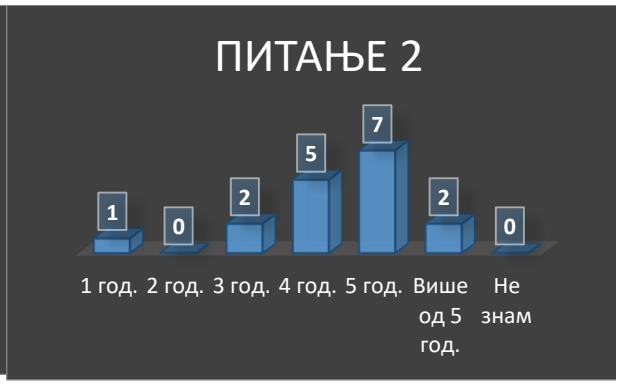
Слика 5. Број забележених одговора у оквиру петог питања из анкете - први узорак

Слика 6. Број забележених одговора у оквиру шестог питања из анкете - први узорак

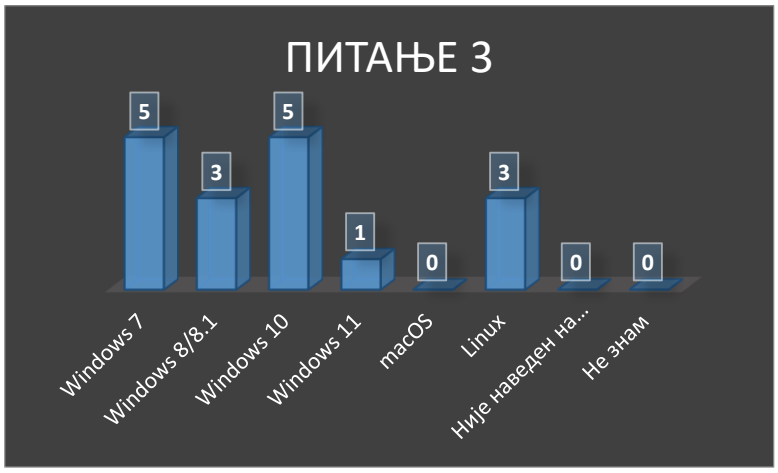
Слика 7. Број забележених одговора у оквиру седмог питања из анкете - први узорак

Други узорак састављен је од одговора који су пружени од стране 17 анкетираних студената четврте године основних академских студија студијског програма Рударско инжењерство Техничког факултета у Бору који су годину уписали у школској 2022/2023. години. Структура одговора на понуђена питања у оквиру поменуте анкете приказана је одговарајућим графиконима приказаних на сликама од 8 до 14.

Слика 8. Број забележених одговора у оквиру првог питања из анкете – други узорак.



Слика 9. Број забележених одговора у оквиру другог питања из анкете - други узорак.



Слика 10. Број забележених одговора у оквиру трећег питања из анкете - други узорак

Слика 11. Број забележених одговора у оквиру четвртог питања из анкете - други узорак

Слика 12. Број забележених одговора у оквиру петог питања из анкете - други узорак

Слика 13. Број забележених одговора у оквиру шестог питања из анкете - други узорак

Слика 14. Број забележених одговора у оквиру седмог питања из анкете - други узорак

Ако се погледају добијени резултати одговора на прво питање, уочава се да скоро 40% испитаника из првог узорка (слика 1) поседује више од једног рачунара, док се у другом узорку проценат испитаника који поседује више од једног рачунара креће око 41% (слика 8). То значи да постоји реална могућност да ће неко од тих 40% испитаника променити рачунар током реализације наставних задатака, или ће можда током наставног процеса користити више рачунара паралелно, при чему је вероватноћа да то буду хардверски и софтверски идентични рачунари врло мала. Може се рећи да ће постојати пораст хетерогености јер на 48 испитаника у првом узорку неће бити 48 рачунара, а такође и на 17 испитаника у другом узорку биће број рачунара који је већи од броја испитаника. Све аспекте хетерогености треба адекватно сагледати јер се жели омогућавање идентичног окружења за рад на задацима на свим рачунарима свих студената.

Међутим, није само број рачунара тај који повећава хетерогеност, односно није само количина рачунарске опреме та која чини границу између адекватног и неадекватног сагледавања хетерогености. Ако се погледа слика 2 за први узорак, односно слика 9 за други узорак, примећује се да постоје велике разлике међу испитаницима по питању старости рачунара које користе у наставном процесу. Овде треба напоменути да су испитаници одговарали када су набавили рачунар, а не које године је рачунар произведен, па узимајући у обзир да приликом куповине реално рачунар може бити произведен и доста раније, могу постојати и неке значајније осцилације у погледу старости рачунара, међутим то не утиче значајније на разматрања која овде износимо. Дакле, у складу са претходним, биће великих варијација у старости рачунара, а самим тим биће и значајнијих осцилација у хардверу на коме су рачунари базирани. То значи да ће и овде бити додатног усложњавања по питању хетерогености јер ће постојати

захтеви за комплекснијим анализама хетерогености како би се могле обухватити различите генерације и типови хардвера. Треба имати на уму да оне карактеристике и параметри који су важили за једну генерацију хардвера не морају важити и за неку другу генерацију хардвера што, како је већ поменуто, повећава степен хетерогености и саму комплексност даљих анализа усмерених ка правилним идентификацијама параметара хетерогености и савладавања исте.

Слично испољавање биће и приликом разматрања смештених у домен коришћеног софтвера, односно постојаће, такође, одређена пораст хетерогености али за разлику од претходног случаја, где је тај пораст базиран на хардверу, овог пута је пораст базирана на разноликости оперативног система који испитаници користе на својим рачунарима. Овде ће се прво размотрити резултати другог узорка. За други узорак заступљеност оперативних система дата је на слици 10 и види се велики степен хетерогености по питању различитих верзија оперативних система који се користе од стране испитаника, али се такође јавља и одређена мања хетерогеност у погледу самих типова оперативних система који се користе приликом реализације наставних задатака. Ако се сада погледају резултати везани за први узорак који су представљени на слици 3, јавља се нешто другачија појава. Овде постоји, такође, хетерогеност по питању употребљаваних верзија оперативних система, али доминантан је процентуални удео Windows 10 оперативног система од 75%, па су остали процентуални удели далеко мањи и тиме се умањује комплексност приступа у анализи хетерогености јер постоји далеко мање осцилација у узорку. Али то не значи да постоји апсолутно одсуство хетерогености и да се она може занемарити, што се у оваквим случајевима доста често чини под изговором коришћења тренда већине. Напротив, иако постоји смањења комплексност, хетерогеност је и даље присутна и она се мора адекватно третирати без обзира на одсуство неких статистички показатеља или њихово испољавање у врло малим вредностима. Овде треба споменути и могуће постојање узрочно-последичне везе између хетерогености која се испољава на основу хардверских карактеристика и хетерогености која се испољава на основу софтверских карактеристика, односно у овом случају та софтверска хетерогеност посматрана је кроз оперативне системе који су покренути на поменутом хардверу. Наиме постоји врло извесна могућност да ће рачунари различитих старосних група, односно различитих генерација хардвера имати и различите оперативне системе у складу са одговарајућом хардверском и софтверском подржаношћу, перформансама и осталим сличним релевантним факторима. На пример, многи рачунари не могу да користе Windows 11 оперативни систем иако се ради о рачунарима који не поседују неку велику старост, али не поседују адекватну хардверску подржаност за овај оперативни систем, па ће корисници инсталирати Windows 10 који је подржан од стране релевантног хардвера. Може постојати и случај преласка са значајно старијег оперативног система на новији на поприлично старим рачунарима. На пример постоје корисници који прелазе на Windows 10 са Windows 7 оперативног система иако имају одређену деградацију перформанси, међутим Windows 7 више није подржан никаквим надоградњама и осталим пратећим активностима будући да је подршка за тај оперативни систем укинута.

У претходним разматрањима указано је на неке кључне разлоге зашто се мора правилно узети у разматрање хетерогеност рачунарског окружења у коме се остварује наставни процес у високошколском образовању када се тај процес делимично или у целости реализује по принципима учења од куће, односно када постоји значајнија употреба личних рачунарских ресурса студената. Међутим, указаће се на још неке додатне појаве које се могу дешавати а које могу директно или индиректно утицати на промену хетерогености, као и на њену правилну анализу и интерпретацију.

На пример, може се десити ситуација да у току школске године студент изврши куповину рачунара и у складу са извршеном куповином промени рачунар који користи

или га почне користити паралелно са већ постојећим рачунаром што значи да можемо постојати одговарајућа промена хетерогености у току школске године. Иако делује да ово нису неки значајнији проценти, резултати анкетирања ипак указују да и ову појаву треба узети у разматрање. Испитаницима је постављено директно питање да ли су од почетка постојеће школске године извршити промену рачунара. Испитаници у другом узорку су у целости одговорили негативно на постојеће питање (слика 11), али код испитаника у првом узорку случај је да је 6 од 48 испитаника одговорило потврдно на постављено питање (слика 4). На први поглед овај број не делује толико значајно, међутим преведено у проценте, он указује на то да је позитиван одговор дало преко 12% од укупног броја испитаника што није мали проценат када се сагледава утицај на хетерогеност и може довести до повећања хетерогености. Ова појава постаје још израженија ако се питање преформулише и не обухвати рачунар у целости, већ примени компонентни приступ, па је у складу са тим испитаницима постављено питање да ли су од почетка постојеће школске године променили неки део у рачунару, или надоградили постојећу конфигурацију неком новом компонентом. Потврдан одговор на постављено питање дало је скоро 19% испитаника у првом узорку (слика 5), односно скоро 18% испитаника у другом узорку (слика 12), што представља значајан број испитаника, а самим тим и потенцијално значајне промене у већ присутној хетерогености рачунарског окружења. На крају, анкетирањем, постојала је потреба да се укаже на још један аспект који може довести до промене у присутној хетерогености разматраног рачунарског окружења. Поред евентуалне куповине новог рачунара, или промене неке компоненте у већ постојећем у смислу надоградње, постојала је потреба да се укаже и на ситуације када се конфигурација рачунарске опреме може значајније променити услед дефекта и активности усмерених на отклањању испољеног дефекта. У складу са тим, испитаницима је постављено и питање да ли су од почетка школске године имали неку сервисну интервенцију на рачунару било да се ради о хардверској или софтверској интервенцији. Потврдно на постављено питање одговорило је 21% испитаника у првом узорку (слика 6), док је у случају другог узорка преко 35% испитаника пружило потврдан одговор, што указује на значајан потенцијал у промени хетерогености рачунарског окружења. Увидом у претходна разматрања, види се да не треба занемарити и само понашање корисника у смислу набавке нове, надоградње, као и одржавања постојеће личне рачунарске опреме, која се користи од стране студената приликом реализације наставних задатака, јер, како је показано, све ове активности значајније могу довести до промене већ постојеће изражене хетерогености рачунарског окружења и усложњавати додатно даље активности на третирању исте.

На основу свега претходно изложеног види се неопходност правилне анализе хетерогености рачунарског окружења у коме се одвија образовни процес како би се адекватно сагледала хетерогеност и одговорило на изазове и додатну комплексност коју она уноси у сам образовни процес. Од изузетне важности је развити једнозначни поступак анализе хетерогености рачунарског окружења како би се могли остварити адекватни увиди у постојеће рачунарско окружење над којим треба омогућити реализовање одговарајућих задатака образовног процеса на један униформни начин без обзира на локално окружење (лични рачунар студента) у коме ће се реализовати. Треба развити такве поступке који ће омогућити аутоматизацију целокупног процеса сагледавања поменуте хетерогености при чему је потребно учешће студената у датом поступку свести на минимум. Ово је од изузетног значаја јер је показано да студенти нису у стању дати објективне и тачне одговоре о свом рачунарском окружењу из мноштва различитих разлога. То је показано и путем предметне анкете где је једно од питања гласило да ли испитаници знају коју тренутно хардверску конфигурацију рачунара користе и ако је одговор потврдан тражено је да се она напише у кратким

цртама. У првом узорку око 3% испитаника је дало потврдан одговор и написало своју тренутну рачунарску конфигурацију, што је приказано на слици 7, док у другом узорку таквих испитаника уопште није ни било, односно сви одговори на дато питање били су негативни, као што се види са слике 14.

Имајући претходно у виду и поштујући начело тачности прикупљених података, развијени су одговарајући аутоматски механизми за анализу хетерогености рачунарског окружења који ће бити разматрани у редовима који следе.

2.3. Поступак анализе хетерогености рачунарског окружења

Приликом развоја решења које ће омогућити правилну имплементацију поступака анализе хетерогености рачунарског окружења, јавила су се два почетна питања на које је требало одговорити, а потом развој усмерити сходно тим одговорима. Прво питање јесте који је то скуп података које треба прикупити како би се каснијим тумачењима могла адекватно сагледати целокупна хетерогеност и друго како обезбедити прикупљање тих података с обзиром на специфичности и различитости оперативних система који се тренутно користе на рачунарима.

Обим података који се може прикупити о свакој рачунарској конфигурацији је импресиван и сваки од тих података има неку своју одређену улогу у рачунарском систему, било да се ради о некој контролној, управљачкој, информативној или некој другој улози. Међутим, то не значи да је у свакој ситуацији, односно приликом третирања неког одређеног проблема, сваки од података значајан и да сваки од података у датом временском тренутку има исту важност. У складу са тим приоритетни задатак, у почетку рада на развоју предметног решења, била је правилна идентификација података који имају значај у анализи хетерогености, водећи при томе посебно рачуна да начело минималности потребних података буде поштовано у целости. Минимизација потребних података које је потребно прикупљати се мора реализовати како са становишта смањења комплексности реализованог система, тако и са становишта перформанси система које ће се испољити приликом његовог реалног рада, односно рада у продукцији. На пример, рачунар прати температуру неких својих компоненти преко одговарајућих температурних сензора (температура кућишта, централног процесора, графичког процесора, диска и слично). Подаци који се генеришу са ових сензора су од виталног значаја за сам систем, јер на основу њих систем добија адекватну информацију о тренутној загрејаности одређених компоненти и на основу те информације производи одговарајуће дејство, односно ако се повећала температура неке компоненте може повећати брзину рада вентилатора, или ако је достигнута горња гранична вредност може предузети акцију гашења целокупног система у циљу превенције настајања значајнијих дефеката на самом систему. Као што се из приложеног види један јако значајан мерни подсистем рачунарског система. Међутим, питање је да ли ови подаци имају једнаки значај за сагледавање хетерогености рачунарског окружења и да ли их треба прикупљати у истом обиму и третирати их у оквиру хетерогености са истим значајем? Одговор је не, будући да ови подаци немају никаквог значаја за анализу хетерогености, па их, без обзира на њихов значај у неком другом домену, не треба прикупљати и тиме усложњавати и додатно оптерећивати предвиђено решење које се развија за домен анализе предметне хетерогености. Оваквих случајева је доста, па треба бити обазрив приликом селекције података који ће се касније прикупљати.

Након детаљне анализе разних сегмената рачунарског система, идентификоване су кључне компоненте које највише утичу на хетерогеност рачунарског окружења, а то су централна процесорска јединица, графички подсистем, дискови, радна меморија и мрежни адаптер. Свака од ових компоненти има мноштво карактеристика и података које

је описују, тако да се морало приступити даљој селекцији како би се дошло до крајњег скупа података који ће се прикупљати. Приликом наведене селекције водило се рачуна да подаци, који ће се прикупљати о наведеним компонентама, буду у највећој могућој мери добри репрезенти утицаја неке компоненте на целокупну хетерогеност рачунарског окружења.

У складу са претходним за централну процесорску јединицу одлучено је да минимални скуп података који ће се прикупљати буде састављен од следећих података:

- назива произвођача процесора,
- назива самог процесора,
- броја језгара,
- броја логичких језгара.

За графички подсистем прикупљаће се једино називи одговарајућих графичких картица које улазе у састав графичког подсистема. Уколико рачунарски систем поседује и интегрисану и дискретну графичку картицу извршиће се прикупљање назива обе графичке картице, или уколико је графички подсистем састављен од више графичких картица биће унети називи за сваку од њих. За разлику од претходно описаног прикупљања података за процесор, овде се примењује само прикупљање назива графичких картица будући да у моменту реализације предметне докторске дисертације нису биле доступне све могућности прикупљања података у библиотекама које покривају ове сегменте рада. Очекује се да врло брзо и остале могућности постану доступне, па се у том моменту дато решење може лако модификовати и проширити по потреби и за остале параметре. Иако се обезбеђује мањи скуп података него код неких других компоненти и сам назив графичких картица може бити довољан за правилну анализу хетерогености рачунарског окружења потребну за рад у предметној докторској дисертацији и не утиче на општост решења.

У погледу меморије коју поседује рачунарска конфигурација, одлучено је да се, у складу са реализацијом минималног сета података који се прикупља, прикупља једино податак о количини радне меморије која је стварно доступна кориснику. То значи да ће се прикупљати податак не о капацитету уграђене меморије, већ о количини која је доступна кориснику за даљи рад након што се изврши умањење доступне меморије по разним основима (на пример, умањи се количина доступне меморије за количину дељене меморије са интегрисаном графичком картицом и слично). Ово је податак који је довољан за опис хетерогености у случају предметне докторске дисертације, међутим треба поменути да се хетерогеност остварује и по основу других параметара везаних за меморију попут типа меморије, њене брзине, латенције и сличних. Будући да претходно наведени подаци нису неопходни за даљи рад у предметној докторској дисертацији, они неће бити узимани у обзир без обзира на њихову присутност, већ ће се бележити само, како је већ речено, количина радне меморије јер је тај податак неопходан за даље успешно реализовање решења предложених у овој докторској дисертацији. Оваква концепција не утиче на општост и поузданост решења изложеног у докторској дисертацији.

За дискове приликом разматрања треба напоменути да они имају, као и свака рачунарска компонента, неке своје основне карактеристике у погледу типа диска, капацитета и сличних, али да овде треба узети у разматрање и неке особености које се испољавају употребом различитих оперативних система. Сваки оперативни систем користи свој датотечни систем па ће у складу са тим и рад са дисковима имати своје различитости у зависности од тога који се оперативни систем буде примарно користио на датом рачунару. У складу са овим чињеничним стањем, одлучено је да се за дискове

узме један шири скуп података који ће бити адекватни репрезент хетерогености не само диска као компоненте већ диска као компоненте стављене у функцију рада одређеног оперативног система, па ће се на овом месту прикупити подаци о партиционисаности сваког од дискова који улазе у састав рачунарског система. Овде се посебно водило рачуна да се направи таква концепција прикупљања података која ће омогућити да се представљени сет података може поуздано и ваљано искористити на различитим оперативним системима без икакве додатне манипулације истим или захтевања неке додатне интеракције са корисником, како би се повећао степен аутоматизације и умањила вероватноћа да подаци који се прикупе буду погрешно интерпретирани. Сходно томе, узимајући у обзир карактеристике најпознатијих оперативних система данашњице, Windowsa, Linux дистрибуција и macOSa, након анализе датотечних система поменутих оперативних система, дошло се до следећег скупа података који ће се прикупљати:

- ознака партиције,
- укупан простор који заузима партиција,
- тренутни слободан простор у оквиру посматране партиције,
- путања на којој се партиција налази.

Овако конципирани подаци пружају и један шири увид у хетерогеност рачунарског окружења у односу на увид који би се остварио прикупљањем само података који представљају техничку спецификацију сваког од дискова који улази у састав одређене рачунарске конфигурације.

Слична ситуација се испољава и код мрежних адаптера. И у овом случају могло би се прикупљати мноштво података који представљају техничку спецификацију самих мрежних адаптера који улазе у састав рачунарске конфигурације и они би указивали и објашњавали хетерогеност, али за само подручје рада докторске дисертације, овакав скуп података би био сувишан. Уместо тога за даља разматрања од веће важности је да се изврши прикупљање два кључна податка, а то су остваривање реалних брзина мрежне конекције у стварном раду и то брзине којом се подаци примају (брзина downloada), као и брзине којом се подаци шаљу (брзина uploada), који су овом случају довољни за описивање хетерогености, па ће се у складу са тим скуп података који се прикупља сатојати од:

- остварене download брзине добијене спровођењем адекватне серије тестова и
- остварене upload брзине добијене спровођењем адекватне серије тестова.

Поред хардвера, постоји и одређена софтверска карактеристика саме хетерогености која се испољава, будући да је рад рачунара заснован на коришћењу одговарајућег оперативног система. Зато је одлучено да се прикупљају и одређене информације о самом оперативном систему који се користи на циљаној рачунарској конфигурацији. Како би се извршила правилна идентификација коришћеног оперативног система, одлучено је да скуп података који се прикупља буде следеће садржине:

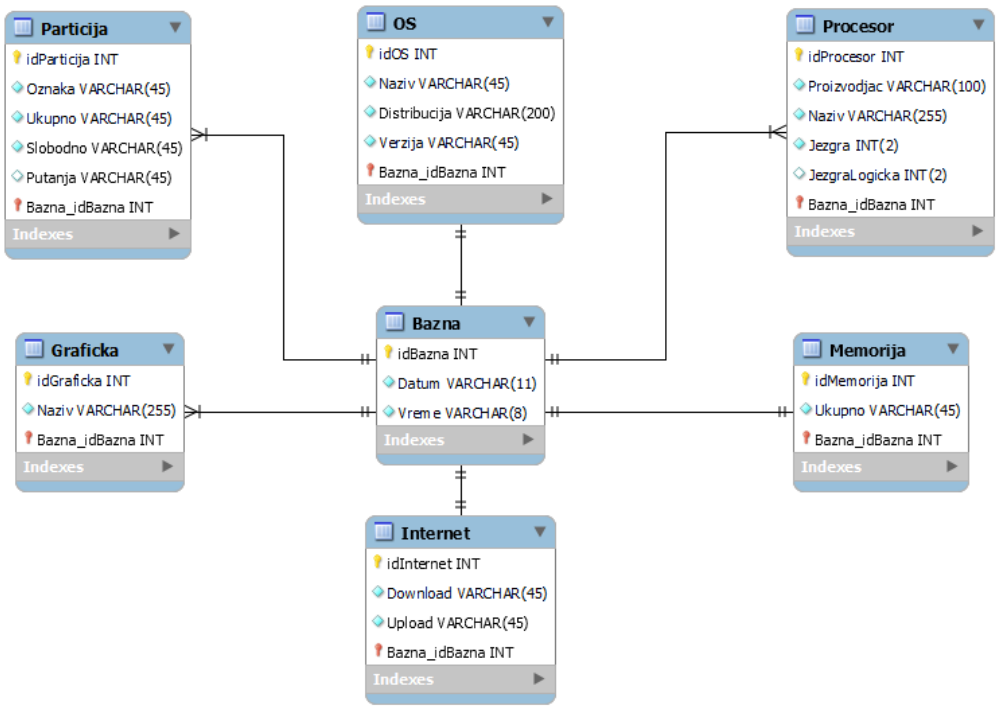
- назив употребљеног оперативног система,
- ознака дистрибуције употребљеног оперативног система и
- тренутна верзија оперативног система који се користи на предметном рачунару.

Овако конципиране ознаке и скуп података покрива могућност правилног идентификовања најпознатијих оперативних система који су данас у употреби на

рачунарима а то су Windows у верзијама 7, 8/8.1, 10 и 11, разноврсне Linux дистрибуције и различите верзије macOS система.

Треба напоменути да се подаци прикупљају у изворном облику, односно у облику какав је пружен од стране самог произвођача, па може у појединим случајевима доћи до одређених одступања од тржишних ознака које су више маркетиншког него техничког карактера. Ова одступања немају никаквог утицаја на квалитет прикупљених података, нити имају утицаја приликом касније анализе добијених података, али свакако у појединим моментима се може јавити сумња у валидност добијених података, па је у складу са тим овде дата претходна напомена.

Након извршене потпуне идентификације свих релеватних података који ће се прикупљати како би се остварио адекватан увид у хетерогеност рачунарског окружења, потребно је обезбедити правилан начин складиштења прикупљених података. Будући да се ради о структуираним подацима, одлучено је да се складиштење изврши употребом одговарајуће релационе базе података. Као систем за управљање релационим базама података одабран је MariaDB Server верзије 10.4.27 који представља бесплатан софтвер отвореног кода [102], па је у складу са лиценцним правилима софтвера у потпуности дозвољено његово коришћење у домену за који је овде коришћен. MariaDB Server има компатибилност са MySQL Server системом за управљање базама података, како у погледу дефиниција табела и података који се складиште у оквиру базе података, тако и у погледу компатибилности клијентских протокола [103]. Ова компатибилност омогућила је да се комплетан развој одговарајуће релационе базе података изврши употребом MySQL Workbench алата, при чему је коришћена верзија 8.0.32 у Community варијанти. Овај алат изабран је због широке палете својих могућности која обухвата моделирање, развој заснован на SQL, конфигурацију и администрацију сервера, креирање резервних копија, извоз података и слично [104]. У поменутом алату извршено је креирање одговарајућег EER (Enhanced Entity-Relationship) модела, коришћењем стандардних концепата и симбола за приказ [105]. Графички приказ добијеног модела представљен је на слици 15.



Слика 15. EER модел на основу кога је реализована одговарајућа релациона база података за прикупљање података о хетерогености рачунарског окружења

На основу добијеног модела, одговарајућим поступцима превођења, врши се трансформација свих описа садржаних у EER моделу у одговарајући скуп SQL наредби које се обједињују у оквиру SQL скрипт датотеке. Наредбе садржане у оквиру поменуте скрипт датотеке систем за управљање базама података је у стању да препозна, изврши одређене акције и формира одговарајућу базу података на основу резултата извршења тих акција, будући да поменут сервер у потпуности подржава рад базиран на SQL, па се сходно томе скрипт датотека прослеђује на даље процесирање серверу. Након процесирања на одговарајућем серверу, у оквиру поменутог система за управљање базама података, реализује се физичка интерпретација одговарајуће релационе базе података која је у потпуности спремна за прихват и упис добијених података.

Поред реализације дела који омогућава прихват и складиштење добијених података како би се исти могли касније анализирати и извести одређени закључци о хетерогености рачунарског окружења, потребно је реализовати и део решења које ће омогућити само генерисање потребних података на извору и њихово слање до одредишта које је у овом случају представљено реализованом релационом базом података. Ово је један комплексан процес јер треба омогућити рад на генерисању и слању података за различите оперативне системе, односно за данас најзаступљеније оперативне системе Windows, Linux и macOS. Поред неких заједничких ствари и именитеља који се испољавају међу овим оперативним системима, постоји велика разноликост међу истима, па није могуће направити универзално решење, већ се за сваку фамилију оперативних система мора реализовати посебан производ како би се могла остварити пуна функционалност прикупљања података за анализу предметне хетерогености. У складу са тим, иако обављају идентичне задатке и прикупљају исти скуп података, реализована су четири засебна софтвера, два за Windows оперативни систем и по један за Linux дистрибуције и macOS оперативне системе. Реализација поменутих софтвера извршена је коришћењем Python програмског језика.

Релеватне информације о оперативном систему за Windows прикупљају се употребом *platform* библиотеке [106] позивањем функције *uname()* и читањем параметра *release* за добијање ознаке дистрибуције, односно параметра *version* за добијање података о верзији оперативног система. Доступност ових информација, када се ради о Linux верзији програма, обезбеђује се библиотеком *distro* [107] и позивањем функције *id()* за добијање података о дистрибуцији и функције *version()* за добијање података о верзији. Код реализације намењене macOS системима није се могла наћи релеватна библиотека која би обезбедила адекватне функције за приступ наведеним подацима, па су се тражене функционалности реализовале употребом екстерног програма који се налази у саставу macOS система. У овом случају коришћењем *subprocess* библиотеке [108] и *run()* функције у оквиру исте реализован је као shell рутина позив програма *sw_vers* [109] и резултат тог позива снимљен у одговарајућу текстуалну променљиву. Потом су из те текстуалне променљиве издвојене одговарајуће линије текста које су садржале податке о дистрибуцији и верзији macOS система.

Подаци о централној процесорској јединици се за случај Windows и macOS система прикупљају на идентичан начин. Идентификација произвођача и модела процесора врши се употребом параметара *brand_raw* и *vendor_id_raw* функције *get_cpu_info()* из библиотеке *cpuinfo* [110]. Број логичких језгара добијен је употребом функције *cpu_count()* из библиотеке *psutil* [111]. На сличан начин одређује се и стварни број језгара, само се приликом позива функције то експлицитно наглашава, односно позив функције се врши као *cpu_count(logical=False)*. У случају неке од Linux дистрибуција основне податке о централној процесорској јединици у виду идентификације произвођача и модела добијају се на идентичан начин оном описаном за случај употребе Windows и macOS система. Међутим, поступак одређивања броја стварних и логичких

језгара у овом случају има другачији облик. Чињеница је да се и у овом случају број језгара може одређивати на идентичан начин претходно описаном, али је током тестирања на Linux системима уочено да употребом описаног начина долази до погрешне идентификације вредности у појединим случајевима, а оваква појава је нарочито изражена код неких AMD процесора. Зато је у случају употребе Linuxa, одређивање броја стварних и логичких језгара извршен употребом одговарајућег екстерног програма. Коришћењем *subprocess* библиотеке и *run()* функције у оквиру исте реализован је као shell рутина позив програма *lscpu* [112] и резултат тог позива снимљен у одговарајућу текстуалну променљиву. Потом су одговарајућим поступцима из мноштва информација које се добијају на основу реализоване команде о самом процесору, адекватним поступцима издвојене оне које дају увид у стварни и логички број језгара. Овим је избегнута појава поменутих спорних случајева и постигнуто је да се у погледу броја језгара добијају увек стварни и тачни подаци.

Добијање информација о количини расположиве меморије у оквиру посматраног рачунарског система извршено је на идентичан начин за сва три посматрана случаја, односно принцип прикупљања поатака је у потпуности идентичан за Windows, Linux и macOS системе. Коришћена је библиотека *psutil* и из ње извршен позив функције *virtual_memory()* која даје основне информације о разним параметрима меморије на посматраном систему, а потом је коришћењем параметра *total* издвојен податак који је од значаја за наше посматрање система. Будући да се добијена вредност добија у бајтовима која је због великих вредности доста непрегледна и отежћавајућа за читање и може довести до погрешне интерпретације добијених резултата, коришћењем *bytes2human()* функције извршена је конверзија добијених вредности у гигабајте, чиме је значајно повећања прегледност добијених података.

За анализу и прикупљање података о доступним партицијима у оквиру Windows оперативног система искоришћен је екстерни Microsoftов програм под називом *Windows Management Instrumentation (WMI)* [113] који је позиван са одговарајућим параметрима за приказ информација о партицијама. Само позивање вршено је на већ поменутом shell принципу коришћењем претходно описаних *subprocess.run()* поступака. Након тога су извлачене потребне информације о свакој појединачној партицији коришћењем прикладних поступака. Овде треба напоменути да је у сам програм уграђен механизам који спречава појаву грешака приликом читања партиција, односно који омогућава да се у обзир узму само HDD или SSD партиције, а да се партиције које потичу од неких других прикључених уређаја (екстерне USB меморије, CD/DVD и слично) одбаце и не узимају у разматрање јер за тумачења у предметној докторској дисертацији те партиције нису валидне. За разлику од Windowsa, у оквиру Linux и macOS система анализа и прикупљање података о партицијама на дисковима вршена је на исти начин коришћењем функција датих у оквиру *psutil* библиотеке. За идентификацију партиција употребљена је функција *disk_partitions()* заједно са својим *mountpoint* параметром, док је за добијање информација о укупном простору, као и тренутно доступном простору, у оквиру партиције коришћена функција *disk_usage()* са својим параметрима *total* и *free*.

За анализу и добијање података о елементима графичког подсистема код Windows базираних система поново је искоришћен *WMI* коришћењем поступака сличним претходно описаним, с тим што су овај пут прослеђени параметри за добијање информација о графичком подсистему. Након добијања обједињених података, врши се екстракција података посебно за сваку од присутних графичких картица у оквиру графичког подсистема, без обзира на њихов тренутни статус, односно на њихову активност. То практично значи да у случају да је у моменту прикупљања података активна интегрисана графичка картица, док је дискретна графичка картица присутна али у датом временском тренутку неактивна, подаци ће ипак бити прикупљени за обе

графичке картице, без обзира на њихов тренутни статус. Тиме се постиже да прикупљени подаци буду тачни и да одсликавају реално, а не тренутно стање у коме се налази посматрана рачунарска конфигурација. За Linux дистрибуције, идентификације везане за графички подсистем остварене су употребом екстерног програма *lspci* [114] кроз већ спомињани shell приступ заснован на употреби *subprocess.run()* функције. Из мноштва прикупљених података о различитом хардверу у оквиру посматране конфигурације врши се потом издавајање релевантних података о графичким картицама. И овде се примењује принцип да се прикупљају подаци о свим графичким картицама присутним у систему без обзира на њихов тренутни статус. И код macOS система прикупљање података о графичким картицама засновано је на употреби спољног програма, који је у овом случају заснован на употреби *system_profiler* [115] програма уз позивање одговарајућих параметара за приступ подацима везаним за графички подсистем. Аналогно претходним разматрањима, поново се користи shell приступ базиран на *subprocess.run()* својствима уз екстраковање информација о свим графичким картицама присутним у систему без обзира на њихов тренутни статус.

Како је већ напред поменуто, у погледу мрежних адаптера неће бити прикупљани подаци о хардверу, већ ће се прикупити подаци о брзини конекције, односно о брзини која се остварује како приликом преузимања, тако и приликом отпремања, користећи одговарајуће тестирање. Идентична процедура реализује се за сва три оперативна система, Windows, Linux дистрибуције и macOS. Тестови који се спроводе за добијање остварених брзина приликом тренутне конекције реализују се коришћењем одговарајућег APIја који је део *speedtest-cli* софтвера [116] и који своје тестове базира на идентичним серијама тестова који се користе путем сајта *speedtest.net* [117] који користи велики број корисника широм света, а присутно је коришћење и од стране многих водећих телекомуникационих компанија широм света који пружају подршку пројекту постављањем одговарајућих speedtest сервера на својој инфраструктури. Услед огромне светске популарности и прихватљивости ове батерије тестова од стране водећих телекомуникационих компанија, изабрано је да се и у предметној докторској дисертацији врше тестирања базирана на овој серији тестова. Имплементација се заснива на коришћењу библиотеке *speedtest*, из које се функцијом *Speedtest()* врши покретање одговарајућег тестирања, а потом се функцијама *download()* и *upload()* преузимају одговарајуће вредности за брзине преузимања и отпремања редом. Како би подаци били разумљивији и једноставнији за интерпретацију, у оквиру развијеног софтвера, одговарајућим математичким формулама, добијене вредности преводе се у вредности исказане у Mbps пошто се најчешће остварене брзине исказују управо у овим јединицама.

У претходним разматрањима приказани су механизми прикупљања свих података који су релевантни за касније анализе у погледу хетерогености рачунарског окружења. Приказане су све специфичности рада на три таргетирана оперативна система Windows, Linux, macOS. Прикупљене податке је потребно отпремити на одговарајућу локацију где ће бити снимљени у одговарајућу базу података. Међутим, пре отпремања података у одговарајућу базу података, подаци се снимају и локално, на рачунару који представља извор разматраних података. Ово се практикује из два разлога. Прво, потребно је, пошто се ради о личном рачунару, да корисник може да оствари увид у податке који су прикупљени са рачунара и послати на одговарајући сервер на коме се налази систем за управљање базом података и одговарајућом релационом базом података. И друго, ти локални подаци ће касније у појединим случајевима бити додатно коришћени о чему ће бити речи у неком од наредних поглавља докторске дисертације. Подаци се локално снимају у облику текстуалне датотеке која је дата у CSV формату, па је сходно томе читљива у великом броју апликација које подржавају датотеке овог формата (попут апликација Microsoft Excel, LibreOffice Calc и сличних). За реализацију ових активности

искоришћена је *csv* библиотека [118] са одговарајућим функцијама које омогућавају реализацију снимања података коришћењем CSV формата. Решење је остварено идентично за Windows, Linux и macOS.

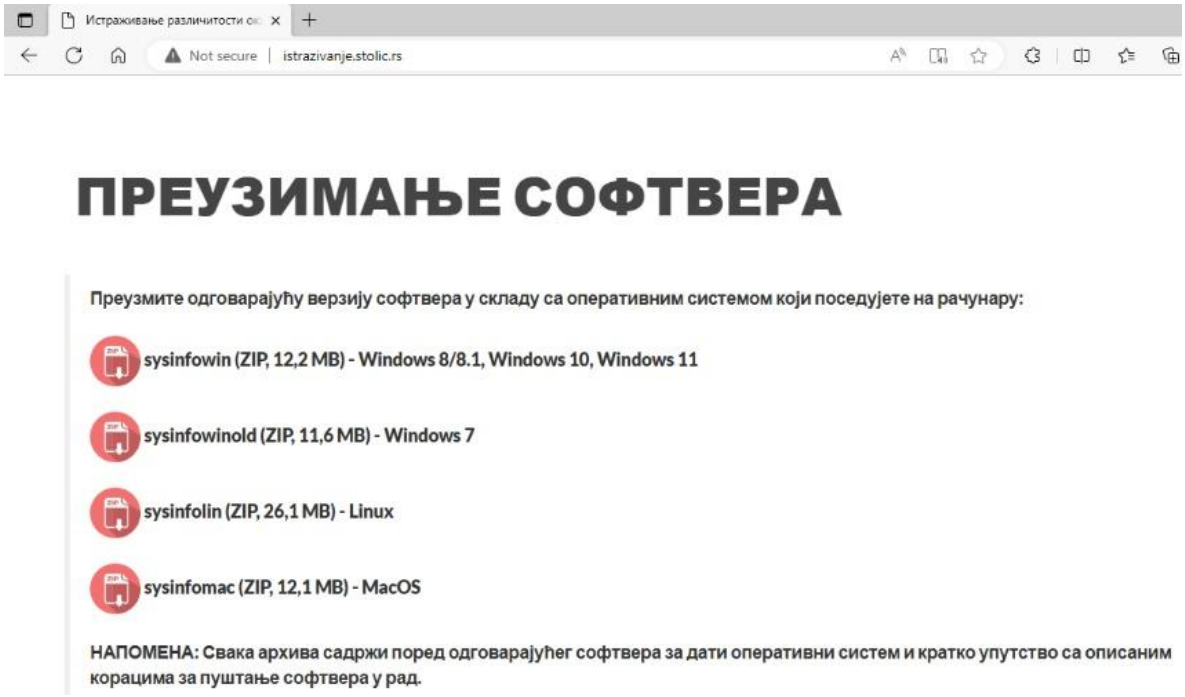
И реализација уписа у одговарајућу базу података се остварује на исти начин за све три платформе и заснива се на употреби званичног *MySQL Connectora* за Python [119], будући да, као што је већ претходно наглашено, постоји компатибилност употребљеног система за управљање релационим базама података са MySQL сервером. Практична реализација заснива се на позивању одређених наредби садржаних у оквиру *mysql.connector* библиотеке. Коришћењем *connect()* конструктора се креира конекција са одговарајућом базом података навођењем следећих параметара: корисничког имена, лозинке, адресе сервера и назива базе података којој се приступа. Овде се могу проследити још неки додатни параметри уколико постоји потреба за истим, међутим у посматраном случају су на серверу ти параметри били подешени на подразумеване вредности, па за додатним навођењем истих није постојала потреба (на пример порт није наведен јер је подешен на подразумевану вредност 3306). Над оствареном конекцијом реализује се одређена структура, познатија као курсор, коришћењем функције *cursor(),* која ће се користити за упис података у одговарајућу базу. Подаци намењени упису у одговарајућу табелу припремају се као непроменљиве листе, а потом се припрема SQL упит који ће реализовати упис у табелу тако што се он придружује одговарајућој текстуалној променљивој. При томе се у упиту на одређени начин означавају тачне позиције на којима ће се уметнути конкретне вредности из напред припремљене непроменљиве листе података који се уписују у табелу. Спремање упита са конкретним вредностима за реализацију уписа извршава се употребом *execute()* функције којој се као параметри прослеђују променљива која садржи SQL упит који је потребно реализовати и назив листе која садржи вредности које треба уписати помоћу поменутог упита. Овај поступак се понавља за све табеле одговарајуће базе података, односно за све упите које је потребно реализовати. Међутим, на овај начин упити су припремљени за реализацију, али конкретана реализација уписа свих података у одговарајућу базу података се неће извршити све до тренутка када се над постојећом конекцијом не позове *commit()* функција. Тек позивом ове функције се сви подаци и физички уписују у базу података. Након извршења свих претходно описаних радњи, затвара се одговарајућа конекција над базом података коришћењем *close()* функције над дефинисаном конекцијом.

Овим је прикупљање података са одговарајућег одредишта завршено и на рачунару са ког је извршено прикупљање података се приказује одговарајућа порука о комплетираним активностима и врши се аутоматско затварање програма. Подаци су снимљени у одговарајућу базу података и могу даље бити коришћени у сврху потребних анализа усмерених ка разумевању посматране хетерогености рачунарског окружења.

Превођење кода у извршне датотеке за све оперативне системе извршено је употребом програма *pyinstaller* [120]. Приликом превођења коришћена је верзија 3.9.17 Python компајлера за Windows 8/8.1, 10, 11, Linux дистрибуције и macOS. Једино одступање у верзији компајлера начињено је у случају превођења за Windows 7, услед старости система и подржаности Python компајлера закључно са верзијом 3.8, па је превођење за овај оперативни систем начињено са старијом верзијом компајлера како би се добио функционалан и стабилан програм који се не мора додатно подешавати и прилагођавати за рад на систему. Након оптимизације процеса компајлирања добијене су извршне датотеке величине око 12 MB за случај Windows 7, 8/8.1, 10, 11 и macOS система, док је за Linux дистрибуције добијена извршна датотека величине око 26 MB. Претпоставља се да се ове величине могу додатно редуковати, али будући да су у последњој итерацији добијене прихватљиве величине датотека и да би даља редукција

захтевала изузетне додатне временске напоре, одлучено је да се прихвати редукција на нивоу на ком јесте тренутно.

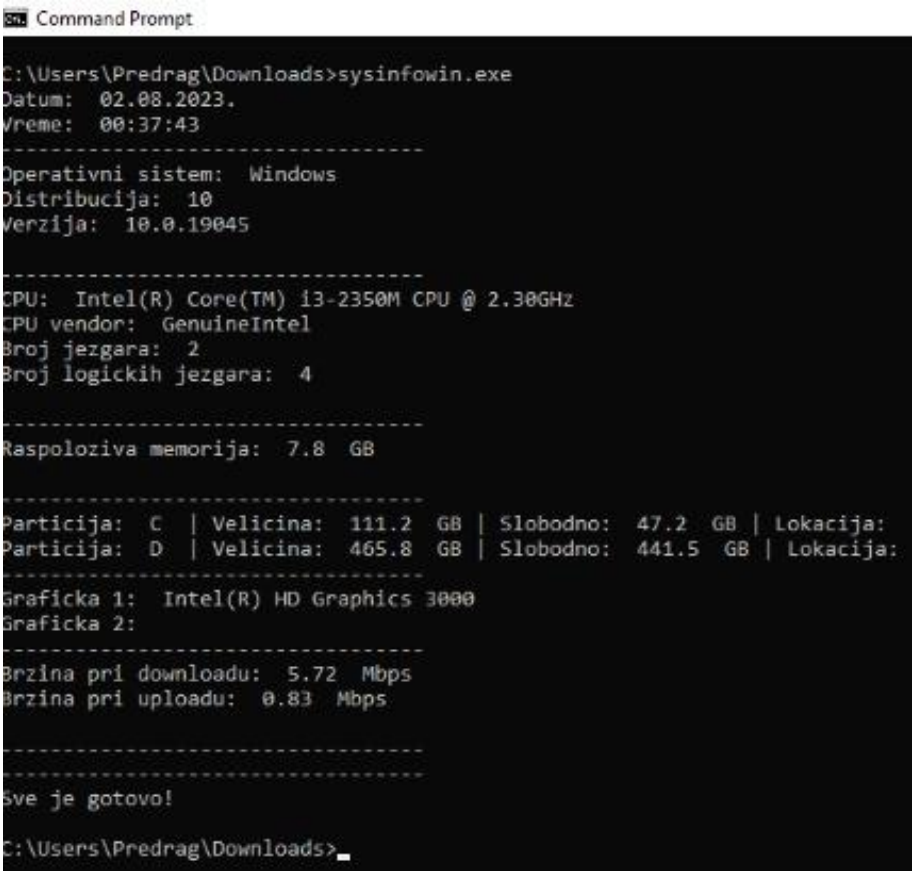
Како би се прикупили реални подаци који ће се искористити касније у даљим анализама, целокупан развијени софтвер, заједно са упутствима, за све подржане оперативне системе обједињен је на једном месту у оквиру сајта istrazivanje.stolic.rs како је приказано на слици 16. Потом су студенти позвани да узму учешће у истраживању тако што ће са предметног сајта преузети адекватан софтвер и покренути га на свом личном рачунару који користе за реализацију наставних процеса. Овакав процес обухватио је као испитанике студенте прве године основних академских студија Техничког факултета у Бору који су слушали предмет Информатика 2 током летњег семестра школске 2022/2023. године. Будући да се студенти на овом предмету деле у две групе које слушају практичну наставу, због једноставности реализовања, формирана су два узорка прикупљених података



Слика 16. Приказ локације за преузимање софтвера за одређени оперативни систем

Поступак је крајње поједностављен. Одласком на поменути сајт преузима се одговарајућа верзија софтвера у зависности од коришћеног оперативног система на рачунару. Потом, пратећи приложена упутства уз сам софтвер, студенти на једноставан начин врше покретање самог софтвера локално, на личном рачунару, а потом софтвер спроводи аутоматски већ поменуте поступке прикупљања података. Софтвер је урађен по минималистичком принципу, без графичког корисничког интерфејса, будући да не постоји никакав вид интеракције са корисником, као што је приказано на слици 17. Корисник у сваком моменту има увид у податке који су прикупљени јер се, у прозору у коме је покренут програм, приказују подаци који се прикупљају редом и у реалном времену. Читав поступак траје врло кратко, у већини случајева завршава се испод једног минута, а у неким случајевима поступак може потрајати и неколико минута (у зависности од старости рачунара, стања хардвера, ресурса којим располаже и слично). Као што је већ претходно поменуто, сви подаци који су послати анонимно и снимљени у одговарајућу базу података, снимају се и локално, у оквиру одговарајуће CSV датотеке под именом local.configuration из које се у сваком моменту може извршити увид у обим

и садржину послатих података. Након успешно реализованог слања података програм аутоматски зауставља свој рад уз адекватну поруку, тако да је, као што је већ више пута напоменуто, интеракција са корисником сведена на минимум.



Слика 17. Илустрација рада софтвера на једном преносном рачунару који је базиран на Windows 10 оперативном систему

2.4. Резултати анализе хетерогености рачунарског окружења

У претходним редовима приказан је развој целокупне софтверске подршке за прикупљање података који ће бити искоришћени приликом анализе хетерогености рачунарског окружења. Циљ те анализе је да потврди постојање изражене хетерогености рачунарског окружења које се остварује у образовном процесу високошколског образовања када се тај процес делимично или у целости спроводи по принципима учења од куће приликом чега студенти користе своје личне рачунарске ресурсе. Како је већ напоменуто приликом прикупљања података формирана су два узорка. Први узорак састављен је од 24 сета података, док је у другом узорку прикупљено 23 сета података, чиме је процес прикупљања података за каснију анализу успешно реализован. Оба узорка ће бити анализирани посебно, а потом ће се извршити и анализа над сједињеним подацима, односно формираће се скуп података састављен од свих сетова података из оба узорка, а потом ће се извршити анализе над овако обједињеним вредностима.

Анализа хетерогености извршиће се употребом основних статистичких показатеља, односно статистичком обрадом добијених података. При томе, у складу са [121], вршиће се прорачун следећих статистичких параметара, заснованих на следећим формулама, уз напомену да се сви чланови узорка сматрају равноправним:

- средња вредност узорка x_s дефинисаће се као

$$x_s = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad (1)$$

при чему је n број чланова у узорку, а x_i вредност i-тог члана узорка,

- на основу претходно изражене средње вредности одступања су исказана као

$$b_i = x_i - x_s \qquad (2)$$

- стандардно одступање узорка је исказано као

$$s = \sqrt{\frac{\sum(x_i - x_s)^2}{n-1}} = \sqrt{\frac{\sum {b_i}^2}{n-1}} \qquad (3)$$

- медијана узорка x_m одређује се по правилу да сума модула одступања од ње има најмању вредност, односно да важи

$$\sum_i |x_i - x_m| = Min \qquad (4)$$

- минимална вредност узорка одређена је као

$$x_{min} = Min(x_i) \qquad (5)$$

- максимална вредност узорка одређена је као

$$x_{max} = Max(x_i) \qquad (6)$$

Поред ових вредности одређује се и број појављивања за сваку од вредности из посматраног узорка.

Овде се мора напоменути да, будући да постоје подаци који се заснивају на измереним вредностима, познато је да таква мерења могу бити подложна одређеној грешци, односно може се јавити одређена мерна несигурност приликом тумачења резултата мерења [122]. Међутим, овде се неће приликом анализе добијених података узимати у обзир мерну несигурност, будући да она не утиче у великој мери на процену хетерогености рачунарског окружења које се посматра у овом случају, а сама идентификација извора мерне несигурности и прорачуни везани за исту могу бити изузетно комплексни и захтевни, па се у складу са наведеним услед смањења опширности и евентуалног губљења фокуса разматрања која се спроводе као што је наведено мерна несигурност неће узимати у обзир овом приликом. Оваква напомена је дата како се не би стекао погрешан утисак о непостојању мерне несигурности и свакако у неким другим разматрањима и истраживањима у овој области треба је узети као врло битан чинилац, али у случају предметне докторске дисертације то није случај.

И за случај анализе прикупљених података је развијен посебан софтвер у програмском језику Python, користећи верзију компајлера 3.9.17 и превођењем написаног кода у извршну датотеку употребом *pyinstaller* софтвера. Како не би

разматрање било преопширно, овде ће се приказати само разматрање везано за прикупљене податке везане за количину меморије, док се поступци за анализу осталих података заснивају на истим принципима. Закључци везани за количину меморије се приказују јер су, између осталог, битни и за неке друге аспекте докторске дисертације о чему ће бити речи у наредним поглављима.

Приступ снимљеним подацима у одговарајућој бази података остварен је поново употребом званичног mysql.connectora који омогућава све фунцкционалности за рад са MySQL и компатибилним релационим базама података у програмском језику Python. И овде ће се у првом кораку формирати конекција над одговарајућом базом на начин који је већ претходно описан коришћењем *mysql.connector* библиотеке. Након успешно реализоване конекције над базом података, формира се упит који ће се реализовати над базом у смислу читања снимљених вредности података из предметне базе како би се приступило даљом анализом истих. Упити могу бити простијег и сложенијег типа у зависности каква врста анализе се спроводи и које податке и из којих табела треба прибавити. У овом случају, будући да се у табели Memorija смештају само вредности расположиве меморије нема потребе примењивати сложен упит, већ ће се прибављање података извршити простим SQL упитом следећег облика:

```
SELECT * FROM sysconf.Memorija
```

где sysconf представља базу података у којој су снимани сви прикупљани подаци, док Memorija представља табелу у којој се налазе вредности расположиве меморије за сваки од рачунара који је анализиран, односно са ког је вршено прикупљање података за анализу.

За статистичку обраду прикупљених података у овом случају коришћена је *pandas* библиотека [123] у програмском језику Python будући да у себи обједињава све потребне механизме за ефикасну, брзу и поуздану статистичку интерпретацију података. Како би подаци били припремљени за спровођење адекватне анализе, потребно је исте, на основу остварене конекције са базом података и горе формираног упита, учитати у одговарајући data frame који се користи у оквиру *pandas* функција за манипулацију подацима будући да представља адекватно решење јер се подаци смештају табеларно и поступци се могу спроводи како над врстама, тако и над колонама. Такође, ова структура омогућава смештај хетерогених података. Како би подаци из базе података били трансформисани у одговарајући data frame употребљена је функција *read_sql()* којој се прослеђују параметри конекције и SQL упит који ће се извршити над датом конекцијом. Реализацијом ове функције одговарајући скуп података преузет је из предметне базе података, извршена његова трансформација без губитака карактеристика података и смештен у одговарајућу променљиву која је по типу data frame.

Све вредности меморије у оквиру базе података су дате као VARCHAR вредности, односно представљене су као текстуалне вредности. Ово је одрађено из простог разлога да би се у базу могао уписати податак као на пример 7.6 GB што недвосмислено указује да је забележена вредност од 7.6 гигабајта расположиве меморије, уместо да се упише бројчани податак 7.6 за који се не би знало у којим је јединицима изражен уколико не постоје додатна појашњења. Већ је речено да се преносом вредности у data frame задржавају карактеристике података на исти начин како су биле представљене у бази података што значи да ће и у оквиру data frame забележене меморијске вредности бити представљене текстуално. Над овако представљеним вредностима не могу се вршити адекватни статистички прорачуни који се заснивају на одређеним аритметичким формулама, па сходно томе треба конвертовати вредности у одговарајући бројчани тип. Ово се врши у два корака. Прво је потребно из податка одстранити све елементе који

нису бројчано представљени. У наведеном примеру у питању је ознака јединице дата као GB. Уклањање јединица је извршено креирањем анонимне lambda функције која се заснива на употреби *rstrip()* функције која врши уклањање са десне стране свих симбола који се прослеђују као параметар функције. Након овог корака и даље постоје текстуални подаци који сада садрже само бројчане вредности. Сада је потребно извршити конверзију текста у број који ће имати вредност записану у оквиру текста. Будући да су у питању бројеви представљени са децималном тачком, конверзија ће се вршити у тип float. Ово се постиже коришћењем функције *astype(float)* над одређеном колоном садржаном у оквиру data frame. Након примене ове функције сада се у дефинисаној колони налазе бројчане float вредности над којим се даље могу вршити одговарајуће статистичке анализе. На овај начин извршена је припрема података за даљу обраду, односно анализу у оквиру реализованог софтвера.

Статистичка карактеризација узорка извршена је позивом одређене функције над data frameом који представља тај узорак. При томе су коришћене следеће функције *pandas* библиотеке:

- *mean()* за израчунавање средње вредности узорка у сагласности са формулом 1,
- *median()* за израчунавање медијане узорка у сагласности са формулом 4,
- *max()* за налажење максималне вредности у узорку у сагласности са формулом 6,
- *min()* за налажење минималне вредности у узорку у сагласности са формулом 5,
- *std()* за налажење вредности стандардног одступања узорка у сагласности са формулом 3 и
- *value_counts()* за налажење броја појављивања свих вредности забележених у посматраном узорку.

Претходним функцијама се на једноставан, ефикасан и поуздан начин извршила статистичка карактеризација узорака. Међутим, у случајевима када узорак обилује мноштвом различитих вредности, преглед резултата добијен функцијом *value_counts()* може бити поприлично непрегледан и тежак за тумачење. Сходно томе, резултате ове функције би требало потпомогнути одговарајућим графичким приказом. За реализацију графичког приказа на овом месту искоришћена је *matplotlib* библиотека [124] чија је функција *pyplot.subplots()* омогућила креирање два графика у оквиру једне радне површине, односно у оквиру једне слике. При томе су као параметри прослеђени распореди приказивања графика (приказивање графика један до другог), као и величина њиховог приказа. За додавање наслова графицима искоришћена је функција *set_title()*. Како се ради о броју појављивања који се жели графички интерпретирати, за графички приказ су одабрани хистограми који су реализовани коришћењем *plot.hist()* функције над одређеним узорком који представља одговарајући data frame. Како би добијени хистограми могли да се тумаче без погрешне интерпретације података забележених на њима, кроз параметре функције за сваки од хистограма дефинисани су називи x и y осе.

У циљу трајног чувања резултата обраде над подацима из узорка, реализована је корисничка функција *DB2XLSX* која омогућава креирање и упис у одговарајућу Excel датотеку коришћењем добро познатог XLSX формата за Microsoft Excel документе. Рад поменуте корисничке функције базиран је на употреби Excel writera садржаног у оквиру *pandas* библиотеке а који се позива коришћењем функције ExcelWriter() при чему се кроз параметре дефинишу Excel датотека у коју ће се вршити упис и покретач дат као *engine="xlsxwriter"*. Сам упис у Excel датотеку реализује се позивом функције *to_excel()* којој се за параметре прослеђују покретач и назив радног листа који се креира у Excel документу у коме се врши упис. Завршетак манипулације над одређеном Excel датотеком реализује се позивом *close()* функције над активним покретачем.

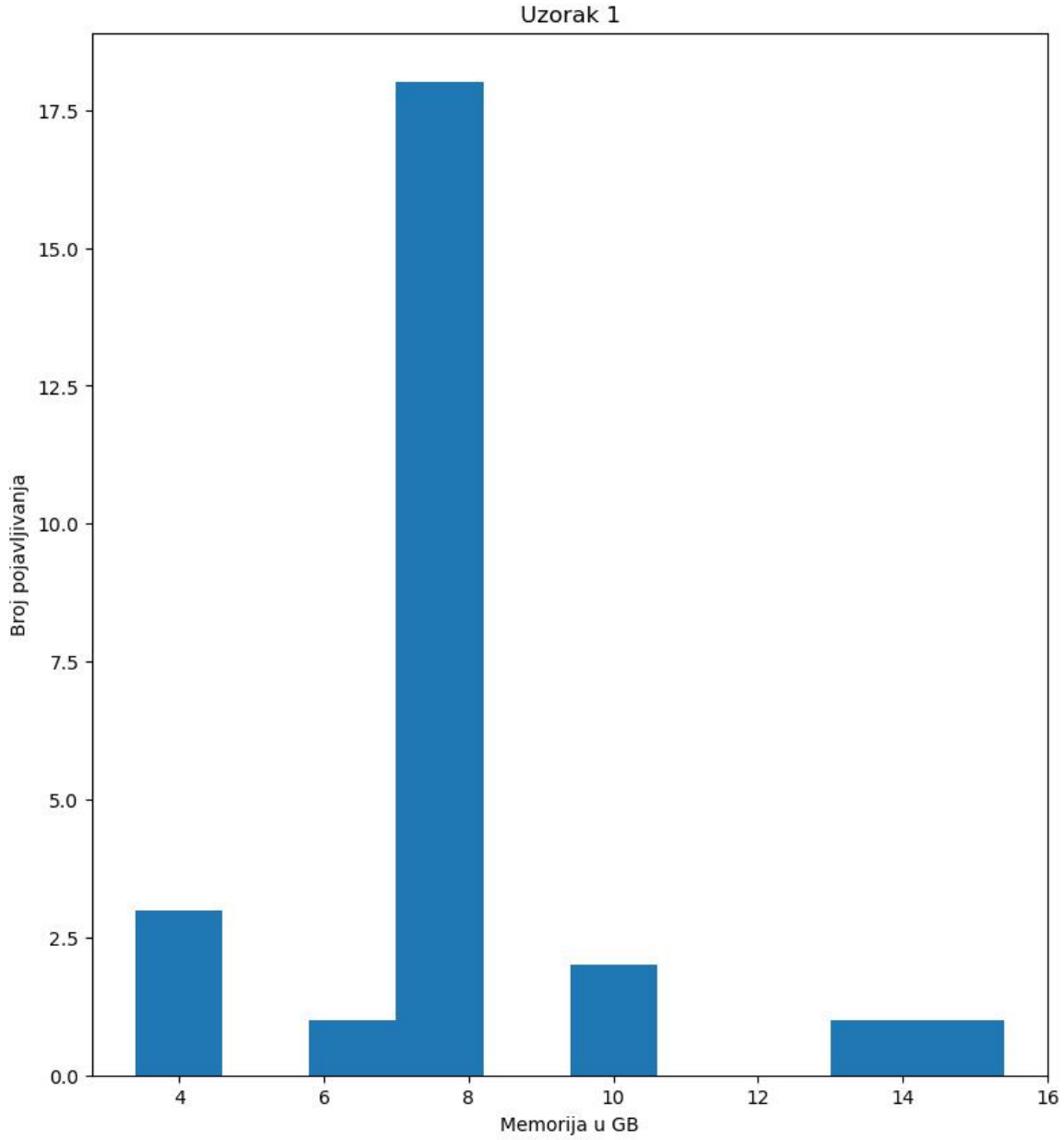
Овим је целокупна статистичка анализа над добијеним подацима завршена и сада се могу реализовати одређени увиди у хетерогеност рачунарског окружења које се анализира на основу реалних показатеља. Резултати су дати у оквиру табела 2 и 3 и слике 18 за први узорак, табела 4 и 5 и слике 19 за други узорак и табела 6 и 7 и слике 20 за обједињене вредности у оквиру једног data framea.

Табела 2. Статистички показатељи за први узорак

Ознака	**Вредност у GB**
x_s	7.93846153846154
x_m	7.8
x_{max}	15.4
x_{min}	3.4
s	2.4713865499532734

Табела 3. Број појављивања података за први узорак

Вредност	**Број појављивања**
7.8	10
7.9	3
3.4	2
7.7	2
9.9	2
7.6	1
6.9	1
3.9	1
13.9	1
8.0	1
7.0	1
15.4	1



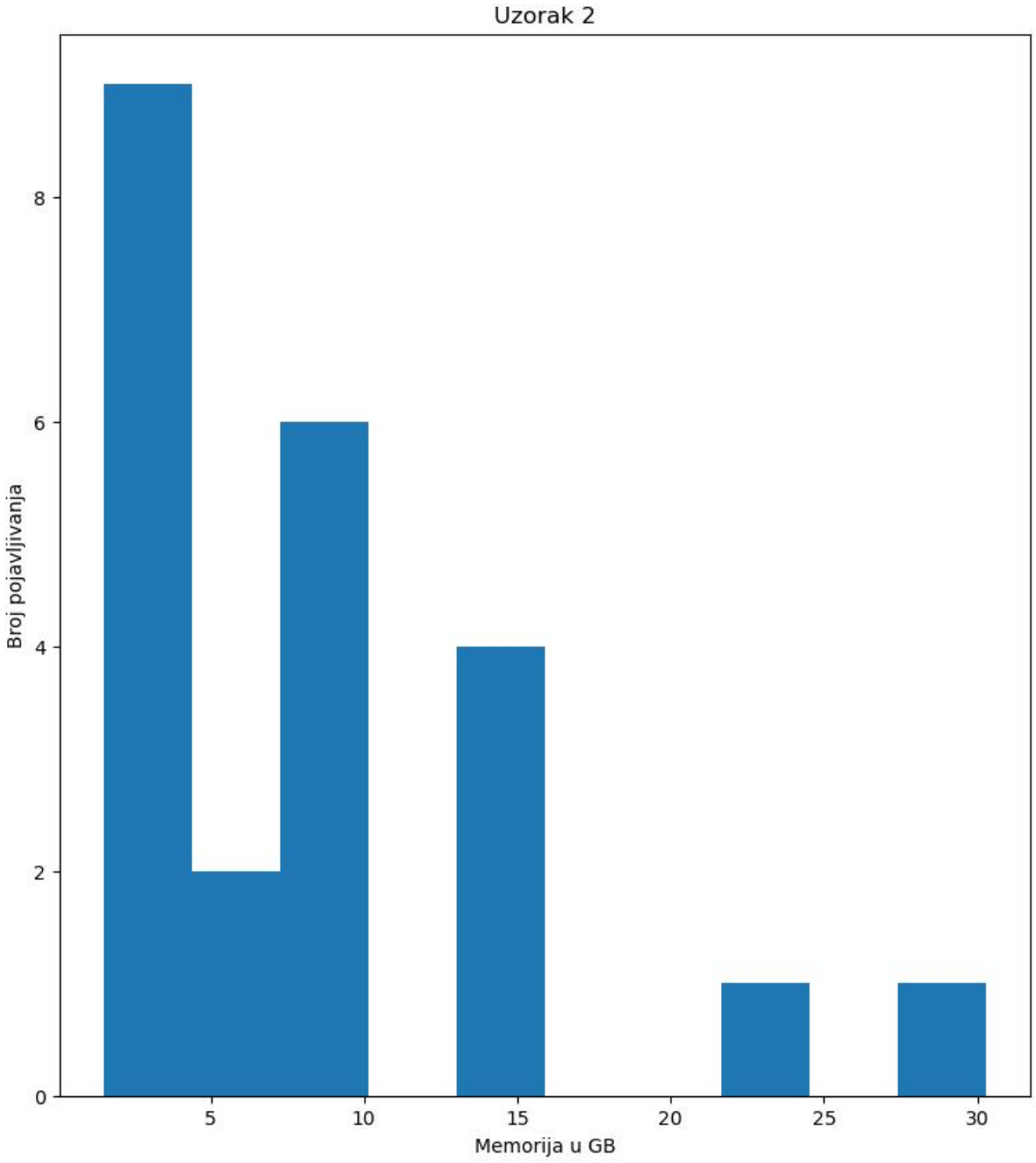
Слика 18. Реализовани хистограм за први узорак

Табела 4. Статистички показатељи за други узорак

Ознака	**Вредност у GB**
x_s	8.465217391304348
x_m	7.6
x_{max}	30.3
x_{min}	1.5
s	7.0313959709695695

Табела 5. Број појављивања података за други узорак

Вредност	**Број појављивања**
7.9	2
7.8	2
14.6	2
2.0	2
3.0	2
7.6	2
4.0	2
22.4	1
5.8	1
15.4	1
13.9	1
1.9	1
3.8	1
1.5	1
30.3	1
5.9	1



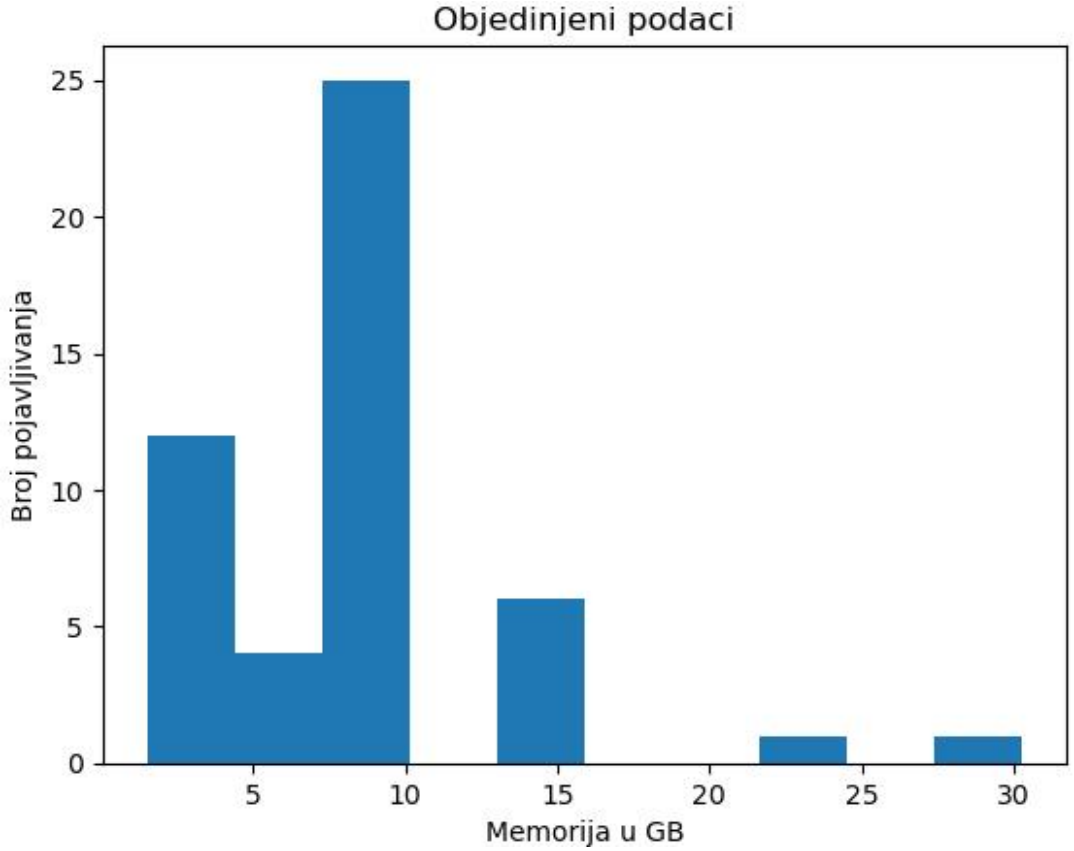
Слика 19. Реализовани хистограм за други узорак

Табела 6. Статистички показатељи за обједињене податке

Ознака	**Вредност у GB**
x_s	8.185714285714285
x_m	7.8
x_{max}	30.3
x_{min}	1.5
s	5.149440231120034

Табела 7. Број појављивања за обједињене податке

Вредност	**Број појављивања**
7.8	12
7.9	5
7.6	3
3.0	2
2.0	2
14.6	2
9.9	2
15.4	2
13.9	2
7.7	2
3.4	2
4.0	2
8.0	1
7.0	1
3.9	1
22.4	1
5.8	1
6.9	1
1.9	1
3.8	1
1.5	1
30.3	1
5.9	1



Слика 20. Реализовани хистограм за обједињене податке

Из приложених анализа прикупљених података може се уочити неколико кључних ствари. Ако се анализирају минималне и максималне вредности расположиве меморије, примећује се велика разлика између ових вредности. Ако би дефинисали ту разлику као:

$$x_{range} = x_{max} - x_{min} \qquad (7)$$

добија се за обједињене вредности износ од чак 28.8 GB што није уопште занемарљива разлика која указује да постоје корисници чији системи имају изузетну малу количину расположиве меморије насупрот корисницима код којих је забележена изузетна количина расположиве радне меморије. Овом тврђењу иде у прилог и чињеница да су забележене врло велике вредности стандардног одступања узорка, које за обједињене вредности има вредност од приближно 5.15 GB, што представља изузетно високу вредност стандардног одступања, која указује на расподелу вредности у врло широком опсегу у односу на средњу вредност која за обједињене вредности износи приближно 8.19 GB. Такође, на основу идентификованих бројева понављања вредности расположиве меморије, види се да постоји врло велика различитост у погледу забележених вредности меморије. Сходно свему наведеном може се извести закључак да постоји изражена хетерогеност испитиваног рачунарског окружења на основу забележених стања расположиве количине радне меморије. Сличне анализе и закључци се могу извести и анализом осталих параметара који су снимани са изворишних рачунара, али као што је већ напред поменуто, због обимности разматрања они се неће посебно анализирати овом приликом.

Поставља се питање зашто је потребно спроводити овакве анализе и обратити пажњу на хетерогеност рачунарског окружења када се ради о спровођењу наставног процеса у високошколском образовању које подразумева и рад од куће. Неопходност лежи у

чињеници да је потребно за сваког студента обезбедити једнаке услове рада како би сваком студенту била пружена једнака шанса за позитивну реализацију исхода учења. Међутим, овде постоји усложњавање реализације претходног, будући да не постоје адекватна сазнања о инфраструктурним основама код студената, односно не постоје адекватна сазнања о параметрима ресурса које ће студенти користити. У већим групама студената немогуће је посветити пажњу сваком студенту појединачно и спровести појединачна испитивања у погледу ресурса којим свако од студената располаже. Зато је важно направити једну овакву анализу и на основу ње спровести одређено дејство у формирању закључака који софтвер ће у настави бити коришћен, која верзија, како, на који начин, како би исти могао радити на сваком рачунару код сваког студента готово на идентичан начин. При томе, као што се види из анализа, треба покрити један широк опсег, што у појединим ситуацијама представља исцрпљујућу и комплексну активност, а у појединим случајевима ту активност је готово и немогуће спровести у целости.

Зато се у наставку ове докторске дисертације предлажу нека нова нетрадиционална решења усмерена управо ка овом циљу, покрити широк дијапазон корисника, омогућити готово идентична окружења уз врло мали степен интеракције са крајњим корисником и при томе омогућити да претходно поменуто буде реализовано без обзира на познавање ресурса који се налазе код крајњих корисника.

3. СОФТВЕРСКЕ ОСНОВЕ ПРЕДЛОЖЕНОГ РЕШЕЊА

3.1. Уводна разматрања

У претходном поглављу примењена је анкета над реалним испитаницима. Прва група испитаника обухватила је студенте прве године основних академских студија Техничког факултета у Бору Универзитета у Београду, док је други узорак сачињен од студената четврте године основних академских студија студијског програма Рударско инжењерство који се реализује, такође, на Техничком факултету у Бору Универзитета у Београду. Том приликом анализирани су резултати одговора на првих седам питања у одговарајућој анкети, док ће се сада, за потребе даљих разматрања, анализирати додатних пет питања из поменуте анкете, а која су приказана у табели 8.

Табела 8. Садржај питања и понуђених одговора питања 8-12 из реализоване анкете

Редни број питања	**Садржај питања и понуђених одговора**
8.	**ПИТАЊЕ:** Оцените параметре интернет везе коју користите приликом реализације свакодневних задатака оценом 1 до 5, при чему је оцена 1 најнижа, а оцена 5 највиша оцена.
	ОДГОВОР: Стабилност интернет везе: 1 2 3 4 5 Брзина интернет везе: 1 2 3 4 5
9.	**ПИТАЊЕ:** Да ли је тренутно на личном рачунару инсталиран идентичан софтвер (тип софтвера, верзија софтвера, надоградње и слично) оном који се користи на рачунарским вежбама у рачунарским лабораторијама.
	ОДГОВОР: 1) Да ; 2) Не
10.	**ПИТАЊЕ:** Да ли сматрате да је од врло велике важности за успешно савладавање наставног градива да софтвер инсталиран на личном рачунару буде идентичан софтверу који користите у рачунарској лабораторији.
	ОДГОВОР: није од велике важности ; Б) делимично је важно ; В) изузетно је важно
11.	**ПИТАЊЕ:** Да ли сте упознати са концептом виртуелизације?
	ОДГОВОР: 1) Да ; 2) Не
12.	**ПИТАЊЕ:** У погледу знања везано за одабир, инсталацију и конфигурисање софтвера, сматрам да:
	ОДГОВОР: 1) немам потребна знања, други то раде за мене ; 2) имам основна знања о одабиру и инсталацији, али немам знања о конфигурацији ; 3) имам основна знања о одабиру, инсталацији и конфигурисању ; 4) поседујем напредна знања о одабиру, инсталацији и конфигурисању

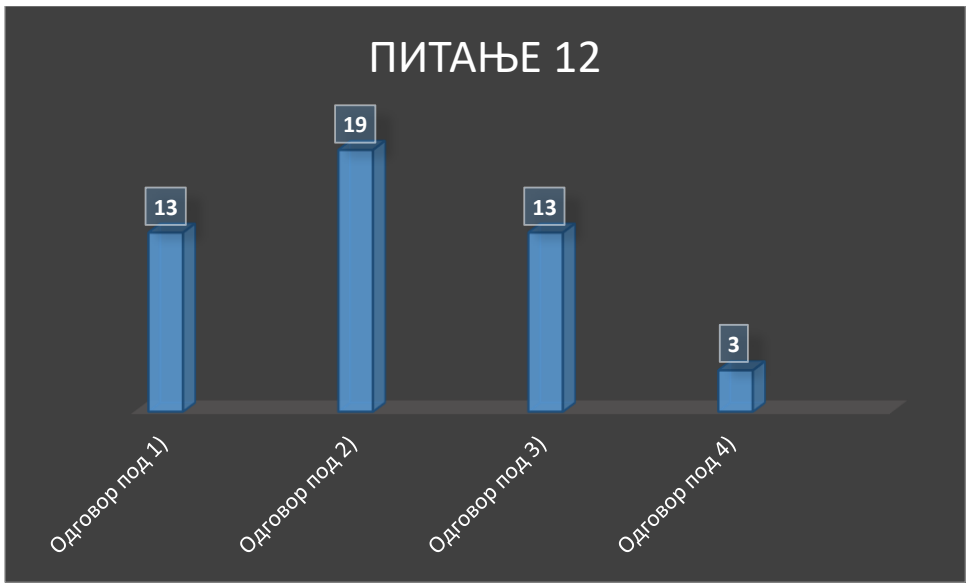
Како је већ у претходном поглављу напоменуто, први узорак састављен је од одговора који су пружени од стране 48 анкетираних студената прве године основних академских студија Техничког факултета у Бору који су годину уписали у школској 2022/2023. години. Структура одговора на понуђена питања у оквиру поменуте анкете приказана је одговарајућим графиконима приказаних на сликама од 21 до 25.

Слика 21. Број забележених одговора у оквиру осмог питања из анкете - први узорак

Слика 22. Број забележених одговора у оквиру деветог питања из анкете - први узорак

Слика 23. Број забележених одговора у оквиру десетог питања из анкете - први узорак

Слика 24. Број забележених одговора у оквиру једанаестог питања из анкете - први узорак



Слика 25. Број забележених одговора у оквиру дванаестог питања из анкете - први узорак

Сходно изнетом у претходном поглављу, други узорак састављен је од одговора који су пружени од стране 17 анкетираних студената четврте године основних академских студија студијског програма Рударско инжењерство Техничког факултета у Бору који су

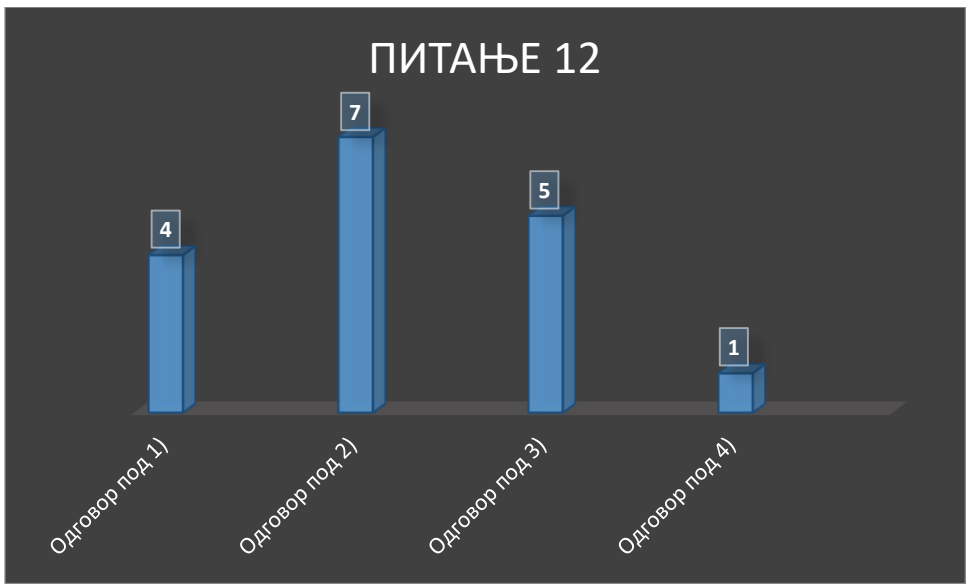
годину уписали у школској 2022/2023. години. Структура одговора на понуђена питања у оквиру поменуте анкете приказана је одговарајућим графиконима приказаних на сликама од 26 до 30.

Слика 26. Број забележених одговора у оквиру осмог питања из анкете - други узорак

Слика 27. Број забележених одговора у оквиру деветог питања из анкете - други узорак

Слика 28. Број забележених одговора у оквиру десетог питања из анкете - други узорак

Слика 29. Број забележених одговора у оквиру једанаестог питања из анкете - други узорак



Слика 30. Број забележених одговора у оквиру дванаестог питања из анкете - други узорак

Анкетом се желело и да се направи увид у стање на личним рачунарим студената по питању софтвера који се користи за реализовање наставних процеса, као и да се виде каква су мишљења студената по разним питањима коришћења и инсталације софтвера. Управо зато девето питање у анкети намењено је утврђивању поклапања софтвера инсталираном код куће, на личним ресурсима, са софтвером који се регуларно користи у оквиру рачунарских кабинета у оквиру високошколске институције. У оквиру првог узорка око 56% испитаника одговорило је да не постоји поклапање софтвера (слика 22), док је у другом узорку тај проценат порастао на око 88% испитаника (слика 27). Уочава се велико непоклапање софтвера који је инсталиран на личним ресурсима са оним који је инсталиран на институционалним ресурсима, што може отежати извођење наставног процеса и унети неке нове непознанице у остварење коначних исхода учења. На пример могу се јавити проблеми о непостојању одређених опција у софтверу који је инсталиран код куће, у односу на онај који је инсталиран на факултету, или опције могу бити другачије размештене, у оквиру неких других алата, менија и слично. Могу се јавити и проблеми са слањем урађених задатака, где се може дестити ситуација да испитивач не може отворити задатак испитаника, или га отвара неадекватно са грешкама, услед различитих верзија софтвера, некомпатибилности и сличног. Све ово може имати велике утицаје на сам наставни процес који се може поприлично отежано одвијати, а у појединим случајевима могу се јавити и погрешне интерпретације по питању нивоа достигнутог знања кандидата, његовог залагања, остварених резултата што у крајњој мери може довести и до погрешног оцењивања кандидата.

Како би било анализирано мишљење студената о важности поклапања софтвера који је инсталиран на личним ресурсима са оним на ресурсима у оквиру високошколске институције, без обзира на тренутно стање по питању инсталираног софтвера, испитаницима је постављено питање о важности поклапања софтвера. Испитаници су могли да одговоре да наведено поклапање није од велике важности, да је делимично важно, или да је од изузетне важности. Како је приказано на слици 23, у првом узорку за скоро 65% испитаника поклапање софтвера није од велике важности, или је од делимичне важности. У другом узорку, приказаном на слици 28, нико од испитаника није тврдио да поклапање софтвера није од велике важности, док се о делимичној важности поклапања софтвера изјаснило 41% испитаника. У складу са добијеним резултатима примећује се да не постоји довољно развијена свест код студената о значају постојања идентичног софтвера на личним ресурсима у односу на онај који је инсталиран на институционалним ресурсима. Испитаници немају довољну информисаност о последицама које могу настати коришћењем различитих верзија, или у крајњој мери потпуно различитог софтвера, а које могу имати велике последице на адекватни трансфер знања, испуњеност предиспитних обавеза и у крајњој мери на позитиван исход остваривања исхода учења.

У погледу инсталације софтвера, кроз анкету се желело да се дође и до неког показатеља о нивоу техничких знања које испитаници поседују у погледу одабира, инсталације и конфигурисања софтвера на личним ресурсима. Тако се, у складу са резултатима приказаним на слици 25, у првом узорку 54% испитаника изјаснило да поседује комплетна знања о одабиру, инсталацији и конфигурисању софтвера, било да се ради о основним или напредним знањима, док је у другом узорку, приказаном на слици 30, тај резултат око 35% испитаника. Приложеним резултатима поткрепљује се чињеница да у погледу инсталације софтвера на личним ресурсима студената могу постојати велике препреке у погледу адекватног одабира софтвера, правилног поступка инсталације, као и његовог конфигурисања, будући да постоји велики број испитаника који указују на то да не поседују довољну количину знања у целости, или постоје одређени недостаци у знању. Код оваквих испитаника постоји велика могућност да ће се

испољити одређени проблеми везани за софтвер који ће утицати и на саму реализацију наставног процеса, односно постоји велика вероватноћа да ће доћи до пада успешности по питању остварења крајњих исхода учења.

Поред наведених, треба у обзир узети још један релевантан фактор када је у питању учење од куће, а то је сам интернет без кога овакав вид наставног процеса не би био изводљив. Брзина и стабилност интернет везе су у многоме напредовали када је у питању приступ интернету намењењен физичким лицима. Време dial-up конекција је превазиђено, конекције засноване на приступу интернета преко телекомуникационе структуре засноване на бакарним парицама, какав је на пример ADSL приступ, су напредовале до изузетних брзина у односу на деценију-две уназад, а приступ интернету коришћењем оптичке инфраструктуре са драстично увећаним брзинама полако постаје стандард за многе. Међутим, кроз анкету се желело да се добије непосредан одговор од испитаника, па је сходно томе испитаницима постављено питање да оцене стабилност и брзину интернет везе које тренутно користе оценама од 1 до 5. Са слике 21 види се да су испитаници из првог узорка већински задовољни тренутним стањем по питању интернет конекције, будући да је скоро 90% испитаника оценило стабилност оценом 3 и већом, односно око 81% испитаника оценило је брзину оценом 3 и већом. Резултати су још бољи кад се сагледа оцењивање у другом узорку приказано на слици 26 где је само један испитаник дао оцену 2 за стабилност, док су оцене свих других испитаника биле једнаке или изнад 3, док су у случају брзине сви испитаници оценили оценом једнаком или изнад 3. Овакво стање је изузетно добро за развој наставе која укључује делимично или у целости учење код куће, али овакво стање може и заварати ако се у потпуности ослонимо да ће квалитет интернет везе увек бити добар, како у погледу стабилности, тако и у погледу брзине.

Све напред наведено указује на то да постоје велике потешкоће по питању адекватне инсталације неопходног софтвера за реализацију наставног процеса на личним ресурсима студената када се о том процесу старају сами студенти без обзира на начин којим се исти реализује. Сходно томе треба остварити одређене поступке и механизме који би омогућили да се потенцијални проблеми на које је претходно указано минимизују, односно да се нађе одређени вид аутоматизације целокупног процеса који би учешће студената свео на минимум и који би, такође, умањио и могућност да дође до грешака у остваривању процеса инсталације неопходних софтвера везаних за учење. При томе решење треба реализовати тако да оно буде изводљиво чак и у случајевима када квалитет интернет везе драстично опадне, без обзира на изузетне резултате постигнуте у реализованој анкети. На пример може доћи до тренутне недоступности одређене инфраструктуре па се интернет саобраћај може преусмерити на алтернативне руте које су значајно мањих капацитета па долази до пада перформанси. Такође, искуства током трајања COVID-19 пандемије су показала да чак и код јако добро пројектованих система са значајном количином резерве и добром димензионисаношћу може доћи до појава одређених пикова у саобраћају где долази до значајнијих, а у појединим моментима и јако критичних оптерећења.

У наставку докторске дисертације биће разматрани овакви механизми који се заснивају на примени виртуелних машина приликом реализације наставног процеса и исхода учења. Будући да се жели максимум доступности наставног процеса, овде ће се, без обзира на постојање јако добрих интернет конекција у већини случајева, изградња целокупног решења усмерити ка једном минималистичком приступу како би он био одржив и у случајевима значајнијих деградација перформанси на мрежи. Студенти поседују различите нивое знања везано за концепте виртуелизације. То показује и спроведена анкета где је око 46% испитаника у првом узорку било упознато са концептима виртуелизације (слика 24), док у другом узорку није било испитаника за које

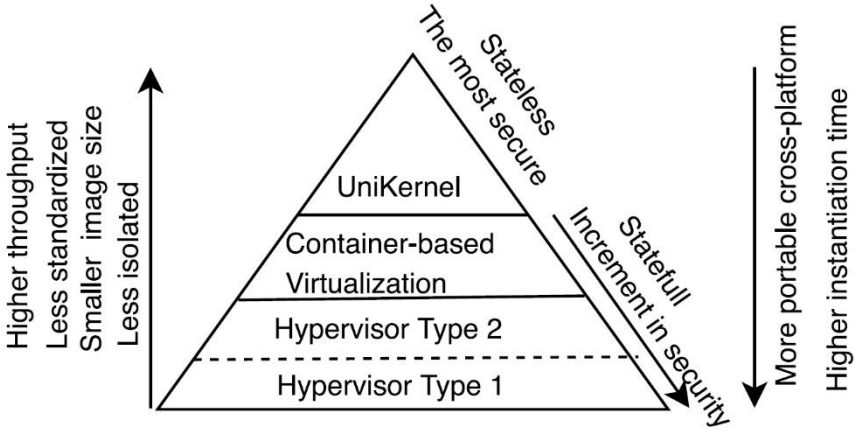
је концепт виртуелизације представљао непознаницу (слика 29). Сходно томе не може се очекивати од свих студената да знају сами да реализују све задатке везано за примену виртуелизације, а будући да се трага за једним универзалним решењем које треба бити једнако доступно за све студенте, опет долазимо до закључка да сваку даљу интеракцију са студентима треба избећи када год је то могуће, односно постићи највећи могући степен аутоматизације свих процеса. Ово је значајно учинити и из разлога што сама употреба виртуелних машина у наставним процесима високошколског образовања зна да унесе неке додатне комплексности у реализацију наставног процеса, те је потребно те потенцијалне комплексности на време уочити и дати адекватан одговор на исте.

3.2. Коришћење виртуелних машина у домену високог образовања

Техничка инфраструктура у обезбеђивању наставног процеса, који се делимично или у целости ослања на принципе учења од куће, представља један од главних фактора успешности реализације самог процеса. Ово су приметили и аутори у [54] који тврде да од добре реализације инфраструктуре зависи и успех или неуспех образовања на даљину. Када је у питању успостављање оваквог система учења где се задаци обављају учењем код куће, инфраструктурни проблем има још једну димензију сложености, будући да постоји коришћење личних рачунарских ресурса студената, који могу међусобно доста варирати једни у односу на друге у погледу хардвера. Ако посматрамо реализацију наставног процеса у оквиру високошколске образовне установе, за рачунарске лабораторије, у којима се изводи практична настава на оним предметима који захтевају коришћење рачунара, постоји адекватно познавање хардверских, будући да су предметне лабораторије формиране по одговарајућим принципима, односно у складу са одговарајућим техничким захтевима и спецификацијама. Можемо рећи да свака рачунарска лабораторија представља добро познато хомогено рачунарско окружење за које постоје адекватна знања не само о компонентама од којих се сама лабораторија састоји, већ се зна и може предвидети тачно понашање сваког појединачног рачунара који се налази у оквиру посматране лабораторије. Супротно томе, коришћењем личних рачунарских ресурса у наставном процесу који је заснован на принципима учења од куће, постоји високо хетерогено рачунарско окружење са много непознаница у погледу коришћеног рачунарског хардвера и његовог понашања током реализације различитих задатака у оквиру спровођења циљева наставне активности. Како поменуто хетерогено рачунарско окружење не би представљало додатни извор проблема у реализацији наставног процеса, потребно је пронаћи одговарајући начин за превазилажење додатне сложености коју поменута хетерогеност уноси у наставни процес.

Претходно поменута комплексност која се појављује услед високо изражене хетерогености може се превазићи применом адекватних решења који се могу пронаћи у оквиру концепата виртуелизације. Коришћењем процеса виртуелизације могуће је креирати одговарајући слој апстракције који омогућава стварање виртуелног хардвера преко стварно присутног, физичког хардвера у личним рачунарима студената [67,70], па се може путем виртуелизације са одговарајућом апстракцијом извршити трансформација уочене хетерогености у неку врсту хомогеног окружења.

Овде треба поменути да постоји више различитих техника виртуелизације које су данас у употреби у различитим доменима примене, а графичко поређење неких њихових карактеристика приказано је на слици 31, преузетој у оригиналном формату из [64].

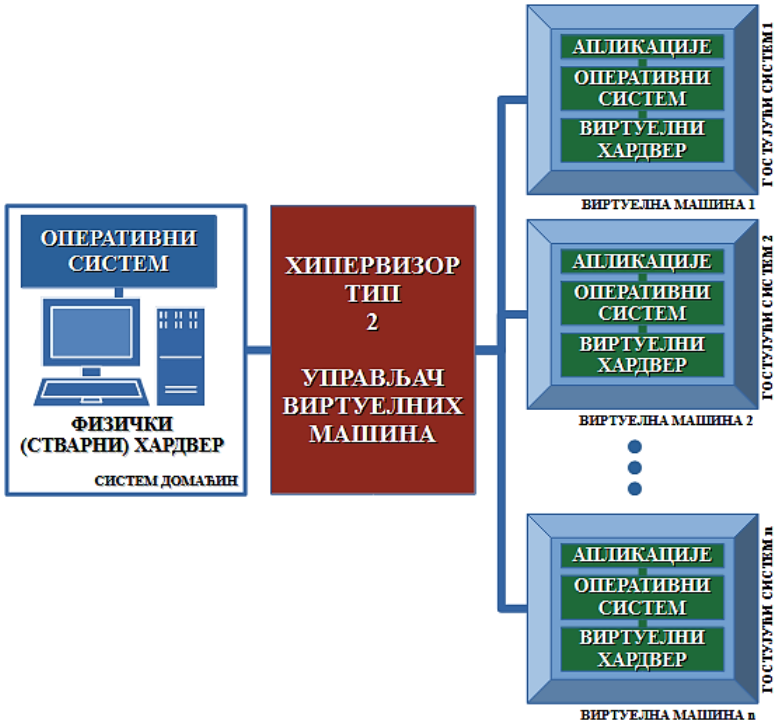


Слика 31. Приказ карактеристика различитих виртуелизационих техника [64]

У предметној докторској дисертацији користиће се виртуелизациона решења заснована на имплементацији коришћењем хипервизора типа 2 који се у литератури често означава и као управљач (менаџер) виртуелних машина (Virtual Machine Manager – VMM). Оваква реализација хипервизора представља специјализовани софтвер који се инсталира у оквиру постојећег оперативног система рачунара у циљу омогућавања имплементације и управљања виртуелним машинама (Virtual Machine – VM) заснованих на одговарајућем виртуелном хардверу, минимално једне, а највише онолико колико сами физички ресурси и спецификације виртуелног хардвера то дозвољавају. Свака виртуелна машина заснована је на коришћењу одговарајућег оперативног система унутар саме виртуелне машине, а сама виртуелна машина испољава одређене карактеристике једног изолованог окружења. Та изолованост практично значи да, у највећем броју случајева, акције спроведене у оквиру виртуелне машине неће имати директне последице по стварни хардвер рачунара, нити по сам оперативни систем рачунара у оквиру којег је инсталиран поменути хипервизор. Овде треба напоменути да се често, како би си правила разлика између стварног рачунара и његовог оперативног система и виртуелне машине и њеног оперативног система, стварни рачунар заједно са његовим софтвером означава као систем домаћин (host system), док се виртуелна машина заједно са својим софтвером означава као гостујући систем (guest system).

Приликом реализације виртуелне машине, засноване на горе описаним принципима, не постоји потреба за коришћењем посебних, наменских, оперативних система, већ се користе стандардни оперативни системи који се користе и приликом инсталације на стварним рачунарима, односно на стварном хардверу. У оквиру виртуелне машине, у зависности од коришћеног виртуелног хардвера и његове конфигурације, могу се инсталирати готово сви оперативни системи који су данас у употреби на рачунарима: Windows, Linux, macOS и други. Захваљујући оваквом приступу у реализацији виртуелне машине, постоји могућност да се у оквиру реализованог виртуелног окружења у највећем броју случајева користе апликације на исти начин као што би биле коришћене у реалном окружењу, односно на стварном хардверу рачунара. Оваква карактеристика виртуелних машина чини их подесним кандидатима за употребу у реализацији наставних процеса у оквиру спровођења програма високошколског образовања, будући да су вештине и знања стечене приликом рада у оквиру апликација реалног система, услед идентичности, употребљиве и приликом рада у оквиру виртуелног окружења, односно употреба истих апликација у виртуелним окружењима неће захтевати стицање нових знања и вештина што у многоме може допринети позитивном остварењу крајњих исхода учења.

Илустративни приказ основних слојева архитектуре претходно разматраног концепта виртуелизације, а који ће бити употребљен у оквиру докторске дисертације, дат је на слици 32.



Слика 32. Основни слојеви виртуелизације коришћењем хипервизора тип 2

Претходно описана архитектура којом се сама виртуелизација спроводи кроз креирање виртуелних машина постоји већ доста дуго и не представља новину у рачунарском свету, али, њена примена у домену високошколског образовања је и даље доста ограничена без обзира на низ погодности које нуди. Један од кључних фактора за и даље поприлично лимитирајућу примену виртуелних машина у наставним процесима који се спроводе у високошколском образовању јесте чињеница да је тренутним стањем у погледу примене виртуелних машина студент стављен у такву позицију да већину одлука у погледу виртуелизације мора донети самостално и да мора значајно време одвојити за разне активности у погледу исте и тиме посветити значајно време активностима које нису директно везане за сам процес учења и реализацију предвиђених наставних активности, већ су везане за припрему виртуелне машине за реализацију наставних процеса. У хардверском смислу, проблеми хетерогености се значајно превазилазе коришћењем виртуелног над стварним хардвером и постиже се довољан привид униформности у употреби код свих актера предвиђених наставних процеса. Међутим, ако се решење посматра кроз један целовит приступ, хардверски домен не може се стриктно одвојити од софтверског домена, а без обзира на примену одређене технике виртуелизације, одређени проблеми и даље коегзистирају у софтверском домену примене.

Како је већ претходно указано, све активности везане за инсталацију и конфигурацију потребног софтвера у оквиру виртуелне машине, која ће се користити у сврху реализације учења од куће, обављаће студент, што представља један од кључних проблема који се могу јавити приликом имплементације оваквог вида виртуелизације у високошколском образовању. На овај начин постоји висока индивидиуализација самог процеса, будући да ће целокупан процес бити спровођен од стране сваког студента појединачно (принцип сваки студент сам за себе), па ће долазити у мањој или већој мери до разноликости у тумачењу поступака, њиховој реализацији, избору софтвера и сличних активности. Наведено доводи до одступања од одговарајуће униформности самог процеса, а самим тим враћа се на почетак решавања постављених задатака, односно врши се удаљавање од хомогених решења и појачава се поново степен хетерогености у

посматраном рачунарском окружењу. Поред овог, треба указати и на чињеницу да немају сви студенти довољна техничка знања из потребних области које се овом приликом захтевају, неки поседују минимална знања, неки поседују парцијална знања, а постоји и онда група студената чија знања нису адекватна, често и врло погрешна.

Постоје многа мишљења да претходно изречено апсолутно не треба узимати у разматрање, односно да треба искључиво разматрати централизована решења где је целокупни развој виртуелних машина искључиво у надлежности високошколске установе, а потом се виртуелна машина предаје на даљу употребу студенту. Међутим, ни обезбеђивање виртуелних машина од стране техничког особља високошколске установе не би било адекватно решење услед постојања два неадекватна сценарија развоја виртуелних машина који не би дали одговоре на постојеће проблеме без увођења неких нових додатних комплексности у целокупно решење. По првом сценарију извршила би се имплементација само једне виртуелне машине за целокупну високошколску установу која би покривала све нивое студија, све студијске програме и све предмете, или би се евентуално радило о виртуелним машинама које би покривале све студијске програме и све предмете, али појединачно за сваки ниво студија. Таква виртуелна машина би била преобимна, ресурсно захтевна, садржала би све софтвере који се користе током школовања у оквиру одређене високошколске установе, од којих велика већина није неопходна студенту (није похађао све курсеве, није укључен у све студијске програме и слично) и на крају би таква виртуелна машина услед поменутих фактора била веома збуњујућа и компликована за употребу од стране студената као крајњих корисника.

Други сценарио подразумева имплементацију појединачних виртуелних машина за сваки модул и сваку годину студија која се реализује у оквиру високошколске установе. То практично значи стварање великог броја виртуелних машина (на десетине виртуелних машина), где је сада доминантно додатно оптерећење техничког особља и ресурса саме високошколске установе, па ни овај сценарио не представља препоручљив концепт за редовну имплементацију у високошколским образовним процесима. Овај случај размотриће се на примеру Техничког факултета у Бору Универзитета у Београду. У оквиру ове високошколске установе настава се организује на свим нивоима студија. На основним академским студијама настава се реализује на четири студијска програма који укупно поседују девет модула. Будући да су у питању четворогодишње студије, долази се до укупно 9·4=36 виртуелних машина које се морају реализовати за потребе извођења наставе на основним академским студијама. На мастер академским студијама настава се реализује на четири студијска програма који укупно поседују шест модула. Будући да су у питању једногодишње студије, долази се до укупно 6·1=6 виртуелних машина које се морају реализовати за потребе извођења наставе на мастер академским студијама. На докторским академским студијама настава се реализује на четири студијска програма који укупно поседују четири модула. Будући да су у питању трогодишње студије, долази се до укупно 4·3=12 виртуелних машина које се морају реализовати за потребе извођења наставе на докторским академским студијама. То значи да ће на нивоу ове високошколске институције бити 36+6+12=54 реализованих виртуелних машина како би се подмириле потребе високошколске институције за све студијске програме на свим годинама и свим нивоима студија. Овај број може бити и већи уколико се додатно изврши усложњавање критеријума по коме се врши формирање виртуелних машина у зависности од процеса спровођења образовног процеса. На пример, могуће је да се једна виртуелна машина „подели“ на две виртуелне машине услед потребе да свака садржи софтвер за одређену групу предмета и слично. Чак и у најбољим случајевима оптимизације броја виртуелних машина које би се користиле по овом сценарију њихов број и даље остаје врло велики, а остају присутни и наведени проблеми у вези захтевности у погледу коришћења ресурса и слично. Поред тога, овде би, вероватно,

дошло до испољавања и још неких додатних појава, попут евентуалног предимензионисања система у појединим моментима услед обезбеђења довољне количине ресурса, повећање степена сложености инфраструктуре, додатних економских напора по саму високошколску институцију и осталих.

Вероватно би се овде јавила замерка зашто уопште разматрати све ове горе наведене приступе и проблеме када се они вероватно могу избећи коришћењем сервиса заснованих на инфраструктури као сервису (Instrastructure as a Service – IaaS) или софтверу као сервису (Software as a Service - SaaS) у облацима (cloud) па ће се кратко дати осврт и на ово питање. Наведени сервиси нису тренутно адекватно решење за примену, посматрано кроз оно што класичне виртуелне машине нуде, у високошколским установама из неколико разлога. Прво, не могу се сви софтвери и њихове примене које се користе у високошколском образовању адекватно и у потпуности реализовати коришћењем наведених сервиса у облацима. Саме цене услуге ових сервиса за велику већину високошколских институција представљају изузетан финансијски напор, па би узимајући ову чињеницу у обзир, заснивање решења на овим архитектурама учинило да решење не буде доступно свима, а нарочито решење не би могле имплементирати, или би их имплементирали делимично уз велике финансијске напоре, високошколске институције са далеко скромнијим буџетима. И на крају треба поменути да постоје одређена законска питања која још нису адекватно решена, посматрано у односу на тренутну законску регулативу Републике Србије, а која се тичу заштите личних података, јавних набавки и сличних законских питања. Без улажења у даљу, опширнију, дискусију, види се да ако би се решење заснивало на архитектури у облацима оно би донело додатне компликације и ограничености у самој имплементацији, тако да о овим сервисима није дискутовано, нити ће се даље разматрати, већ је одлучено да се решење заснива на поменутим концептима везаним за употребу виртуелних машина на начин како ће бити представљене у предметној докторској дисертацији.

Као што се види из претходно изложеног и разматраног, главну окосницу проблема представља чињеница да је у свим наведеним случајевима тежиште интересних група делимично померено са остваривања исхода учења на техничко обезбеђење услова и одређених норми што изазива значајно додатно оптерећење свих укључених страна у наставном процесу. Претходно поменуто у значајној мери може утицати на постизање позитивних исхода учења који се остварују кроз реализацију одређених наставних процеса. Такође, претпоставља се да сви заинтересовани поседују довољан обим техничког знања из различитих домена рачунарске технологије (оперативни системи, софтверско инжењерство, виртуелизација итд.), док се у пракси најчешће јављају ситуације везане за недостатак неких основних знања из области рачунарске технологије, или поседовање таквог знања у недовољном обиму, а чести су и случајеви погрешног тумачења, као и недовољне информисаности приликом доношења важних одлука и спровођења кључних радњи. Постизање уједначености целокупног процеса отежано је и због тежње да сваки корисник самостално спроводи процес инсталације виртуелне машине онако како он то разуме и доживљава, што може увести нове непознанице у реализацију целокупног наставног процеса које је у неким случајевима јако тешко тумачити и разумети због великог присуства индивидуализма у самом процесу, а могу се појавити и нови, додатни проблеми које је потребно адекватно превазићи, што непотребно повећава комплексност реализације самог наставног процеса који се посматра.

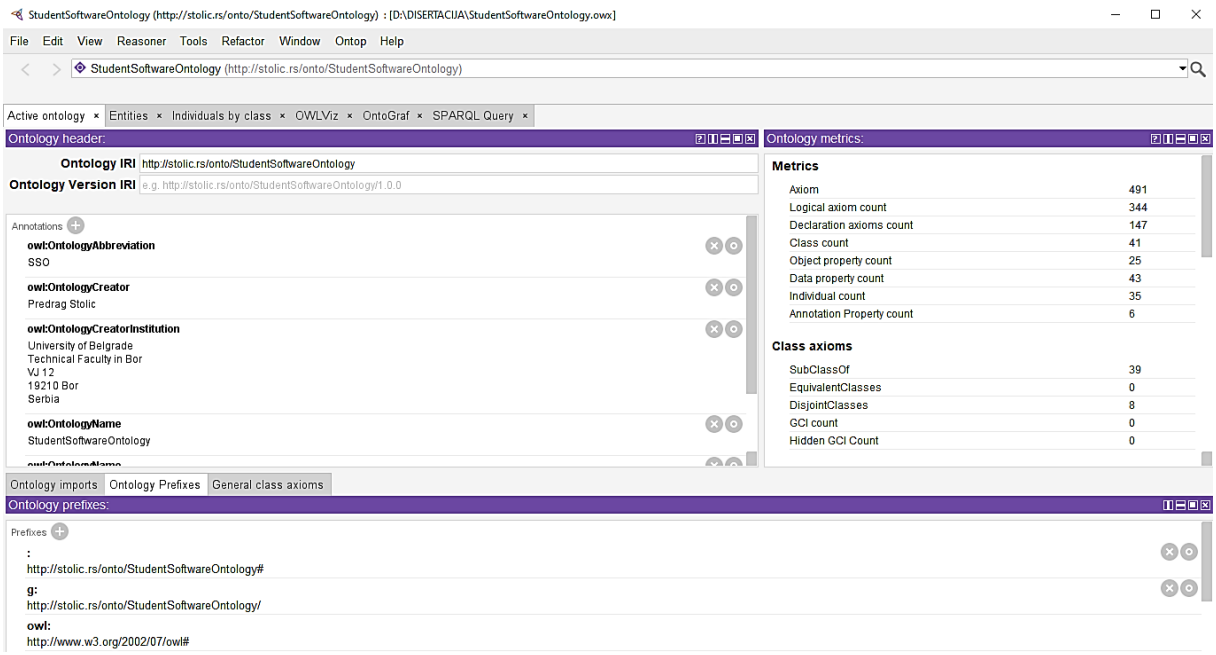
Минимизација горе наведених и описаних проблема из софтверског домена извршиће се употребом принципа онтолошког инжењеринга. Решење засновано на употреби одговарајуће онтологије ће омогућити адекватну заступљеност и применљивост знања неопходних за превазилажење проблема из софтверског домена

коришћења виртуелних машина. Сходно томе, у наставку ће бити представљен развој адекватне онтологије која омогућава адекватно управљање софтвером за виртуелне машине када се оне користе за потребе реализације наставних процеса у области високошколског образовања.

3.3. Развој онтологије

Употреба онтолошког инжењерства за превазилажење претходно идентификованих проблема наметнула се као императив будући да је уочена потреба за дељењем и поновном употребом стечених знања без увођења неког великог нивоа комплексности имплементираног решења, уз избегавање редунданције која се јавља је постоји велики део општег знања за које се стицање знања врши од почетка приликом увођења сваког новог софтвера [13]. Кроз онтологије врши се адекватно моделирање на основу реализације која се постиже коришћењем одговарајућих класа, атрибута и међусобних односа и које кроз спецификацију, концептуализацију и имплементацију у хетерогено рачунарско окружење пружају једну заједничку семантику са могућношћу адекватне размене знања и поновне употребе [14-17]. Принципи засновани на онтологијама омогућили су превазилажење широког спектра проблема који су настали у домену софтверског инжењеринга и имплементације информационих система развојем нових приступа, а постоје и случајеви допуњавања већ постојећих методологија онтологијама и креирањем интегрисаних приступа [18,19], па можемо рећи да су онтологије већ доказано средство у области управљања информацијама и у области управљања конфигурацијом [20,21] и дају адекватан одговор на хетерогено окружење јер обезбеђују неопходне технике за идентификацију, као и технике неопходне за креирање адекватног одговора на промене у окружењу [22,23]. Ако посматрамо системе намењене високошколском образовању, како се тврди у [40], због убрзаног технолошког развоја и изузетног испољавања хетерогености постоји проблем успостављања одговарајућег рачунарског окружења и информационог система, али управо на овом месту принципи онтолошког моделирања дају изузетно добру основу за превазилажење ових проблема. Управо из ових разлога напред представљена онтологија намењена употреби у области високог образовања омогућава креирање идентичних софтверских окружења у оквиру хетерогеног рачунарског окружења, при чему се, на одређеном нивоу апстракције, врши трансформација хетерогености у хомогеност средине.

За развој онтологије, настале као резултат истраживања спроведених у оквиру ове докторске дисертације [125], коришћен је бесплатни софтвер отвореног кода Protégé [30,31] у својој десктоп мултиплатформској варијанти верзије 5.6.1 који у потпуности подржава развој онтологија коришћењем OWL језика [32] и који представља највише широко распрострањено и популарно окружење за развој онтологија [33]. У оквиру поменутог окружења може се извршити адекватна визуелизација одговарајућих класа и међусобних односа, као и провера конзистентности самих онтологија кроз интегрисане расуђиваче (reasoners). Типично Protégé десктоп окружење представљено је на слици 33.



Слика 33. Изглед мултиплатформског Protégé десктоп окружења коришћеног за израду предметне онтологије

У литератури постоји више приступа одређивању типа онтологије у зависности од предмета класификације, као и неколико приступа дефинисању методологија за развој онтологија. Не улазећи у дубљу анализу претходног исказа, онтолошки приступ који ће се овде користити било би најпогодније да се сврста у групу приступа директно зависних од примене, односно саме онтологије према типу у онтологије апликације [25]. У домену онтолошког инжењерства развијено је неколико језика који се могу успешно користити при развоју различитих врста онтологија. Приликом развоја предметне онтологије искоришћене су предности које нуди OWL2 Web Ontology Language [26] због своје широке распрострањености и доказане употребљивости, будући да се највећи број онтолошких решења у софтверском домену реализује управо коришћењем поменутог језика [27,28]. Такође, коришћењем OWL/XML формата, представљање онтологије се постиже у разумљивијем формату [29], који је погоднији за даљу употребу у одговарајућим програмским језицима и апликацијама.

Тренутне вредности које улазе у метрику развијене онтологије, а које су добијене генерисањем вредности у Protégé развојном окружењу, дате су у табели 9.

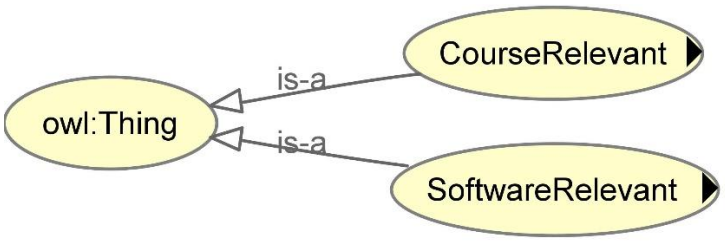
Табела 9. Тренутна метрика развијене онтологије генерисана од стране Protégé развојног окружења

Metrics	
Axiom	491
Logical axiom count	344
Declaration axioms count	147
Class count	41
Object property count	25
Data property count	43
Individual count	35
Annonation Property	6
Class axioms	
SubClassOf	39
DisjointClasses	8

Object property axioms	
SubObjectPropertyOf	24
InverseObjectProperties	12
ObjectPropertyDomain	25
ObjectPropertyRange	25
Data property axioms	
SubDataPropertyOf	42
DataPropertyDomain	42
Individual axioms	
ClassAssertion	35
ObjectPropertyAssertion	28
DataPropertyAssertion	63

3.3.1. Класе

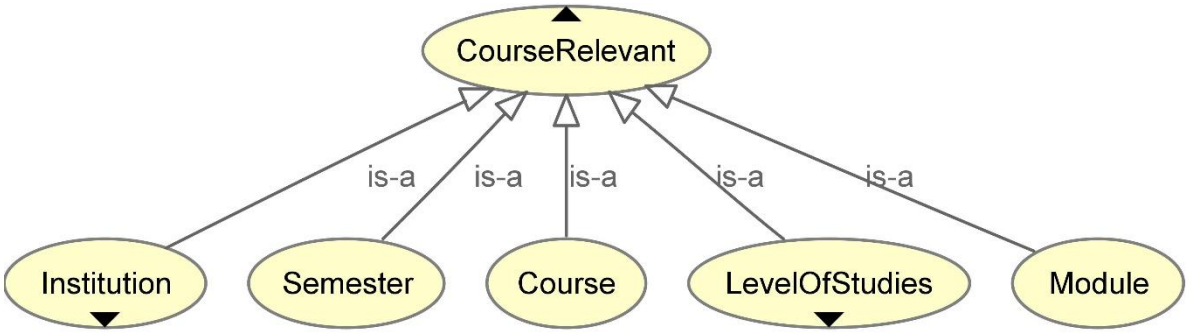
У Protégé развојном окружењу све класе дефинишу се као потклаце owl:Thing која хијерархијски представља увек највишу класу. Ова класа је предефинисана класа у OWL2 чији је главни циљ обезбеђивање адекватне хијерархије класа у онтологији која се реализује [126]. У складу са тим, у односу на ову класу дефинисане су две класе CourseRelevant и SoftwareRelevant, за реализовану онтологију хијерархијски највише класе, од којих ће се даље вршити формирање осталих потклаца, као што је приказано на слици 34.



Слика 34. Хијерархијски приказ две главне класе развијене онтологије коришћењем OWLViz додатка у оквиру Protégé десктоп окружења

У циљу адекватног превазилажења претходно представљеног проблема, постоји потреба за повезивањем знања из два домена. Будући да се посматрају проблеми везани за појаву хетерогености рачунарског окружења у реализацији наставних процеса у високошколском образовању, неопходна ће нам бити одређена знања из домена високошколског образовања. За реализацију неопходних знања користиће се потклаце формиране у оквиру класе CourseRelevant, па се може рећи да CourseRelevant у суштини означава академски домен. Међутим, пошто се у овом случају разматрају проблеми софтверске природе који настају приликом превазилажења сложености хетерогеног рачунарског окружења, поред знања о академском домену биће потребна и одређена знања из софтверског домена, па ће се за реализацију потребних знања у домену софтвера извршити одговарајуће формирање потклаца у оквиру класе SoftwareRelevant. Из напред наведених разлога, као што је већ поменуто, класе CourseRelevant и SoftwareRelevant заузимаће највиши ниво у хијерархији класа, непосредно испод класе највишег нивоа, односно owl:Thing класе. У наставку ће обе класе заједно са својим потклasама бити детаљније објашњене будући да се, као што је већ наведено, односе на два различита домена која представљају.

Кључне потклаце садржане у оквиру класе CourseRelevant приказане су на слици 35.



Слика 35. Приказ главних потткласа класе CourseRelevant коришћењем OWLViz додатка у оквиру Protégé десктоп окружења

Приликом идентификације кључних елемената битних за развој онтологије у академском домену, коришћен је систем високошколског образовања Републике Србије садржан у Закону о високом образовању Републике Србије [127]. У имплементацији студија које се реализују у оквирима високошколског образовања, полазна тачка је сама високошколска институција (класа Institution), односно установа у оквиру које се реализују сви процеси неопходни за адекватно извођењем поменутих студија. Та установа реализује одговарајући студијски програм који је планиран за реализацију у оквиру одговарајућег нивоа студија (класа LevelOfStudies). Нивои студија су дефинисани у складу са националном класификацијом нивоа студија у образовном систему Републике Србије [128]. Настава у оквиру студијских програма, у складу са важећим акредитационим актима Републике Србије, остварује се у оквиру одговарајућих студијских модула (класа Module) којих може бити и више, али минимално мора бити један. Сваки предмет (класа Course) који студент похађа реализује се у оквиру одговарајућег модула студијског програма, истовремено за сваки предмет важи и правило да је исти предвиђен за похађање у одређеном семестру (класа Semester).

Приликом претходног разматрања и приказа класа датог на слици 35 направљене су одређене генерализације. Међутим, како би знање садржано у оквиру развијене онтологије представљало реално стање, морају се извршити одговарајуће специјализације над појединим елементима, па ћемо у складу са тим сада за класу CourseRelevant имати једну ширу имплементацију која је приказана на слици 36.

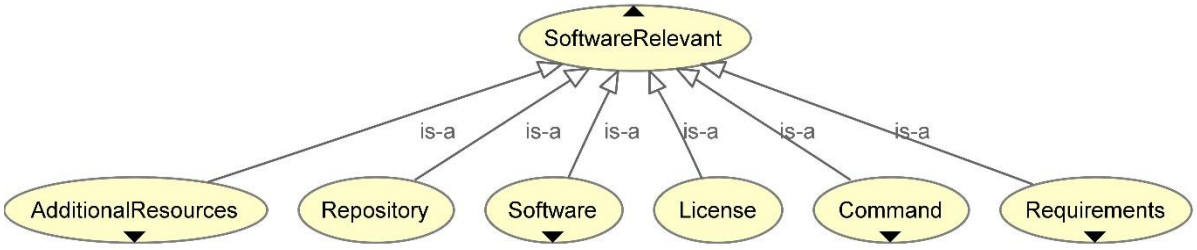
У оквиру модела који предвиђа законска основа која се примењује у Републици Србији, постоји више установа које могу реализовати студијске програме у оквиру спровођења система високошколског образовања. То су одговарајуће академије (класа Academy) са својим високим школама (класа College) и универзитети (класа University) са својим инситутима (класа Institute) и факултетима (класа Faculty). Без обзира на постојање одговарајуће институционалне припадности (на пример, један факултет или институт припада једном универзитету), све класе које означавају институције се реализују на истом хијерархијском нивоу, јер не можемо, на пример, да генерализујемо институте са универзитетима. Они деле неке своје заједничке карактеристике и интересовања, али са становишта моделирања наставног процеса који се у њима одвија, треба их посматрати одвојено, па је у складу са тим и извршено моделирање у оквиру онтологије на начин како је то претходно приказано.

Ниво студија обухвата студије првог, другог и трећег нивоа (класе LevelI, LevelII и LevelIII), као и посебне кратке студијске програме (класа ShortStudyProgram). Студије првог нивоа обухватају основне академске студије (класа UndergraduateAcademic), као и основне (BachelorApplied) и специјалистичке (класа BachelorApplied) струковне студије. Студије другог нивоа обухватају мастер (класа MasterAcademic) и специјалистичке (класа SpecializedAcademic) академске студије, као и мастер струковне студије (класа

MasterApplied). Студије трећег нивоа тренутно обухватају само докторске академске студије (класа DoctoralAcademic).

Слика 36. Хијерархија класа садржаних у оквиру класе CourseRelevant коришћењем OWLViz додатка у оквиру Protégé десктоп окружења

По сличним принципима претходно коришћеним за класу CourseRelevant, извршиће се идентификација свих кључних потклаcа садржаних у оквиру класе SoftwareRelevant на начин приказан на слици 37.



Слика 37. Приказ главних потклаcа класе SoftwareRelevant коришћењем OWLViz додатка у оквиру Protégé десктоп окружења

Логично, први кључни елемент овде ће бити сам софтвер (класа Software). Сваки софтвер има неке специфичне захтеве (класа Requirements) који морају бити испуњени да би се софтвер могао уопште имплементирати и да би се та имплементација постигла на одговарајући захтевани начин. Такође, сваки софтвер, без обзира на врсту којој припада, дистрибуира се под одређеном лиценцом (класа License) и правилима

садржаним у њој. Софтвер може имати и одговарајуће додатне компоненте (класа AdditionalResources) без којих софтвер може да функционише, али које га на одређени начин допуњују, па су и оне додате у крајње разматрање јер некада пружају неке додатне функционалности везане за софтвер које могу бити битне за студенте. Да би процес инсталације био адекватно започет, систему мора бити прослеђена одговарајућа команда за инсталацију софтвера (класа Command) или скуп команди које ће омогућити систему да изврши неопходне иницијализације и започне процедуру инсталације. Овде треба напоменути да се целокупна конструкција онтологије и предложеног система изграђеног око ње заснива на коришћењу одговарајуће Linux дистрибуције као оперативног система гостујућег система, односно као оперативног система који ће бити инсталиран у оквиру виртуелне машине, због специфичности Linuxa да се може слободно редистрибуирати и да се може слободно прилагођавати разним потребама [129]. Софтвер на Linux системима је лоциран у оквиру одговарајућих структурираних локација са којих се врши преузимање софтвера током инсталације, а које се називају спремишта (repositories) [130]. Сходно томе, и у овом случају ће се морати извршити одговарајуће референцирање на та спремишта (класа Repository) због специфичности коришћеног система.

И у овом случају приликом разматрања извршене су одређене генерализације, међутим и овде важи правило које је примењено у случају CourseRelevant класе везано за адекватну презентацију знања садржаног у оквиру развијене онтологије. У складу са тим ће и на овом месту бити извршене одговарајуће специјализације над појединим елементима, па ће класа SoftwareRelevant, такође, имати једну ширу имплементацију која је приказана на слици 38.

Прво треба поменути да је у случају представљене онтологије, софтвер подељен у три категорије: апликативни софтвер (класа ApplicationSoftware) који представља најчешћи тип софтвера који користе студенти, системски софтвер (класа SistemSoftware) који представља различите надоградње оперативног система, као и софтвер намењен за системска подешавања и софтвер за програмирање (класа ProgrammingTools) који је намењен за кодирање, развој, тестирање, отклањање грешака и сличне задатке. Што се тиче наведених захтева, који суштински представљају у већини случајева системске захтеве, дефинисана су три типа захтева: минимални (класа Minimal) који представљају доњу хардверску и софтверску границу потребну за нормалну инсталацију и функционисање жељеног софтвера, оптимални (класа Optimal) који представљају такве хардверске и софтверске критеријуме који омогућавају удобан рад током инсталације и коришћења софтвера и препоручених (класа Recommended) који су дати на основу одређеног корисничког искуства, било да потиче од студената или од техничког и другог особља саме установе. Претходно је споменуто да постоје и неке додатне компоненте (AdditionalResources) које нису обавезне током инсталације, али се могу инсталирати и укључити ако постоји потреба за њима, а оне су у случају предметне онтологије идентификоване као примери (класа ExampleFile), конфигурациони фајлови (класа ConfigurationFile) и упутства (класа ManualFile) који омогућавају бољу припрему крајњег корисника (студента) за задатке које је потребно извршити путем обезбеђеног софтвера, као и обезбеђивање неких додатних функционалности, као и остварења бољих перформанси софтвера кроз одговарајуће конфигурисање. Када говоримо о командама које се реализују у циљу имплементирања инсталационог процеса неког одређеног софтвера, овде је такође било важно извршити одређену специјализацију како би се јасно раздвојиле одређене фазе кроз које се морају проћи пре стављања софтвера у употребу и постизања његове употребљивости за реализацију задатака у наставном процесу. У складу с тим, идентификоване су команде за иницијализацију саме инсталације и почетак инсталационог процеса (класа InstallationCommand) без којих никако не би било могуће остварити саму инсталацију софтвера, али су, поред ових команди, укључене и

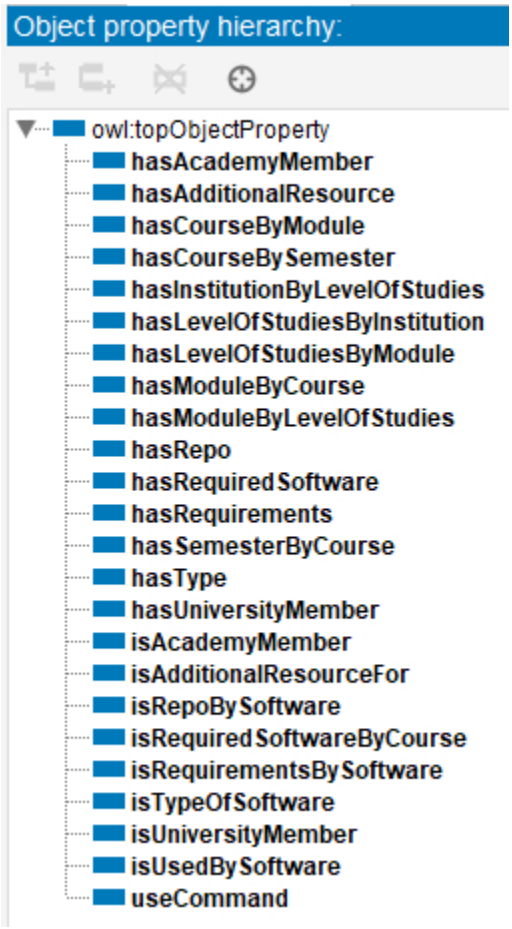
постинсталационе команде (класа PostInstallationCommand) како би се омогућило евентуално додавање неке додатне функционалности у постојећи софтвер касније, као и конфигурационе команде (класа ConfigurationCommand) јер је често потребно променити конфигурацију из подразумеване (default) у кориснички дефинисану (user defined), као што су на пример операције попут промене директоријума у којем се врши снимање одговарајућих датотека релевантних за сам софтвер, укључивање неких софтверских додатака и слично.

Слика 38. Хијерархија класа садржаних у оквиру класе SoftwareRelevant коришћењем OWLViz додатка у оквиру Protégé десктоп окружења

3.3.2. Везе (својства објеката)

Претходно дефинисане класе представљају један од основних елемената модела који се реализује остваривањем одговарајуће онтологије. Међутим, да би се реализовао модел

употребљив у стварном окружењу, потребно је остварити одговарајућу повезаност између класа, односно дефинисати одређене релације које се реализују у оквиру модела. У Protégé развојном окружењу везе (релације) које се остварују међу класама су означене као својства објеката (Object Properties), као што се види са слике 39.

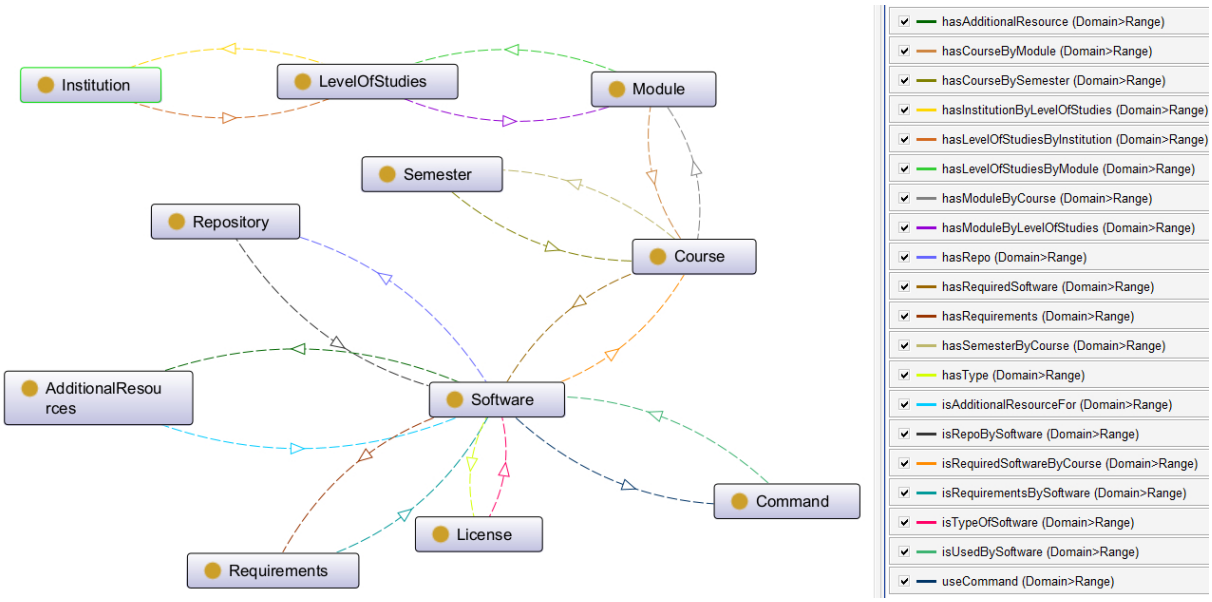


Слика 39. Хијерархија веза (особина објеката) приказана у оквиру Protégé десктоп окружења

Као што се види са слике 39, све везе, односно сва својства објеката су, користећи сличне принципе које смо сусретали и код хијерархијског приказа класа, дефинисана као подособине owl:topObjectProperty која хијерархијски представља највишу особину објекта. Објашњење је слично објашњењу које смо имали код класа везано за класу owl:Thing. Највише својство објекта дефинисано као owl:topObjectProperty представља унапред дефинисано својство објекта у OWL2 и главни циљ овог својства објекта је да обезбеди адекватну хијерархију својстава објекта.

У оквиру онтологије која се овде разматра постоји 24 идентификована својства објекта, међутим, у наставку ће бити размотрено само половина од укупног броја, будући да друга половина, као што се може видети из метрике приказане у табели 1, представљају инверзна својства објекта у односу на прву половину, тако да није потребно додатно објашњење.

На слици 40 приказана је реализација главних веза коришћењем репрезентације засноване на софтверу GraphViz [131] за визуелизацију преко OntoGraph додатка у Protégé окружењу.



Слика 40. Графички приказ остварених главних веза у оквиру онтологије коришћењем OntoGraph додатка у оквиру Protégé десктоп окружења

Ако боље погледамо графички приказ дат на слици 40, видимо да се график условно може поделити на два повезана дела, горњу половину графика који представља повезане класе из CourseRelevant домена и доњу половину који представља повезане класе из SoftwareRelevant домена.

За CourseRelevant домен приказану међусобну повезаност класа тумачимо на следећи начин: Нека високошколска институција (Institution) реализује студијски програм у оквиру одређеног нивоа студија (LevelOfStudies) који поседује одређени модул (Module) на том студијском програму у оквиру ког се слуша дати предмет (Course) у одређеном семестру (Semester). Као што се види, повезивање је смислено и на овај начин стварно се може добити на основу унетих података једна комплетна информација, а на основу одговарајућих информација се може произвести адекватно знање.

Слично претходном, можемо растумачити повезаност класа и за SoftwareRelevant домен: Софтвер (Software) поседује одређене системске захтеве (Requirements) како би радио у складу са задацима који се постављају пред њега. Рад са софтвером (Software) се остварује под условима дефинисаним у одговарајућој лиценци (License). Инсталација софтвера (Software) врши се путем одговарајуће команде (Command), а сама инсталациона верзија софтвера (Software) садржана је у одговарајућем спремишту (Repository). Уз софтвер (Software) могу се инсталирати и додатни материјали (AdditionalResources). Када се упореде претходна тврђења са оним што је приказано на претходном графичком приказу и у овом случају долази се до закључка да је, такође, извршено смислено повезивање и да се оваквим повезивањем може произвести одговарајуће доменско знање.

Примењујући основне принципе онтолошког инжењерства, који између осталог, кажу да је субјект повезан са одговарајућим објектом преко одговарајућег предиката, основне везе које се користе међу класама у одговарајућим доменима посматрања дефинисаћемо на начин представљен у табели 10 за CourseRelevant домен, односно у табели 11 за SoftwareRelevant домен. Овде треба напоменути да је у мултиплатформском Protégé декстоп развојном окружењу приликом дефинисања веза субјекат означен као Domain, објекат је означен као Range, док је предикат, као што је поменуто више пута у претходним излагањима означен као Object Property.

Табела 10. Дефинисање основних веза за CourseRelevant домен

Субјекат	Предикат	Објекат
Institution	*hasLevelOfStudiesByInstitution*	LevelOfStudies
LevelOfStudies	*hasModuleByLevelOfStudies*	Module
Module	*hasCourseByModule*	Course
Course	*hasSemesterByCourse*	Semester

Табела 11. Дефинисање основних веза за SoftwareRelevant домен

Субјекат	Предикат	Објекат
Software	*hasRequirements*	Requirements
Software	*hasType*	License
Software	*useCommand*	Command
Software	*hasRepo*	Repository
Software	*hasAdditionalResource*	AdditionalResources

Из претходног се види да се остварује одређено доменско знање у доменима CourseRelevant и SoftwareRelevant. Међутим, како би се остварила пуна функционалност решења за које је предметна онтологија предвиђена, парцијална доменска знања потребно је синтетисати у једно глобално знање, па је у складу са тим неопходно остварити адекватну повезаност претходна два домена. Претходно ће се остварити на следећи начин: Одговарајући предмет (Course) користи одговарајући софтвер (Software) за остваривање својих задатака и циљева. Поштујући претходну дефиницију остварених веза кроз форму субјекат – предикат – објекат, биће:

Course *hasRequiredSoftware* Software

чиме је сада остварена потпуна повезаност унутар разматране онтологије и добијено комплетно представљање знања унутар саме онтологије на начин који одговара решењу постављеног проблема који се савладава принципима онтолошког инжењерства.

Поред презентованих основних веза, треба поменути још две додатне везе које су реализоване у оквиру CourseRelevant домена, а које су приказане у оквиру табеле 12 по већ поменутим принципима. Обе везе су додате како би дефинисале посебне институционалне припадности. Прва веза дефинише чињеницу да су, у складу са високошколским уређењем у Републици Србији, академије (Academy) састављене од чланица које су представљене високим школама (College), док се у другој вези по сличном принципу дефинише чланство у оквиру универзитета (University) где чланице могу бити факултети (Faculty), али и институти (Institute).

Табела 12. Дефинисање додатних веза за CourseRelevant домен

Субјекат	Предикат	Објекат
Academy	*hasAcademyMember*	College
University	*hasUniversityMember*	Institute, Faculty

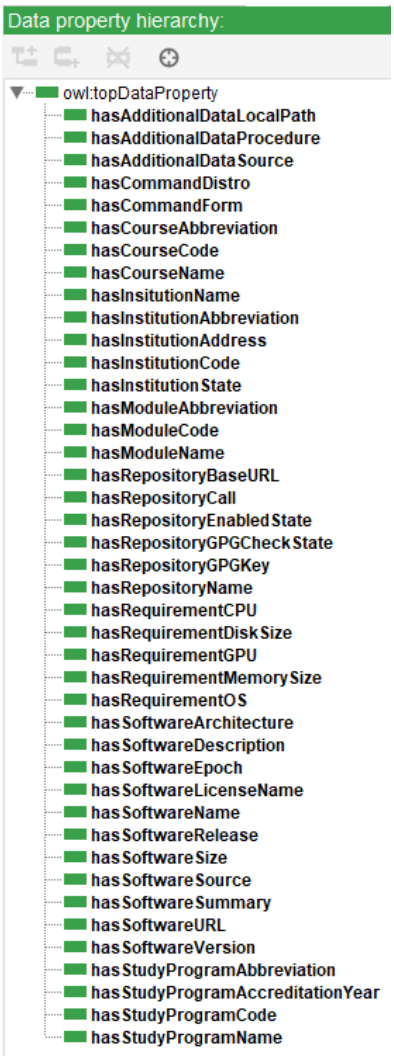
Као што је већ поменуто, сваки од предиката има одговарајући инверзни предикат, па су на основу инверзних предиката реализоване одговарајуће инверзне везе. У овом случају за све везе дефинисане у табелама 10, 11 и 12, као и за везу која повезује CourseRelevant и SoftwareRelevant домен, субјекат постаје објекат и обрнуто, објекат постаје субјекат, уз повезивање одговарајућим инверзним предикатом, али се ове везе неће посебно разматрати.

3.3.3. Атрибути

Претходно је указано на изградњу предметне онтологије на основу одређених класа и њихових међусобних повезаности реализацијом одговарајућих веза како би се остварила адекватно доменско знање. Међутим, да би се остварила потпуност и недвосмисленост знања које ће се добијати на основу садржаних података у оквиру онтологије, неопходно је увести и одређене атрибуте који ће омогућити даљи приказ потребних карактеристика.

На пример, ако се уведе инстанца класе Институција (Institution) под називом Факултет_1, без адекватних атрибута знаће се само за њено постојање, а постоји потреба и да се о инстанци Факултет_1 зна назив те институције, локација, држава и многе друге информације. Овакве важне информације за сваку инстанцу класе додаваће се у одговарајуће атрибуте, што говори о изузетној важности правилног дефинисања атрибута поред претходних активности дефинисања класа и међусобних веза.

У Protégé развојном окружењу атрибути су означени као својства података (Data Properties). Списак коришћених својстава података у посматраној онтологији приказан је на слици 41.



Слика 41. Хијерархија атрибута (особина података) приказана у оквиру Protégé десктоп окружења

И код атрибута се јавља сличан принцип оном који се јавио при дефинисању класа и одговарајућих веза, односно јавља се ситуација да су сва својства података дефинисана као подособина owl:topDataProperty који хијерархијски представља највише својство података, који је унапред дефинисано својство и чија је примарна улога обезбеђивање неопходне хијерархијске структуре која се односи на сва генерисана својства података. Тренутно постоји идентификованих 42 својстава података у складу са метрикама приказаним у табели 1. Описи идентификованих својстава података приказани су у оквиру табеле 13.

Табела 13. Тренутно идентификована својства података

Назив својства података	Класа	Кратак опис
hasAdditionalDataLocalPath	AdditionalResources	Путања на којој се врши смештај додатака везаних за софтвер
hasAdditionalDataProcedure	AdditionalResources	Начин на који се врши преузимање додатака везаних за софтвер
hasAdditionalDataSource	AdditionalResources	Извор са ког се врши преузимање додатака везаних за софтвер
hasCommandDistro	Command	Linux дистрибуција за коју је команда применљива
hasCommandForm	Command	Облик команде
hasCourseAbbreviation	Course	Скраћено име предмета
hasCourseCode	Course	Шифра предмета
hasCourseName	Course	Име предмета
hasInstitutionName	Institution	Име институције
hasInstitutionAbbreviation	Institution	Скраћено име институције
hasInstitutionAddress	Institution	Адреса институције
hasInstitutionCode	Institution	Шифра институције
hasInstitutionState	Institution	Држава институције
hasModuleAbbreviation	Module	Скраћено име модула
hasModuleCode	Module	Шифра модула
hasModuleName	Module	Име модула
hasRepositoryBaseURL	Repository	Адреса на којој се налази спремиште софтвера
hasRepositoryCall	Repository	Начин приступа спремишту софтвера
hasRepositoryEnabledState	Repository	Провера стања спремишта софтвера (укључено/искључено)
hasRepositoryGPGCheckState	Repository	Да ли се приликом рада са спремиштом софтвера врши GPG провера
hasRepositoryGPGKey	Repository	GPG кључ за спремиште софтвера
hasRepositoryName	Repository	Назив спремишта софтвера
hasRequirementCPU	Requirements	Потребан CPU за рад са софтвером

hasRequirementDiskSize	Requirements	Потребан слободан простор на диску за рад са софтвером
hasRequirementGPU	Requirements	Потребан GPU за рад са софтвером
hasRequirementMemorySize	Requirements	Потребна количина меморије за рад са софтвером
hasRequirementOS	Requirements	Потребан оперативни систем за рад са софтвером
hasSoftwareArchitecture	Software	Архитектура за коју је намењен софтвер
hasSoftwareDescription	Software	Детаљан опис софтвера
hasSoftwareEpoch	Software	Епоха софтвера
hasSoftwareLicenseName	Software	Назив софтверске лиценце
hasSoftwareName	Software	Назив софтвера
hasSoftwareRelease	Software	Издање софтвера
hasSoftwareSize	Software	Величина инсталације софтвера
hasSoftwareSource	Software	Инсталациона датотека софтвера
hasSoftwareSummary	Software	Кратак опис софтвера
hasSoftwareURL	Software	Web страница софтвера
hasSoftwareVersion	Software	Верзија софтвера
hasStudyProgramAbbreviation	LevelOfStudies	Скраћено име студијског програма
hasStudyProgramAccrediationYear	LevelOfStudies	Година акредитације студијског програма
hasStudyProgramCode	LevelOfStudies	Шифра студијског програма
hasStudyProgramName	LevelOfStudies	Име студијског програма

Сва идентификована својства података су тренутно заснована на коришћењу четири типа података. Најзаступљенији тип података је xsd:string, праћен типовима података xsd:anyURI, xsd:integer и xsd:decimal. У овом тренутку није било потребе да се дефинишу додатни типови корисничких података, па су коришћени само примитивни типови података дефинисани у оквиру XSD (XML Schema Definition) [132].

Постојећа онтологија може се лако прилагодити увођењем нових својстава података поред постојећих, ако постоји потреба за тим без додатне промене класа и веза, тако да суштински може да реагује на одговарајуће испољавање одређених промена у окружењу у коме онтологија испуњава свој задатак (нпр. промена кључних софтверских својстава и слично).

3.3.4. Унапред дефинисани ентитети

У Protégé развојном окружењу свака инстанца одговарајуће класе означена је као индивидуа (Individual) како би се јасно назначило да се ради о појединачној појави објекта једне класе. Формирање већине индивидуа врши се уносом одређених података од стране крајњег корисника, међутим, постоје и одређене индивидуе које се морају формирати и које су формиране током пројектовања предметне онтологије, односно, у

односу на крајњег корисника, оне се понашају се као унапред дефинисане индивидуе (pre-defined indviduals).

Увођење унапред дефинисаних индвидуа чини се како би се смањила комплексност уноса података за крајњег корисника и како би се обезбедила конзистентност у означавању одређених појмова које покрива рад са предметном онтологијом. Генерално посматрајући, адекватан рад онтологије би могао да се оствари и без увођења унапред дефинисаних индивидуа, односно њиховим накнадним уносом, али оваква реализација чини једну добру праксу са циљем умањења вероватноће појаве грешке накнадним радом са онтологијом услед самосталног тумачења и примене одговарајућих концепата који би требали бити јединствени на нивоу целокупног високошколског образовног система.

Узимајући у обзир напред наведено, у посматрану онтологију унете су индивидуе које представљају одговарајуће семестре приликом реализације наставних активности, пошто су семестри идентични и јединствени у целокупном високошкослком образовном простору Републике Србије, односно не постоји различита примена истих у различитим институцијама. Такође, током пројектовања предметне онтологије креиране су и одговарајуће унапред дефинисане индивидуе над класом која покрива рад са лиценцама, пошто постоје добро познате и дефинисане врсте лиценци које нису подложне променама, односно користе се такве какве јесу.

Као што смо имали случај код атрибута и овде важи правило да се, ако постоји потреба за одговарајућом реакцијом на променљивост окружења, може извршити одређена промена над унапред дефинисаним индивидуама, додати нове и извршити сличне активности у циљу што бољег представљања знања садржаног у оквиру предметне онтологије. Када се разматра уношење инвидивуда од стране крајњег корисника, мисли се на одређено овлашћено лице са високошколске установе (техничко особље, наставно особље и слично) које ће унети све потребне елементе за правилно дефинисање сваке појединачне индивидуе кроз одговарајуће форме и генерисане функције које омогућавају одређену аутоматизацију процеса. При томе овде треба напоменути да треба имати на уму да дефинисање индвидуе на подразумева само доделу одговарајућег имена, већ морају адекватно бити додељени и други елементи, попут дефиниције припадности одговарајућој класи, одговарајућа својства уколике постоје и остали неопходни елементи, што некада за особу која не поседује неке основне представе о принципима онтолошког инжењерства може бити изузетно комплексна процедура. Зато треба бити обазрив када се врше овакве промене над онтологијом како се не би нарушила конзистентност саме онтологије и како не би евентуално дошло до представљања знања која не одговарају стварном окружењу.

3.3.5. Расуђивање

Претходно дефинисани елементи пружају основу у циљу пружања адекватних знања студенту као крајњем кориснику како би их адекватно и на прави начин применио током процеса инсталирања софтвера и на тај начин добио одговарајуће рачунарско окружење за реализацију задатака у оквиру наставног процеса чији је студент активни учесник. Сходно томе, посебна пажња се мора обратити на адекватност добијене онтологије и проверу присуства одређених недодстатака пре стављања у употребу у стварном наставном процесу.

Како би се избегли потенцијални недостаци у оквиру изградње онтологије користи се процес расуђивања заснован на употреби одговарајућих специјализованих софтвера под именом расуђивачи (reasoners). У случају предметне онтологије за проверу онтологије процесом расуђивања коришћена су два расуђивача: FaCT++ у верзији 1.6.5 и HermiT у верзији 1.4.3.456. Оба расуђивача коришћена су у оквиру мултиплатформског

Protégé десктоп развојног окружења. Након покретања оба расуђивача један за другим и увидом у одговарајућу лог датотеку која је том приликом креирана унутар развојног окружења, може се видети да није идентификована ниједна грешка приликом покретања оба расуђивача, што је приказано на слици 42.

```
INFO  15:48:11  ------------------------------ Running Reasoner ------------------------------
INFO  15:48:11  Pre-computing inferences:
INFO  15:48:11      - class hierarchy
INFO  15:48:11      - object property hierarchy
INFO  15:48:11      - data property hierarchy
INFO  15:48:11      - class assertions
INFO  15:48:11      - object property assertions
INFO  15:48:11      - same individuals
INFO  15:48:11  Ontologies processed in 33 ms by FaCT++
INFO  15:48:11
INFO  15:48:33  ------------------------------ Running Reasoner ------------------------------
INFO  15:48:33  Pre-computing inferences:
INFO  15:48:33      - class hierarchy
INFO  15:48:33      - object property hierarchy
INFO  15:48:33      - data property hierarchy
INFO  15:48:33      - class assertions
INFO  15:48:33      - object property assertions
INFO  15:48:33      - same individuals
INFO  15:48:33  Ontologies processed in 456 ms by HermiT
INFO  15:48:33
```

Слика 42. Извод из лог датотеке Protégé десктоп окружења након покретања FaCT++ и HermiT расуђивача

Као што се може видети са горње слике, расуђивачи проверавају конзистентност онтологије обављањем неколико задатака. Прво се доносе одређени закључци везани за хијерархије које су дефинисане у оквиру постојеће онтологије, односно анализира се хијерархија класа, хијерархија веза, односно предиката (хијерархија својстава објекта), као и хијерархија атрибута (хијерархија својстава података). У следећем кораку се обрађују и анализирају одговарајући искази, односно тврдње везане за класе (Class Assertions) и везе (Object Property Assertions). У завршном кораку врши се анализа индивидуа. У оквиру Protégé окружења, уколико дође до било какве недоследности током рада расуђивача, то се евидентира на одговарајући начин у оквиру поменуте лог датотеке и може се јасно и недвосмислено утврдити где је наступила одређена недоследност, на шта се она односи. Тада се у самом окружењу могу добити додатне информације о уоченим недоследностима које су значајне приликом даљих поступака у отклањању истих. Међутим, овде треба бити опрезан, јер расуђивачи нису свемоћно средство за проверу ваљаности онтологије, будући да они могу да провере конзистентност онтологије, али не могу да провере да ли онтологија у потпуности тачно представља жељену семантику.

3.4. Развој пратећег софтвера за онтологију

Претходно је показан комплетан развој формалне онтологије са циљем да омогући адекватно складиштење знања које ће се користити за превазилажење проблема из софтверских домена приликом превазилажења хетерогености рачунарског окружења када се оно користи за потребе реализације наставних процеса у високошколском образовању на напред описане начине. Сазнања у оквиру развијене онтологије омогућена су формирањем одговарајућих исказа о одговарајућем домену, на основу којих се касније могу извршити одређена резоновања користећи ове исказе и њихове градивне елементе

[133]. Искази се формирају на основу реализованих класа, међусобних веза (особина објеката) и атрибута (особина података) унутар саме онтологије, на основу којих се могу дефинисати појединачне инстанце и њихове специфичне улоге у оквиру посматраног домена [134], користећи описну логику (Description Logic) на којој се заснива онтологија и која представља технику представљања знања ниског нивоа [135]. Овде треба напоменути и чињеницу да се онтологије могу веома добро искористити приликом кодирања различитих типова података који ће даље бити коришћени у различитим врстама анализа [136].

Напред објашњена, реализована, онтологија омогућава повезивање неопходних знања из два домена, академског и софтверског, на адекватан начин како би се добила употребна знања везана за управљање софтвером у наставним процесима високошколског образовања кроз јасно и недвосмислено успостављену везу између предмета који се слуша у оквиру реализације наставног процеса и софтвера предвиђеног и неопходног за реализацију тог предмета. Предмет је јасно дефинисан унутар онтологије кроз одговарајуће односе, дефинисано је у којој се образовној установи реализује, у оквиру којих студија, модула и семестра, тако да можемо прецизно утврдити потребу студента који похађа одговарајући предмет за адекватним софтвером. С друге стране, сам софтвер је прецизно одређен дефинисаним атрибутима, који се, у складу са репрезентацијом софтвера у Linux окружењу, описују кроз његово име, епоху, верзију, издање, архитектуру и осталим атрибутима. У складу са процедурама инсталације Linux софтвера, софтвер се поставља у одговарајући однос са спремиштем у коме се налази и из којег се може добити на одговарајући начин, а тај начин се описује одговарајућим командама које се поново постављају, коришћењем адекватних веза, у однос са самим софтвером. Да би се избегле потенцијалне нежељене ситуације, у смислу покушаја инсталирања софтвера у неадекватном окружењу, дефинисани су одговарајући захтеви који морају бити испуњени приликом реализације инсталације самог софтвера. У том смислу дефинисани су минимални, оптимални и препоручени захтеви, при чему су минимални и оптимални захтеви дефинисани на основу препорука произвођача софтвера, док су препоручени захтеви дефинисани од стране особља образовне установе на основу њиховог непосредног корисничког искуства.

Међутим, интегрисано знање у оквиру предметне онтологије, само по себи не би имало посебан значај за заинтересоване стране ако му се не може приступити на адекватан начин и уколико се одговарајућа сврсисходна информација која је у датом тренутку потребна за реализацију одређене акције не може издвојити из мноштва ускладиштених информација у самој онтологији. У складу са претходним, извршена је имплементација посебног наменског софтвера чији је задатак анализа одговарајућих података садржаних у предметној онтологији на основу пружених улазних параметара и синтетисање одговарајућих употребљивих информација које ће бити даље обрађене и претворене у адекватно потребно знање из домена који се посматра. Анализа података садржаних у поменутој онтологији, за пружене одговарајуће улазе, врши се коришћењем имплементираног софтвера базираног на употреби SPARQL језика [137]. SPARQL представља најбољи избор у овом случају јер представља упитни језик за податке који омогућава даљу адекватну интеграцију у апликације [138] без обзира на то на ком програмском језику се та интеграција остварује.

Креирање одговарајућих SPARQL упита, елиминисање уочених грешака, као и тестирање реализованих упита за дефинисане улазне величине, обављено је коришћењем Snap SPARQL Query додатка верзије 6.0.0 у оквиру поменутог мултиплатформског Protégé десктоп коришћењем HermiT расуђивача који представља расуђивач базиран на употреби описне логике (Description Logic – DL reasoner) [139]. Рад поменутог расуђивача заснива се на употреби такозваних „hypertableau“ прорачуна [140], а битно је

споменути да и расуђивач у потпуности подржава рад са онтологијама реализованих употребом OWL синтаксе [141], као што је случај и са онтологијом развијеном за потребе предметне докторске дисертације. SPARQL упити, употребом HermiT расуђивача, омогућени су коришћењем посебног омотача (wrapper), који омогућава адекватан одговор на постављени упит [142]. Оно што треба поменути као врло велику предност приликом овакве употребе расуђивача приликом реализовања упита, јесте да HermiT расуђивач врши упозоравање на појаву некозистентних упита [143], што је од врло великог значаја за правилну реализацију и имплементацију предвиђеног софтверског решења. Коришћење расуђивача, у процесима који укључују рад са упитима, може поједноставити саме упите, а у неким случајевима долази до откривања информација које се не би могле добити у случајевима реализације упита без подршке одговарајућег расуђивача.

Пример реализованог SPARQL упита и добијених информација за одговарајуће улазне величине коришћењем HermiT расуђивача приказан је на слици 43.

```
prefix : <http://stolic.rs/onto/StudentSoftwareOntology#>

select ?faculty ?studies ?module ?course ?software ?commandform ?name ?epoch ?version ?release ?architecture
where
{
?facultystr a :Faculty .
?facultystr :hasInstitutionCode "TFB" .
?facultystr :hasLevelOfStudiesByInstitution ?studiesstr .
?studiesstr a :UndergraduateAcademic .
?studiesstr :hasStudyProgramName "Rudarsko inzenjerstvo" .
?studiesstr :hasModuleByLevelOfStudies ?modulestr .
?modulestr :hasModuleCode "RTOR".
?modulestr :hasCourseByModule ?coursestr .
?coursestr :hasCourseName "Procesna merna tehnika" .
?coursestr :hasRequiredSoftware ?softwarestr .
?softwarestr :useCommand ?command .
?command a :InstallationCommand .
?command :hasCommandForm ?commandstr.
?softwarestr :hasSoftwareName ?namestr .
optional {?softwarestr :hasSoftwareEpoch ?epochstr} .
?softwarestr :hasSoftwareVersion ?versionstr .
?softwarestr :hasSoftwareRelease ?releasestr .
?softwarestr :hasSoftwareArchitecture ?architecturestr .
bind(strafter(str(?facultystr),"#") as ?faculty)
bind(strafter(str(?studiesstr),"#") as ?studies)
bind(strafter(str(?modulestr),"#") as ?module)
bind(strafter(str(?coursestr),"#") as ?course)
bind(strafter(str(?softwarestr),"#") as ?software)
bind(str(?commandstr) as ?commandform)
bind(str(?namestr) as ?name)
bind(str(?epochstr) as ?epoch)
bind(str(?versionstr) as ?version)
bind(str(?releasestr) as ?release)
bind(str(?architecturestr) as ?architecture)
}
```

Execute

?faculty	?studies	?module	?course	?software	?commandform	?name	?epoch	?version	?release	?architecture
Technical_faculty_in_Bor	Rudarsko_inzenjerstvo	RTOR	Procesna_merna_tehnika	Octave	sudo dnf -y install	octave	6	6.4.0	5.fc36	x86_64
Technical_faculty_in_Bor	Rudarsko_inzenjerstvo	RTOR	Procesna_merna_tehnika	PSPP	sudo dnf -y install	pspp		1.6.2	4.fc36	x86_64

Слика 43. Пример извршења SPARQL упита коришћењем Snap SPARQL Query додатка у оквиру Protégé десктоп окружења

Као што је приказано у коду којим је представљен упит на претходној слици, види се употреба назива класа, веза и атрибута чија је дефиниција дата у претходним разматрањима. Међутим, када би се користили само ти називи, добио би се широк скуп информација које задовољавају исте и не би био добијен узак скуп информација који је стварно потребан у даљем процесу реализације одговарајућег задатка. Сходно томе процес инсталације софтвера који би се заснивао на тим информацијама не би био успешно реализован, односно или не би био уопште реализован, или би био реализован уз присуство одговарајућих грешака, или би се реализовао у целости, али би се поред жељених инсталација извршио и одговарајући скуп инсталација које нису биле

предвиђене јер реализацију заснивамо на поменутом широком скупу информација. Зато је потребно на одговарајући начин сузити избор на тачно одређене инстанце класа. У складу са тим, како би се тачно одредиле одговарајуће међузависности и извршила одговарајућа претрага података, а потом састављање података у одређену потребну информацију, користе се специфичне карактеристике SPARQL језика кроз реализацију његових упита. Уводе се одговарајуће променљиве, дефинишу се одговарајући услови и користе се одговарајуће SPARQL функције у циљу манипулисања подацима, који се налазе у предметној онтологији, на одговарајући начин. Поред наведеног, користе се и неки додатни механизми, како бисмо добили комплетан и ваљан скуп информација, попут врло битних механизама који омогућавају препознавање недостајућих вредности (missing values).

SPARQL упити су само један део система на коме се заснива софтверско решење. Циљ је да се понуђеним решењем омогући аутоматска интерпретација података и њихова синтеза у одговарајуће циљне команде које ће касније бити примењене над одговарајућом виртуелном машином која се користи од стране студента у остваривању постављених циљева наставних процеса у високошколском образовању. При томе се жели постићи минимизација интеракције са студентом као крајњим корисником предложеног решења. У складу са тим, не жели се да студент мора да сам, ручно, инсталира адекватно окружење, анализира и тумачи добијене податке и интегрише добијене информације у одговарајуће команде, већ се жели постојање независне апликације која ће то урадити аутоматски уз минималну интеракцију са самим студентом. У ту сврху, развијена је посебна апликација у програмском језику Python која омогућава интеграцију одговарајућих SPARQL упита и применљивост генерисаних резултата на одговарајућу Linux дистрибуцију у оквиру виртуелне машине.

Употреба програмског језика Python у проблемима чије решење подразумева коришћење принципа онтолошког инжењеринга је позната широј заједници [144] и већ постоје доказани примери коришћења у разним подручјима. Постоји већи број Python библиотека у којим је подржан развој апликација базираних на коришћењу онтологија. Једна од тих библиотека је и библиотека *owlready2* која је коришћена приликом развоја софтвера заснованог на употреби развијене онтологије у овој докторској дисертацији, пошто у потпуности подржава кључне карактеристике дефинисане у оквиру OWL2 синтаксе, као и коришћење наведених SPARQL упита [145]. Поменута библиотека је, између осталог, изабрана за имплементацију предметног софтвера услед чињенице да користи напред поменути HermiT расуђивач као подразумевани [146], па га у складу са тим није потребно експлицитно укључивати и вршити нека додатна подешавања. Тиме је омогућено лака имплементација претходно добијених SPARQL упита из Protégé развојног окружења, будући је HermiT расуђивач, такође, коришћен у Protégé окружењу приликом реализације поменутих упита. Овим је омогућена конзистентност добијених података, односно омогућено је да подаци добијени коришћењем онтологије у софтверу буду у потпуности идентични оним добијеним током развоја и тестирања SPARQL упита у већ поменутом окружењу.

Пример интеграције SPARQL упита приказаног на слици 43 у оквиру Python програмског језика коришћењем синтаксе коју пружа *owlready2* библиотека приказан је на слици 44.

```
from owlready2 import *
import os
ontologija="StudentSoftwareOntology.owx"
onto=get_ontology(ontologija).load()
Data=["\"TFB\"","\"Rudarsko inzenjerstvo\"","\"RTOR\"","\"Procesna merna tehnika\""]
Query="""
prefix : <http://stolic.rs/onto/StudentSoftwareOntology#>
select ?faculty ?studies ?module ?course ?software ?commandform ?name ?epoch ?version ?release ?architecture
where
{
?facultystr a :Faculty .
?facultystr :hasInstitutionCode """+Data[0]+""" .
?facultystr :hasLevelOfStudiesByInstitution ?studiesstr .
?studiesstr a :UndergraduateAcademic .
?studiesstr :hasStudyProgramName """+Data[1]+""" .
?studiesstr :hasModuleByLevelOfStudies ?modulestr .
?modulestr :hasModuleCode """+Data[2]+""" .
?modulestr :hasCourseByModule ?coursestr .
?coursestr :hasCourseName """+Data[3]+""" .
?coursestr :hasRequiredSoftware ?softwarestr .
?softwarestr :useCommand ?command .
?command a :InstallationCommand .
?command :hasCommandForm ?commandstr.
?softwarestr :hasSoftwareName ?namestr .
optional {?softwarestr :hasSoftwareEpoch ?epochstr} .
?softwarestr :hasSoftwareVersion ?versionstr .
?softwarestr :hasSoftwareRelease ?releasestr .
?softwarestr :hasSoftwareArchitecture ?architecturestr .
bind(STRAFTER(str(?facultystr),"#") as ?faculty)
bind(STRAFTER(str(?studiesstr),"#") as ?studies)
bind(STRAFTER(str(?modulestr),"#") as ?module)
bind(STRAFTER(str(?coursestr),"#") as ?course)
bind(STRAFTER(str(?softwarestr),"#") as ?software)
bind(str(?commandstr) as ?commandform)
bind(str(?namestr) as ?name)
bind(str(?epochstr) as ?epoch)
bind(str(?versionstr) as ?version)
bind(str(?releasestr) as ?release)
bind(str(?architecturestr) as ?architecture)
}
"""
collected=list(default_world.sparql(Query))
```

Слика 44. Пример интеграције SPARQL упита у Python програмски код

Као што је на претходној слици приказано, пре било каквог даљг рада, неопходно је да онтологија буде учитана на адекватан начин. Овде треба поменути чињеницу да онтологија може бити учитана на два различита начина у зависности од локације где се одговарајућа онтологија налази. У складу са тим онтологија се може користити са неке интернет локације навођењем комплетне адресе и путање до те онтологије, или локално, као што је овде случај, навођењем комплетне путање до саме онтологије.У наведеном примеру онтологија се налази на истој путањи као и одговарајућа извршна датотека у којој се налази Python програмски код, па је у оквиру одговарајуће промењљиве дато само пуно име датотеке (заједно са одговарајућом екстензијом) која садржи одговарајућу онтологију. Приступ одговарајућој онтологији извршен је позивом функције *get_ontology()* из *owlready2* библиотеке, а потом је њено учитавање извршено одговарајућом *load()* функцијом.

Улазни подаци се обезбеђују коришћењем листе адекватних елемената. За потребе приказа у датом примеру, елементи листе су дати експлицитно, док се у стварном раду ти подаци прикупљају из постојећег система високошколске установе за одговарајућег корисника, односно за студента и потом прослеђују софтверу у оквиру поменуте листе. У датом примеру приказаном на слици 44, за улазне податке дато је да студент студира на Техничком факултету у Бору (елемент листе „TFB“), студира студијски програм основних академских студија под називом Рударско инжењерство (елемент листе „Rudarsko inzenjerstvo“), модул Рециклажне технологије и одрживи развој (елемент листе

„RTOR“) и у оквиру поменутог модула предмет Процесна мерна техника (елемент листе „Procesna merna tehnika“) .

За тако добијене улазне податке, формира се одговарајући SPARQL упит смештен у одговарајућој текстуалној променљивој, која је у већини случајева због своје обимности реализована у више редова (променљива *Query* у примеру). Упит се извршава коришћењем *default_world.sparql()* функције из поменуте библиотеке *owlready2* и проналази податке о одговарајућем софтверу који се користи у оквиру предмета. Добијене информације из онтологије лако се обликују за даљи рад, пошто се генеришу у у оквиру адекватних листи коришћењем *list()* функције, па тако у суштини целокупно добијање података реализацијом упита добија облик *list(default_world.sparql(Query))* .

За сваку од пронађених ставки читају се одређени атрибути (назив софтвера, епоха, верзија, издање, архитектура) и на основу одговарајуће структуре команде која се користи за инсталирање софтвера на Linux заснованим системима, а која се такође чита приликом реализације упита над предметном онтологија (*commandform*), одговарајућа команда за инсталацију се формира за строго дефинисан софтвер. За претходни пример са слике 42 можемо видети да су за дефинисане критеријуме пронађена два софтвера: Octave и PSPP. За инсталацију Octave софтвера, реализовани софтвер зна да ће, на основу вредности прикупљених упитом из онтологије, извршити команду у виртуелној машини на следећи начин:

?commandform+” “+?name+”-“+?epoch+”:”+?version+”-“+?release+”.”+?architecture

Такође за инсталацију PSPP софтвера, реализовани софтвер зна да ће, на основу вредности прикупљених упитом из онтологије, извршити команду у виртуелној машини на следећи начин:

?commandform+” “+?name+”-“+?version+”-“+?release+”.”+?architecture

Као што се из наведеног види, имплементирани софтвер на адекватан начин препознаје информације добијене упитом из онтологије и на основу структуре добијених информација адекватно припрема одговарајућу команду која ће бити прослеђена виртуелној машини на реализовање у оквиру оперативног система уз успешно препознавање карактеристика Linux окружења које се користи унутар виртуелне машине. Такође, поређењем добијених команди за идентификоване софтвере који ће се инсталирати за дати предмет, примећује се да се недостајуће вредности успешно обрађују захваљујући уграђеним механизмима. Тиме се искључује могућност да корисник инсталира погрешан софтвер, примени погрешну команду, инсталира погрешну верзију, покуша да инсталира неадекватан софтвер у смислу архитектуре и слично.

Формиране команде прослеђују се оперативном систему виртуелне машине на извршење коришћењем одговарајућих системских команди Python програмског језика. Python поседује доста библиотека посвећених креирању решења у области системског инжењеринга. У конкретном случају, представљеном у докторској дисертацији, коришћена је *os* библиотека [147], јер пружа могућност имплементације добијених команди коришћењем одговарајућег shella који се налази у оквиру Linux дистрибуције на којој се заснива рад виртуелне машине. Претходно се лако постиже коришћењем *system()* функције у оквиру *os* библиотеке којој се као параметар прослеђује команда коју потом сама функција реализује у оквиру shella на циљаној Linux дистрибуцији. То омогућава да се знање синтетизовано на основу одговарајућих информација добијених

из предметне онтологије, а које се тиче одговарајућих софтверских задатака, везаних за остварење наставних процеса, адекватно пренесе у реални рад виртуелне машине.

О начину извршавању команди, односно њиховом прослеђивању виртуелној машини, биће детаљније речи касније, када ће се конкретније објашњавати сви механизми везани за рад саме виртуелне машине.

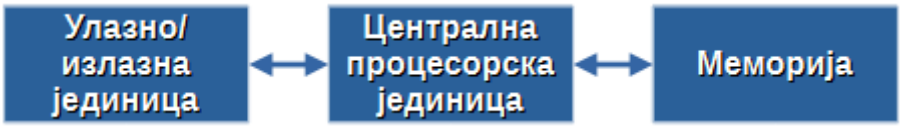
4. ХАРДВЕРСКЕ ОСНОВЕ ПРЕДЛОЖЕНОГ РЕШЕЊА

4.1. Идентификација узрочно-последичних веза

У претходном поглављу размотрени су софтверски аспекти предложеног решења које се описује у оквиру докторске дисертације. Међутим, добро је познато да у правилном раду било ког софтвера, поред правилног одабира истог, извршене инсталације на одговарајући начин, правилног конфигурисања и осталих активности, кључни значај има управо хардвер на коме ће се софтвер покретати. Неодговарајући или неисправан хардвер може спречити покретање софтвера, може довести до неправилног рада или испољити озбиљне недостатке по питању остварења добрих перформанси приликом коришћења софтвера. Ово важи, како за реалан хардвер, тако и за виртуелни хардвер на коме се заснива рад виртуелних машина, а које представљају основ предложеног решења које се разматра.

Када се разматрају претходно поменуте хардверске везе, разматрања се углавном базирају на појави неке грешке у функционисању самог хардвера и замене неисправне хардверске компоненте новом како би се систем довео у стање нормалне функционалности. То свакако може бити један од случајева, али није једини случај кога треба разматрати и искључити све остале узрочно-последичне везе које се јављају и које се могу јавити. Како би се правилно сагледао однос хардвера према функционисању софтвера на неком посматраном систему, сагледавање мора кренути од самих основа, односно мора се базирати на анализи понашања и утицаја хардвера заснованог на концептима рачунарске архитектуре.

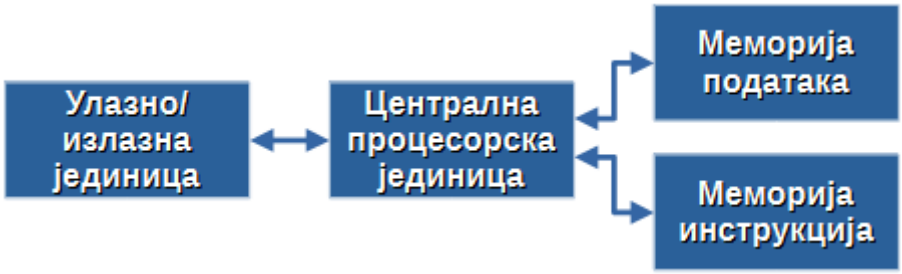
Овакви закључци нису новина у рачунарском свету и датирају од времена после другог светског рата, односно од неких кључних историјских момената и прекретница у развоју рачунара који су се дешавали у поменуто доба. Првобитни облици рачунарске архитектуре који су указивали на неке основне концепте који утичу на рад софтвера приказани су на слици 45, где је приказана Принстон рачунарска архитектура.



Слика 45. Принстон (Фон Нојманова) архитектура рачунара

Најзаслужнији за развој модела који описује рачунарску архитектуру, а који је развијан у оквиру програма на универзитету Принстон, јесте амерички математичар мађарског порекла Џон фон Нојман (John von Neumann) па се зато овакав концепт чешће назива Фон Нојманова архитектура рачунара уместо Принстон архитектура рачунара. Концепт архитектуре дигиталног рачунара, по фон Нојману, без улажења у детаље модела и користећи одређени ниво апстракције, заснивао се на повезаности централне процесорске јединице, меморије и улазно/излазе јединице. При томе се овде мора нагласити да овај модел суштински представља рачунар са одређеним програмом, односно софтвером, код којег није могуће истовремено извршење преузимања инструкција и операција над подацима услед коришћења заједничке магистрале и њеног дељења међу собом [148].

Ово је касније унапређено појавом новијег модела рачунарске архитектуре који је познат под називом Харвард рачунарска архитектура и који је приказан шематски на слици 46.

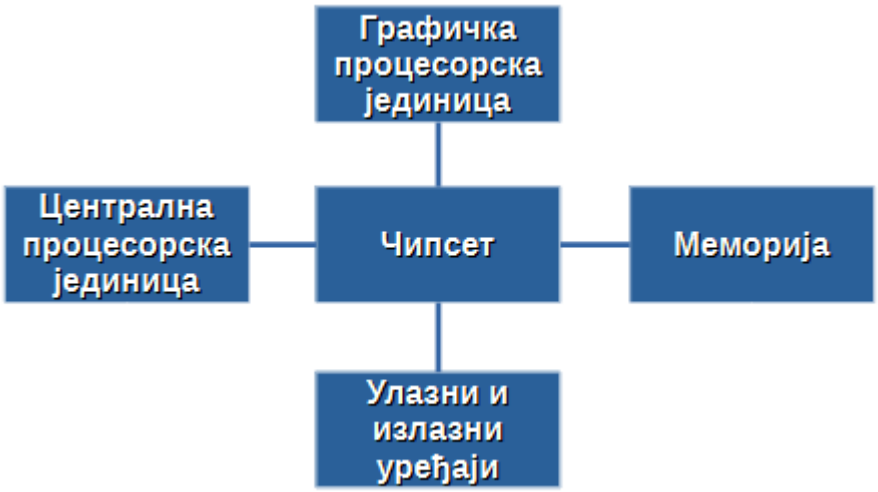


Слика 46. Харвард архитектура рачунара

Као што се са слике види, задржавају се основни елементи архитектуре рачунара, који су били присутни и у фон Најмановом моделу, али сада у нешто другачијем облику. Највеће унапређење у односу на претходни модел рачунарске архитектуре дешава се у меморијском делу, будући да Харвард архтектура рачунара дефинише два меморијска простора који су међусобно одвојени, један намењен искључиво програмском коду и други намењен искључиво подацима. Претходно подразумева и увођење засебних адресних магистрала и засебних магистрала података чиме се отвара простор и за увођење паралелизама у преузимању и извршавању [149].

Значај претходно поменутих модела у погледу дефинисања рачунарске архитектуре јесте да дефинишу извршење софтверских задатака кроз три кључне компоненте: централне процесорске јединице, меморије и улазно-излазних уређаја, односно ако се сада врши посматрање са становишта хардвера, те компоненте постају могући извор одређених узрока који ће дефинисати одређена последична стања над самим софтвером као и над задацима које он спроводи. Ове три компоненте срећемо и у данашњој модерној рачунарској архитектури, тако да је значај и утицај претходних модела видљив и у најсавременијим разматрањима рачунарске архитектуре.

Рачунарска архитектура која би адекватно описала већину данашњих рачунара приказана је на слици 47. Овде треба напоменути да је у самом приказу коришћен највиши ниво апстракције. Будући да је овакав приказ довољан за даља разматрања која се тичу виртуелних машина и виртуелног хардвера, неће се дубље залазити у детаље и спуштати на ниже нивое хардверских имплементација и разматрања.



-

Слика 47. Архитектура рачунара приказана на високом нивоу апстракције

Као што се из представљене архитектуре види, у односу на претходне моделе сада је додат чипсет и графичка процесорска јединица као додатне целине у оквиру рачунарске

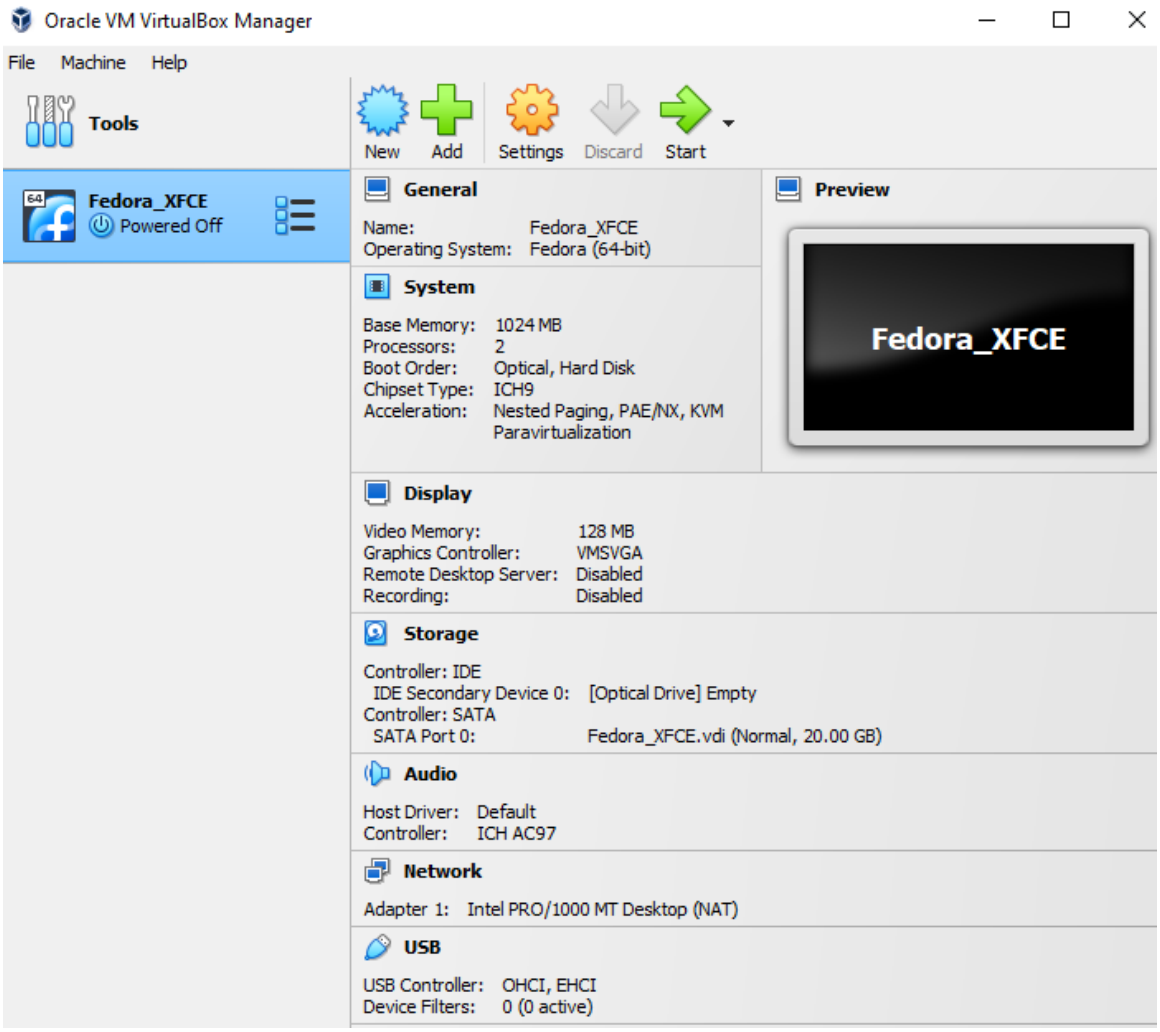
архитектуре како би било могуће да се, заједно са централном процесорском јединицом, меморијом и улазно-излазним уређајима, оствари приказ адекватне организације компоненти које чине рачунарски систем и одговарајућа семантика или значење операција које реализују његову функцију у складу са самом дефиницијом рачунарске архитектуре [150]. Преко чипсета (chipset) се остварује повезаност свих осталих компоненти у приказаној архитектури, будући да се у оквиру чипсета налазе одговарајући мостови (bridges) који омогућавају повезивање различитих магистрала (buses), односно једне врсте са другом [151]. Поменућемо само да постоје два главна моста у оквиру чипсета: северни мост (North Bridge – NB или негде означен и као MCH – Memory Controller Hub) који остварује одређена повезивања компоненти са меморијом и (South Bridge – SB или негде означен и као ICH – I/O Controller Hub) који остварује одређена повезивања са улазно-излазним јединицама.

Овако представљена рачунарска архитектура кључна је за тумачења која ће бити даље изнета у циљу дефинисања одређених узрочно-последичних веза која се јављају приликом реализације и коришћења виртуелних машина и самог виртуелног хардвера. Ако би се реализовао приказ архитектуре виртуелне машине заснован на употреби виртуелног хардвера и ако би при томе био коришћен висок степен апстракције био би добијен модел који има многа преклапања са претходно приказаним моделом. Постоји вероватноћа да би, на неком највишем нивоу апстракције, били добијени, у појединим случајевима, скоро идентични модели архитектура. Зато се у складу са претходним, у даљим разматрањима врши базирање управо на компонентама претходно описаног модела рачунарске архитектуре.

На слици 48 приказани су детаљи виртуелног хардвера једне виртуелне машине која је реализована у оквиру Oracle VirtualBox хипервизора верзије 7.10 који, као што је већ претходно наглашено, представља хипервизор типа 2. Са слике се уочавају одређени елементи у складу са архитектуром приказаном на слици 47. У оквиру секције System уочава се део конфигурације виртуелне меморије (на слици под ознаком Base Memory), део конфигурације виртуелне централне процесорске јединице (на слици под ознаком Processors), као и део конфигурације виртуелног чипсета (на слици под ознаком Chipset Type). У наредној секцији означеној као Display, дати су делови конфигурације који се односе на виртуелну графичку процесорску јединицу. Прикази остварени у осталим секцијама које следе могу се сврстати у конфигурисања различитих виртуелних улазно-излазних уређаја. Наравно, овде треба поменути да је ово само један информативни приказ детаља виртуелног хардвера који чине виртуелну машину, а да се подешавање саме виртуелне машине остварује на много више нивоа у односу на оно што је приказано у склопу информативног приказа.

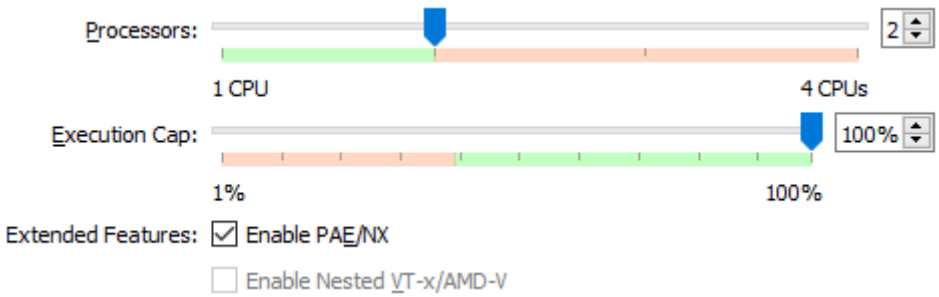
Све напред наведено неопходно је како би се могла направити адекватна анализа о томе који су хардверски елементи битни за посматрање понашања физичког система, али виртуелне машине током остваривања задатака намењених остваривању адекватних наставних процеса у оквиру високошколског образовања. Ове анализе су јако битне јер ће на основу одређених посматрања бити дефинисане узрочно-последичне везе које се остварују између хардверских елемената физичког система, односно реалног хардвера и хардверских елеманата виртуелне машине, односно виртуелног хардвера. Сама важност доброг дефинисања узрочно-последичних веза лежи у чињеници да када се на адекватан начин спозна њихова појава и начин функционсања, тада се могу саградити адекватни механизми који ће минимизирати ефекте који настају њиховом појавом, односно механизме које ће омогућити прилагођавање виртуелне машине на њихову егзистенцију и тиме побољшати услове рада студентима који исту буду користили за остваривање предвиђених исхода учења. Сва разматрања у наставку реализоваће се у односу на имплементације виртуелних машина у оквиру VirtualBox хипервизора верзије 7.10, при

чему ће се терминолошки користити одреднице дате у званичном корисничком упутству за наведену верзију хипервизора [152].



Слика 48. Приказ детаља виртуелног хардвера реализоване виртуелне машине у оквиру одговарајућег хипервизора

У анализи кренућемо редом, користећи усвојену архитектуру са слике 47, па ће се прво узети у разматрање централна процесорска јединица и њене особености које треба посматрати на релацији физички систем – виртуелна машина. Својства виртуелне централне процесорске јединице (процесора) у оквиру дефинисаности виртуелне машине приказана су на слици 49.



Слика 49. Приказ могућих подешавања централне процесорске јединице (процесора) у оквиру конфигурације виртуелне машине

Прва опција на горњој слици означена је као Processors и означава број виртуелних језгара централне процесорске јединице које ће виртуелна машина бити у могућности да препозна и користи. Максимални број виртуелних језгара који се може доделити одговара броју логичких језгара које поседује централна процесорска јединица физичког рачунара. Препоручљиво је да се број виртуелних језгара изабере тако да одговара броју физичких језгара централне процесорске јединице која се користи на стварном рачунару.

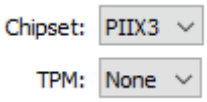
Следећа опција означена је као Execution Cap и представља максималну количину времена која се може доделити стварној централној процесорској јединици приликом емулације виртуелне централне процесорске јединице. Подразумевана вредност износи 100% и у већини случајева ову вредност не треба мењати будући да значајно утиче на целокупне перформансе приликом рада на виртуелној машини. Приликом тестирања уочено је да промене ових вредности чак и у моментима врло великог оптерећења рада физичке централне процесорске јединице нису значајније редуковале оптерећење физичког процесора, али су довеле до значајнијег редуковања преформанси виртуелне машине чак и приликом додељивања вредности у распону од 50 до 90 процената.

Опција Enable PAE/NX омогућава доступност PAE и NX могућности физичке централне процесорске јединице виртуелним ресурсима. Подразумевана вредност ове опције је да је укључена и ту вредност не треба мењати у случају коришћења виртуелних машина заснованих на Linux окружењу будући да постоје дистрибуције које у случају да ова опција буде искључена неће моћи радити у оквиру виртуелних машина.

Последња опција омогућава повезивање одговарајуће Intelове VT-x или AMDове хардверске виртуелизације са функцијама виртуелне машине. Ова опција је искључена по подразумеваној вредности и такво стање треба задржати, будући да може довести до потребе за додатним напредним подешавањима које могу бити изазов чак и за искуснije кориснике.

Као што се из претходних разматрања види, сва подешавања везана за централну процесорску јединицу довољно је дефинисати само једном, приликом креирања виртуелне машине. Наведене опције не испољавају високи степен променљивости и нема потребе за адаптацијом на исте, тако да се подешене вредности приликом првог стартовања виртуелне машине на личном рачунару студената касније неће мењати. У складу са тим ове опције нису доминантне у реализацији узрочно-последичних веза између стварног и виртуелног хардвера.

Опције везане за подешавања виртуелног чипсета су следеће које ће бити размотрене, а чија су најзначајнија својства приказана у оквиру слике 50.



Слика 50. Приказ могућих подешавања чипсета у оквиру конфигурације виртуелне машине

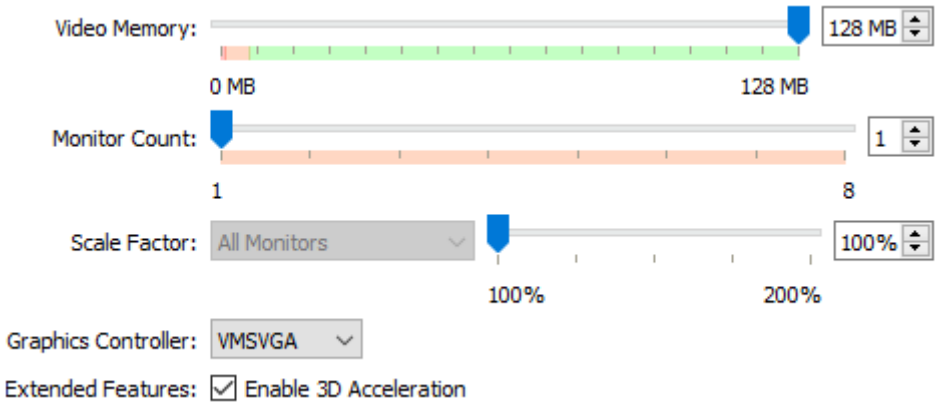
У опцији Chipset врши се избор одговарајућег виртуелног чипсета. На располагању су две могућности, избор PIIX3 или ICH9 чипсета. Овде треба напоменути да поред тога што је ICH9 чипсет већ дуже време присутан за избор приликом конфигурисања виртуелних машина, по званичној документацији се његов статус још увек води као експерименталан, па иако нуди веће могућности, још увек се не препоручује као примаран избор осим у случајевима кад не постоји алтернатива. Такви су на пример случајеви појединих виртуелних машина заснованих на macOS оперативним системима, случајеви када виртуелна машина користи више од осам виртуелних мрежних адаптера

и још неки случајеви који захтевају да се у оквиру виртуелне машине изврши конфигурација значајнијег броја виртуелних ресурса. У случају предметне докторске дисертације користиће се подразумевани PIIX3 чипсет јер нуди сасвим довољно могућности у складу са захтевима који се пред њега постављају у смислу реализације виртуелне машине за реализацију задатака у домену остварења наставних процеса у високошколском образовању. Подразумевана вредност конфигурисаће се само приликом креирања виртуелне машине и као таква ће после бити непроменљива.

Следећа опција је намењена остваривању повезивања са посебним безбедносним чипом на физичком хардверу рачунара који омогућава постизање такозваних TPM (Trusted Platform Module) функционалности и у оквиру виртуелне машине. Поједини савремени оперативни системи захтевају ове функционалности како би могли уопште бити покренути на одређеном хардверу. Linux дистрибуција која се користи у сврху остваривања виртуелне машине у оквиру ове докторске дисертације не захтева присуство наведених функционалности, па ће ова опција бити искључена што је и подразумевана вредност и као таква биће у потпуности надаље непроменљива.

Као што се види из анализираног и овде ће важити слични закључци изведени у погледу коришћења виртуелне централне процесорске јединице, односно не постоји променљивост параметара приликом коришћења виртуелне машине, па сходно томе нема потребе за додатном адаптацијом на исте, већ ће се користе у оном облику како су дефинисани приликом креирања. Опције везане за виртуелни чипсет, такође, нису доминантне у реализацији узрочно-последичних веза између стварног и виртуелног хардвера.

Следећи део који ћемо размотрити, у складу са напред наведеном архитектуром, јесу опције везане за виртуелизацију графичке процесорске јединице, односно графичког подсистема виртуелне машине, а које су приказане на слици 51.



Слика 51. Приказ могућих подешавања графичке процесорске јединице у оквиру конфигурације виртуелне машине

Video Memory опција одређује колико ће се меморије доделити виртуелној графичкој карти која се користи у оквиру реализације виртуелне машине, при чему су вредности изражене у мегабајтима (MB). Треба напоменути, да не буде забуне, да се меморија која се додељује овде не додељује од меморије која се налази у оквиру физичке графичке карте на рачунару (у већини случајева означена као GDDR), већ се додела врши од стране стандардне меморије рачунара (у већини случајева означена као DDR). Приликом иницијализације виртуелне машине овде је препоручљиво увек ићи на максималне вредности. Дозвољена максимална вредност приликом тестирања у склопу докторске дисертације је обично била декларисана на 128 MB, али је било случајева када је максимална вредност била декларисана и на 256 MB. У сваком случају ово нису велике

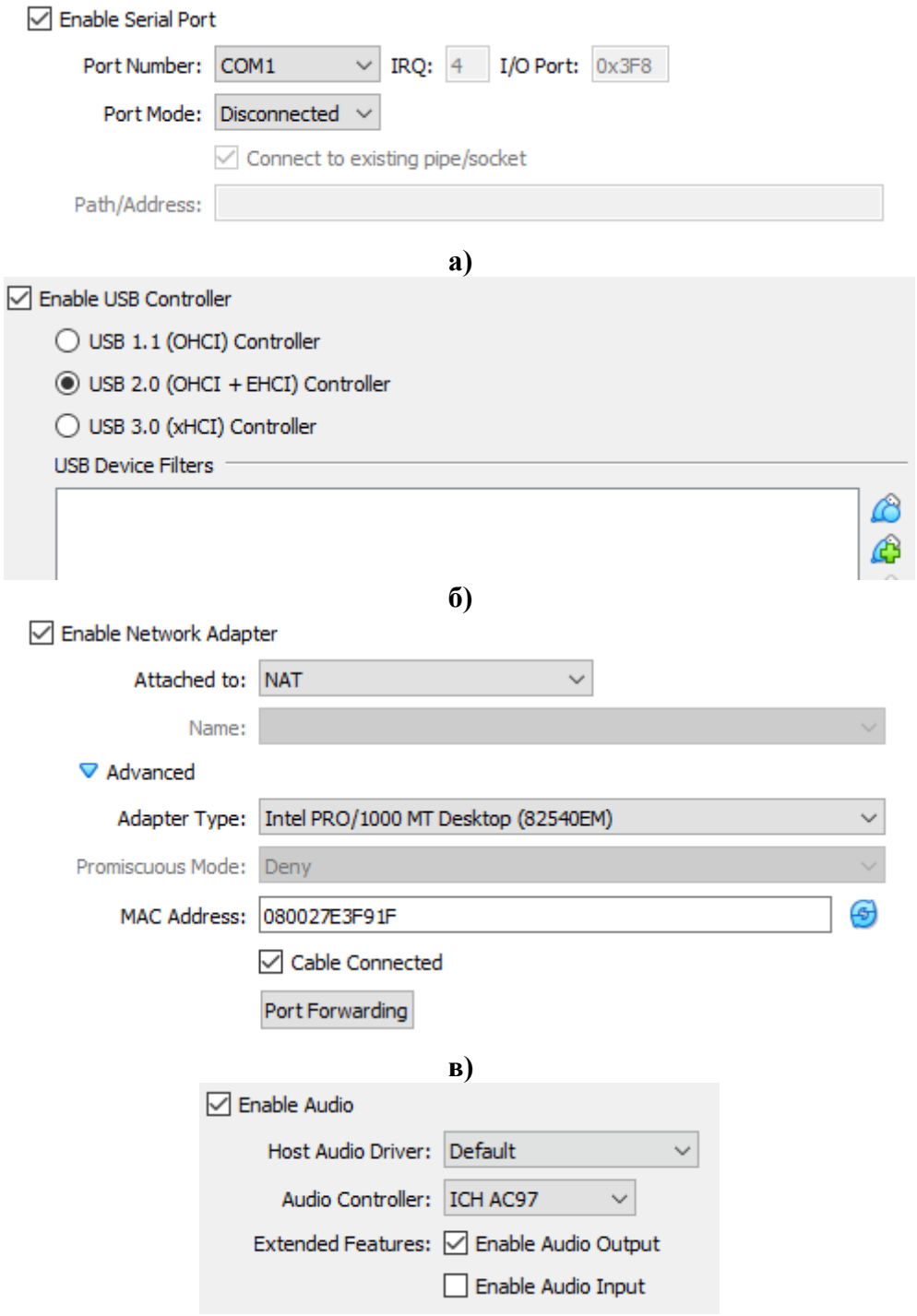
вредности и велика оптерећења за количину расположиве меморије физичког рачунара, па у складу са тим, овде ће се увек примењивати максималне вредности које ће бити константне током рада виртуелне машине. Непроменљивост додељене количине меморије је одлука која је у складу са чињеницом да би се и у режимима великог меморијског оптерећења физичког рачунара, променом додељених вредности не би добило значајније растерећење физичког рачунара, али би се зато значајно деградирале перформансе виртуелног графичког подсистема који користи виртуелна машина, што имплицира да би дошло до деградирања перформанси и саме виртуелне машине и самим тим би дошло до повећања ризика појаве нестабилности у раду. У складу са тим приликом реализације виртуелне машине додељује се максимална могућа вредност посматране меморије и она се у даљем процесу не мења ни под каквим условима.

Следећа опција која се узима у разматрање јесте Graphics Controller у коме се врши избор виртуелног графичког контролера који ће бити коришћен у раду виртуелне машине. Постоји неколико могућности које се могу имплементирати приликом рада виртуелне машине у зависности од оперативног система на коме ће рад бити заснован. Будући да се у овом случају користи одговарајућа Linux дистрибуција, у складу са званичном документацијом која покрива конфигурисање хипервизора, користиће се VMSVGA виртуелни адаптер који се препоручује као подразумевани за Linux засноване виртуелне машине. Избор виртуелног графичког контролера реализује се приликом креирања виртуелне машине и остаје непроменљив током целокупног рада. Овде треба напоменути да коришћење поменутог виртуелног графичког контролера у пуној снази и са свим својствима није могуће без инсталације додатног софтвера под називом Guest Additions, који представља одређене додатке од стране произвођача хипервизора који се инсталирају у оквиру оперативносг система виртуелне машине. Ови додаци обухватају одређене апликације и управљачке програме који врше одређене оптимизације система, побољшавају функционалност и додају додатне перформансе самом систему. Инсталирају се након инсталирања оперативног система у оквиру виртуелне машине.

Опције Monitor Count и Scale Factor неће се посебно разматрати будући да нису од значаја за предметну употребу виртуелне машине у области дефинисаној овом докторском дисертацијом. Ове опције остављају се на подразумеване вредности као што је приказано на слици и биће непроменљиве током рада виртуелне машине.

Што се тиче додатне опције означене као Enable 3D Acceleration на слици је ова опција означена само како би се указало да је поменути изабрани виртуелни графички контролер у стању да подржи рад са укљученом опцијом уколико за њом постоји потреба. Ова опција не може радити без инсталираног поменутог додатног софтвера Guest Additions. Уколико поменути софтвер није правилно инсталиран може доћи до нестабилности виртуелне машине, а у појединим моментима може доћи и до потпуне немогућности стартовања. У складу са тим препорука је да се виртуелна машина користи без ове опције када год услови то дозвољавају, осим у случајевима када сам рад са софтвером у оквиру виртуелне машине захтева 3D акцелерацију и када та опција мора бити укључена како би се омогућила функционалност софтвера који се користи. У оба случаја, користили опцију или не, коришћење се дефинише приликом процеса креирања виртуелне машине, а потом се сматра да нема променљивости по питању коришћења ове опције.

У складу са свиме горе наведеним и у овом случају постоји непроменљивост параметара приликом коришћења виртуелне машине, па неће бити додатних адаптација, односно немамо доминантних узрочно-последичних веза које морамо додатно разматрати како бисмо утицали на стабилност веза остварених између стварног и виртуелног хардвера.

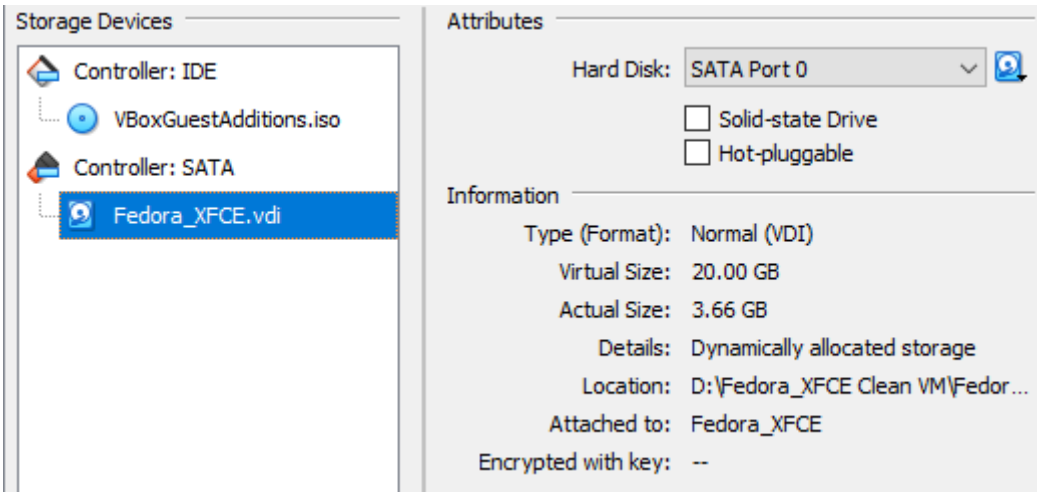


Слика 52. Приказ могућих подешавања различитог виртуелног улазно-излазног хардвера у оквиру конфигурације виртуелне машине: а) виртуелни серијски портови, б) виртуелни USB портови, в) виртуелни мрежни адаптер, г) виртуелна звучна карта

У складу са напред описаном архитектуром, треба размотрити и постојање виртуелног улазно-излазног хардвера, међутим разматрањем свих виртуелних улазно-излазних уређаја и могућностима њиховог конфигурисања би се значајно продубила дискусија без неког већег значаја за даљи рад на савладавању проблема описаног у докторској дисертацији. Зато ћемо само представити заједнички именитељ целој групи

ових уређаја а то је, као и у претходним случајевима, њихова непроменљивост током коришћена виртуелне машине. То значи да ће се ови уређаји конфигурисати приликом креирања виртуелне машине или непосредно пре првог пуштања виртуелне машине у рад на личном рачунару студента и неће накнадно захтевати било какву адаптацију приликом коришћења виртуелне машине за реализацију предвиђених наставних активности у домену високошколског образовања. Такође и овде, као и у претходно описаним случајевима, не постоје доминантне узрочно-последичне везе које треба додатно анализирати. На слици 52 су приказани неки од виртуелних улазно-излазних уређаја са одређеним скупом својстава.

Као једини изузетак од овог тумачења можемо навести случај уређаја за смештање података, односно дискова, било да се ради о стварним или виртуелним компонентама система, будући да се и они сврставају на одређени начин у групу улазно-излазних уређаја у складу са приказаном архитектуром. Карактеристично за ове уређаје јесте да они, за разлику од других уређаја из претходно поменуте групе, ипак поседују нека својства која испољавају велику променљивост оком рада виртуелне машине. Управо једно такво својство јесте промена преосталог слободног простора на диску која испољава изразиту временску променљивост, односно постојаће различита стања за различите временске тренутке или интервале посматрања. Нека од својстава виртуелних дискова која се испољавају у оквиру виртуелне машине приказана су на слици 53.



Слика 53. Приказ могућих подешавања виртуелних дискова у оквиру конфигурације виртуелне машине

Без обзира на одређене уочене временске променљивости и без обзира на могућност идентификовања врло изражене узрочно-последичне везе, ове уређаје ћемо ипак дефинисати као непроменљиве током коришћења виртуелне машине. Образложење ове недоследности лежи у самим својствима виртуелних дискова. Виртуелне дискове можемо имплементирати у статичком и динамичком смислу. Приликом креирања виртуелног диска задаје се његов капацитет. Уколико се врши статичка имплементација, онда се на физичком диску резервише комплетан простор једнак капацитету виртуелног диска, док се у динамичкој имплементацији не врши резервација простора, већ се простор на физичком диску попуњава пропорционално заузећу капацитета виртуелног диска. У динамичкој имплементацији, на физичком диску заузети простор се не смањује уколико дође до смањења заузетог простора на виртуелном диску, он је увек на максималној вредности заузећа виртуелног диска која је забележена током рада виртуелне машине. У оба случаја капацитет виртуелног диска је непроменљив, односно капацитет ће бити оне величине која је дефинисана приликом креирања виртуелног

диска у процесу реализације виртуелне машине. Овде треба напоменути да постоје поступци промене капацитета виртуелног диска, али се они не примењују будући да не спадају у стандардне предвиђене поступке, а и пракса је показала да се у већини случајева приликом промене капацитета виртуелног диска врши директан утицај на конзистентност саме виртуелне машине. Дакле, без обзира на присуство ових поступака, узимаће се да је капацитет виртуелног диска непроменљив. У складу са том непроменљивошћу нећемо имати било какву адаптацију у погледу капацитета виртуелног диска, па важи тврдња изнета на почетку разматрања о непроменљивости ове категорије улазно-излазних уређаја. О неким посебним проверама које се морају периодично вршити услед смањења простора на физичком диску биће речи накнадно.

Насупрот виртуелним дисковима, капацитет виртуелне радне меморије је променљив у оквиру реализоване виртуелне машине као што је показано на слици 54 и овде се јавља доминантна узрочно-последична веза, па у складу са њеним постојањем треба обезбедити одговарајуће механизме за умањење њеног утицаја како на саму виртуелну машину, тако и на физички систем у оквиру ког се она извршава.

Слика 54. Приказ подешавања капацитета виртуелне радне меморије у оквиру конфигурације виртуелне машине

Приликом рада виртуелне машине укупна расположива меморија у оквиру физичког система биће умањена за вредност додељене меморије виртуелној машини која се састоји од додељене виртуелне радне меморије и додељене виртуелне графичке (видео) меморије. То практично значи да може доћи до значајног умањења расположиве радне меморије у оквиру физичког система. С друге стране сам корисник у оквиру физичког система не користи само виртуелну машину, па радна меморија неће бити оптерећена само од стране виртуелне машине, него и од целокупног софтвера који се у моменту датог посматрања извршава на физичком рачунару и користи меморијске ресурсе. Чак и ако корисник након подизања оперативног система у оквиру физичког рачунара не би покренуо ни једну корисничку апликацију осим виртуелне машине, опет може доћи до проблема приликом коришћења меморијских ресурса јер виртуелна машина конзумира одређени део меморије, а поред тога конзумира и оперативни систем физичког рачунара и његови разни процеси, као и пратеће апликације у виду антивируса, разних додатака, системских апликација и сличног. У зависности од расположиве радне меморије на рачунару и конзумирања исте може доћи до разних сценарија, почев од тога да машина има довољно меморијских ресурса и ради у једном стандардном нормалном режиму, па до оног крајњег сценарија у коме су сви меморијски ресурси искоришћени до максимума и рачунар улази у домен нестабилног рада који може резултирати разним непредвидим последицама (губитак података, пад система и слично).

С друге стране, ако посматрамо виртуелну машину, у већини случајева у циљу смањења великих утицаја на физичке меморијске ресурсе, корисници смањују количину виртуелне радне меморије којом виртуелна машина располаже. Ово може дати одређене позитивне ефекте, али у већини случајева овакво решење проузрокује нови низ проблема, овог пута у домену виртуелне машине. У већини случајева доноси се неинформисана одлука која није базирана на анализама и утврђивању чињеничних стања, већ се редукција количине виртуелне радне меморије ради насумичном методом где су корисници превасходно фокусирани на минимизацију ефеката везаних за физички рачунар, а мање или у појединим случајевима у потпуности не обраћају пажње на ефекте

проузроковане на самој виртуелној машини. Најчешће долази до тога да се количина виртуелне радне меморије смањи нерационално и постане недовољна и проузрокује „уско грло“ на виртуелној машини и тада се дешавају сценарији идентични поменутим у оквиру дискусије везане за сам физички рачунар, односно може се десити да виртуелна машина настави да ради у стандардном режиму, а може се десити да дође до значајног умањења перформанси виртуелне машине, као и до најгорег могућег сценарија који подразумева отежано функционисање софтвера, губитак података, пада система.

Као што се види из приложеног постоји јасна, изражена узрочно-последична веза, па у складу са тим датом проблему се не може приступити на начин „или-или“, односно не сме се занемаривати ни једна страна у целом проблему, већ се мора трагати за целовитим решењем које истовремено не угрожава нормално функционисање физичког система, односно рачунара, али и саме виртуелне машине. Зато је ово ситуација у којој се мора обезбедити адекватно праћење, анализа, синтеза добијених разултата, а потом на основу тога реализовати и одговарајући адаптивни механизми.

У складу са напред наведеним у наставку разматрања фокус ће бити на опису решења које се примењује у домену меморијских ресурса, односно приликом истовременог паралелног коришћења физичких ресурса и оних виртуелних. Решење се заснива на отклањању ових проблема јер су упрао они доминантни извор већине проблема приликом примене виртуелних машина у реализацији задатака у оквиру наставних процеса који се реализују у високошколском образовању.

4.2. Иницијализација виртуелног хардвера

Креирање основне виртуелне машине врши одговарајуће овлашћено лице из високошколске институције, које поседује одговарајућа неопходна знања за спровођење овог поступка и које може бити из редова ненаставног (на пример лабораторијски или информатички техничар, систем инжењер и слично) или наставног особља (на пример предметни сарадник у настави или асистент). Приликом креирања основне виртуелне машине дефинише се виртуелни хардвер који ће користити виртуелна машина и постављају се почетне вредности разних својстава тог виртуелног хардвера на неке подразумеване вредности довољне за почетни рад са виртуелном машином. Тај почетни рад подразумева инсталирање оперативног система на виртуелној машини, који је у случају ове докторске дисертације нека од верзија Fedora Linux дистрибуције, као и инсталирање Guest Addition софтвера, захтеваног од стране хипервизора, након инсталације оперативног система. Након овако формиране виртуелне машине, иста је спремна за даљу дистрибуцију за коришћењем од стране крајњих корисника, односно студената.

Поред ове могућности, која ће у докторској дисертацији бити подразумевана, постоји могућност преузимања готове виртуелне машине и из неког другог извора. На пример постоје виртуелне машине намењене за коришћење у оквиру VirtualBox хипервизора које се нуде за бесплатно преузимање од стране произвођача оперативног система које се могу, такође, успешно користити за задатке описане у овој докторској дисертацији. Поред ових, постоје виртуелне машине и неких других произвођача софтвера, као и виртуелне машине креиране од треће стране, али ове виртуелне машине није препоручљиво користити за задатке описане у докторској дисертацији из разлога што не поседујемо довољно информација о самој виртуелној машини (који је све софтвер инсталиран, изабрани параметри при инсталацији и неке друге информације) па ове виртуелне машине могу довести до испољавања понашања различитог од предвиђеног, а у неким ситуацијама може крајњег корисника ставити и у поприлично незгодан

положај (кршење лиценцних права, прикупљање података без знања корисника и слично).

Најбољу информисаност и највећи степен сигурности крајњи корисник, односно у овом случају студент, постиже коришћењем виртуелне машине обезбеђене од стране одговарајуће високошколске институције. У складу са тим, у овој докторској дисертацији ће се подразумевати да се процес формирања и обезбеђења виртуелне машине за студента у целости обавља од стране високошколске установе као легитиман процес у оквиру пратећих активности који обезбеђују квалитет самог наставног процеса.

Међутим, виртуелна машина ће се користити од стране различитих студената, на различитим рачунарима, односно виртуелна машина је намењена да буде имплементирана у изразито хетерогеном рачунарском окружењу. При томе се жели постићи да виртуелна машина на сваком од рачунара, на ком ће бити имплементирана у оквиру одговарајућег хипервизора, постиже најбоље перформансе, а то значи да се постигне најбоља могућа оптимизованост параметара виртуелне машине. Ти параметри виртуелне машине су суштински одређена својства виртуелног хардвера који чини основу виртуелне машине. Зато, пре првог стартовања виртуелне машине, на крајњем одредишту, односно на личним рачунарима студената, треба извршити одговарајуће прилагођење параметара виртуелне машине на услове у којим ће она бити коришћена, односно треба прилагодити виртуелни хардвер новонасталим условима. Овај поступак називамо иницијализацијом виртуелног хардвера и спроводимо га на крајњем одредишту.

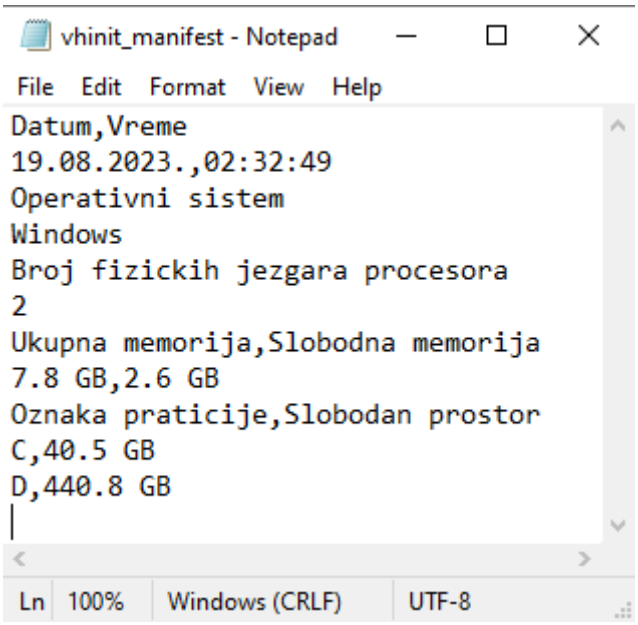
Иницијализација виртуелног хардвера може бити примарна и секундарна. Примарна иницијализација виртуелног хардвера врши се након преузимања виртуелне машине од стране високошколске институције и пре првог пуштања у рад на одредишном рачунару како би се виртуелна машина акомодирала на постојеће услове физичког хардвера који су затечени у датом тренутку на одредишном рачунару. Секундарна иницијализација спроводи се након првог пуштања у рад у било ком временском тренутку када се поново захтева прилагођење виртуелне машине на услове околине дефинисане од стране рачунара студента на ком се виртуелна машина покреће. Поновно захтевање прилагођења виртуелне машине на услове околине настаје када дође до неке промене услова који су владали у моменту претходне иницијализације. Та променљивост може бити условљена заменом неке хардверске компоненте, њеним сервисирањем, привременом недоступношћу неке компоненте услед њеног отказа и било ком другом ситуацијом којом је настала значајнија промена окружења. Поступак иницијализације је идентичан, без обзира да ли се ради о примарној или секундарној инсталацији и он ће бити описан у наредним редовима.

Програм за иницијализацију, настао у оквиру докторске тезе, написан је у програмском језику Python и реализоване су верзије за Windows, Linux и macOS оперативне системе чиме су покривени најпознатији оперативни системи и чиме је омогућено да се сам поступак обави на сваком рачунару који користи студент без обзира на употребљени оперативни систем. Одговарајуће верзије су преведене адекватно за сваки од наведених оперативних система и у свим оперативним системима извршна датотека је дата под именом *vhinit* (скраћеница од енглеског Virtual Hardware INITalizaion). По покретању, програм прикупља основне податке о тренутном статусу физичког хардвера и инсталираној верзији оперативног система. Обим података који се прикупља са стварног система је следећи:

- врста инсталираног оперативног система,
- број физичких језгара централне процесорске јединице,
- укупна количина инсталиране радне меморије,

- тренутна количина слободне радне меморије и
- тренутна количина слободног простора на партицијама.

Прикупљени подаци се снимају у оквиру текстуалне датотеке *vhinit_manifest*, дате у CSV формату,како би корисник имао увид у податке који су прикупљени и како би у случају настанка неког проблема та датотека могла бити касније искоришћена у сврху анализе и отклањања истог. Пример једне реализоване *vhinit_manifest* датотеке дат је на слици 55.



Слика 55. Пример датотеке *vhinit_manifest*

Команде за прикупљање података о тренутном стању физичког хадвера су детаљније објашњене у другом поглављу приликом дискусије о хетерогености рачунарског окружења, па ће се на овом месту дати сажетији приказ како се не би вршило понављање појединих појашњења. Наравно, треба напоменути да се у односу на прикупљање података о физичком систему у оквиру другог поглавља докторске дисертације, овде прикупља један далеко ужи сет података, будући да је сет података специјализован за подешавање кључних параметара виртуелног хардвера.

За идентификацију оперативног система употребљена је функција *uname()* која се налази у оквиру библиотеке *platform*. Команда враћа листу која садржи мноштво информација о самом оперативном систему. Међутим, за само подешавање виртуелног хардвера битно је само знати који је оперативни систем у питању, без дубље анализе његових осталих карактеристика. Зато се чита само први елемент наведене листе и у зависности од његове вредности предузимају одређене акције везане за виртуелни хардвер на оним местима где је виртуелни хардвер директно зависан од оперативног система инсталираним над физичким хардвером.

Број физичких језгара централне процесорске јединице у случају Windows и macOS оперативних система одређује се употребом функције *cpu_count* из библиотеке *psutil* уз прослеђивање параметра *logical=False* употребљеној функцији. За случај коришћења Linux оперативног система, као што је већ наведено у другом поглављу, постоје ситуације када претходна функција не врши тачно очитавање. Зато се уместо ње користи користи екстерни програм *lscpu* чије се позивање омогућава кроз *run()* функцију *subprocess* билиотеке и излази снимају у одговарајућу текстуалну променљиву, а потом се врши очитавање дела где је забележен података о броју физичких језгара.

За прикупљање података о количини меморије коришћена је функција *virtual_memory()* из *psutil* библиотеке. Употреба ове функције обезбеђује један шири скуп података са разноврсним информацијама о физичкој меморији рачунара. Будући да су у овом случају потребне информације о укупној радној меморији физичког система, као и тренутно расположивој количини радне меморије, из уређене n-торке која је снимљена у оквиру одговарајуће променљиве, опцијом *total*, односно *free*, вршимо екстракцију тражених вредности. Будући да функција враћа вредности у бајтовима и да оперише са јединицима са бинарним представљањем, а да хипервизор користи мегабајте приликом представљања меморије, добијене вредности се деле са 2^{20} како би се добиле адекватне вредности за даљу употребу.

За идентификацију партиција на дисковима и добијање тренутног преосталог расположивог простора на њима, када је у питању Windows оперативни систем, употребљен је екстерни Microsoftов програм под називом *Windows Management Instrumentation (WMI)* коришћењем одговарајућих параметара који приказују одређена својства дискова који се налазе у оквиру физичког система и који је позван кроз адекватну *subprocess.run()* команду. Када су у питању Linux и macOS оперативни системи, идентификација партиција обављена је кроз употребу *disk_partitions()* фунције из *psutil* библиотеке. Потом је над идентификованим партициама покренута функција *disk_usage()* која уз коришћење параметара *total* и *free*, пружа редом информације о укупној величини партиције, као и о преосталом слободном простору у оквиру дефинисане партиције. Слично уоченој ситуацији код анализе прикупљених вредности радне меморије и овде се очитани подаци исказују у бајтовима употребом јединицима са бинарним представљањем. Хипервизор користи за рад са виртуелним дисковима гигабајте, па се добијене вредности деле са 2^{30} како би се добиле адекватне вредности исказане у гигабајтима.

За синхронизацију параметара виртуелног хардвера виртуелне машине са добијеним вредностима коришћен је екстерни програм *VBoxManage* који је обезбеђен од стране произвођача хипервизора и чије коришћење има следећи облик на Windows оперативном систему:

VBoxManage modifyvm "*naziv_virtuelne_masine*"*--oznaka_parametra vrednost_parametra*

односно са обликом на Linux и macOS оперативним системима:

vboxmanage modifyvm "*naziv_virtuelne_masine*" *--oznaka_parametra vrednost_parametra*

при чему вредности *oznaka_parametra* у овде разматраним случајевима могу бити следеће:

- chipset – за избор одговарајућег виртуелног чипсета (PIIX3 или ICH9),
- cpus – за дефинисање броја језгара виртуелне централне процесорске јединице,
- memory – за дефинисање вредности виртуелне радне меморије.

Све *VBoxManage* циљаној виртуелној машини прослеђују се коришћењем више пута споменуте *subprocess.run()* функције уз коришћење свих особености Python програмског језика у оквиру кога се решење које се овде описује реализује.

Будући да се у оквиру виртуелне машине користи Linux оперативни систем, виртуелни чипсет ће увек бити подешен са вредношћу PIIX3, као што је већ било спомињано у дискусији у оквиру овог поглавља.

Број језгара виртуелне процесорске јединице, одговарајућом командом, поставиће се на вредност која је очитана као број стварних језгара централне процесорске јединице посматраног физичког система.

Што се тиче иницијализације количине виртуелне радне меморије, она ће се додељивати употребом правила исказаних у табели 14

Табела 14. Правила додељивања вредности виртуелне радне меморије

Количина радне меморије на физичком систему (M_{ht})	**Количине виртуелне радне меморије (M_{gt}) и гранична вредност количине слободне радне меморије на физичком систему (M_{hw})**
$M_{ht} \leq 4$ GB	$M_{gt} = 1$ GB $M_{hw} = 1{,}5$ GB
4 GB < $M_{ht} \leq 6$ GB	$M_{gt} = 2$ GB $M_{hw} = 2{,}5$ GB
6 GB < $M_{ht} \leq 8$ GB	$M_{gt} = 3$ GB $M_{hw} = 3{,}5$ GB
8 GB < $M_{ht} \leq 12$ GB	$M_{gt} = 4$ GB $M_{hw} = 4{,}5$ GB
12 GB < $M_{ht} < 16$ GB	$M_{gt} = 6$ GB $M_{hw} = 6{,}5$ GB
$M_{ht} \geq 16$ GB	$M_{gt} = 8$ GB $M_{hw} = 8{,}5$ GB

Као што се уочава из приказа датог у горњој табели, на основу очитане вредности количине радне меморије на посматраном физичком систему (M_{ht}), циљаној виртуелној машини се додељује одређена вредност количине виртуелне радне меморије (M_{gt}). Одлучено је да се овде не примењује рестриктивни заштитни механизам, већ да он буде реализован као прослеђивање одређеног упозорења крајњем кориснику, односно студенту на чијем рачунару се врши имплементација циљане виртуелне машине. Процес упозоравања корисника се покреће уколико је вредност очитане количине слободне радне меморије на посматраном физичком систему мања или једнака од граничне вредности количине слободне радне меморије на посматраном физичком систему (M_{hw}). У том случају извршиће се додела вредности M_{gt}, али се, уз додељивање вредности, избацује порука кориснику да је количина радне меморије на посматраном физичком систему недовољна за стабилан рад виртуелне машине уз препоруку повећања количине радне меморије или повећања количине слободне радне меморије посматраног физичког система. Одлука о нерестриктивним мерама донета је на основу искуства у раду са студентима где је уочено да су поруке упозорења дале боље резултате приликом коришћења рачунара од стране студената од примене рестриктивних мера (на пример рестрикција одређених опција софтвера и слично).

Подаци о заузећу партиција на дисковима немају сврху подешавања виртуелне машине, већ се прикупљају у сврху постизања што боље информисаности крајњег корисника о потенцијалним инцидентним ситуацијама које могу наступити приликом коришћења виртуелне машине у оквиру физичког система. Величина виртуелних дискова, у случајевима коришћења описаним у овој докторској дисертацији, дефинише се коришћењем динамичког принципа. То значи да ће се задати максималан капацитет виртуелног диска, али се физички тај капацитет неће у целости одмах алоцирати на физичком диску. Алокација простора на физичком диску ће се вршити инкременталним путем како се буде попуњавао простор и на виртуелном диску у оквиру посматране виртуелне машине с тим да важи правило да не постоји могућност умањења алоцираног

простора. Када се виртуелна машина стави на располагање крајњем кориснику и он је имплементира у оквиру капацитета свог физичког система неће одмах заузети максималан простор који је дефинисан у оквиру виртуелне машине у складу са поменутим динамичким принципом. На пример ако је виртуелни диск максималног капацитета 20 GB приликом преузимања виртуелне машине и првог стартовања он ће заузети физички свега неколико гигабајта што може заварати крајњег корисника и можда га и навести у појединим случајевима на неке погрешне закључке.

Зато се у оквиру система реализује информисаност корисника о потенцијалном достизању критичног заузећа физичког диска. Као што је већ речено претходно, прикупљају се подаци о тренутном слободном простору на свим идентификованим партицијама. За сваку партицију се потом врши процена заузећа тако што се од тренутне забележени вредности слободног простора (p_{fs}) одузме максимални капацитет виртуелног диска (d_{gmax}) и на основу добијене вредности (Δ) издају се одговарајуће поруке кориснику у складу са оним приказаним у табели 15.

Табела 15. Поруке које се издају кориснику на основу процене заузећа физичког диска

$\Delta = p_{fs} - d_{gmax}$	**Порука која се издаје кориснику**
$\Delta \leq 0$ GB	Будите опрезни, преостали простор у оквиру партиције мањи је од капацитета диска виртуелне машине. Коришћењем виртуелне машине на овој партицији може доћи до максималне попуњености капацитета ове партиције што може резултовати нестабилношћу целокупног система. Ослободите додатни простор у оквиру партиције или користите другу партицију са довољно простора.
0 GB $< \Delta \leq$ 10 GB	Будите опрезни, коришћењем виртуелне машине на овој партицији може значајно редуковати слободни простор у оквиру партиције. Уколико се оствари максимално заузеће диска виртуелне машине слободан простор на партицији биће мањи од 10 GB што може узроковати одређене проблеме. Препоручујемо да ослободите додатни простор у оквиру постојеће партиције или да користите другу партицију са више простора.
$\Delta >$ 10 GB	Партиција поседује довољно простора за несметану употребу виртуелне машине чак и у случају максималне попуњености диска виртуелне машине.

Као што се из приложеног види, поруке су осмишљене у виду одређених смерница за корисника како би што боље приступио имплементацији виртуелне машине. Генерално говорећи процес имплементације виртуелне машине могао се спровести и без посматрања заузетости партиција на диску или дисковима у оквиру посматраног физичког система, међутим сматрало се да постоји потреба за остварењем и оваквог вида информисаности крајњег корисника како би се избегле накнадне могуће инцидентне ситуације које су у почетку тешко уочљиве услед динамичке карактеристике алокације простора на партицији од стране виртуелне машине.

За остали виртуелни хардвер у оквиру циљане виртуелне машине (виртуелни мрежни адаптер, виртуелни графички адаптер, виртуелни аудио адаптер и слично) није потребно вршити локалну иницијализацију на корисничком хардверу. Остале опције виртуелног хардвера се користе у унапред дефинисаном облику од стране високошколске установе и не захтевају додатна подешавања и промене јер су подешене и одабране у складу са

потребама како виртуелне машине, тако и циљаног физичког система на коме ће се виртуелна машина извршавати.

4.3. Прикупљање података о хардверу у реалном времену

4.3.1. Партиције

У одређеним деловима овог поглавља идентификоване су одређене узрочно-последичне везе између стварног и виртуелног хардвера и том приликом адекватном анализом је утврђено да је најдоминантнија узрочно-последична веза она која обухвата стање радне меморије како на физичком рачунару, тако и на виртуелној машини. Свакако, радна меморија је јако битан фактор када је у питању стабилан рад виртуелне машине, а и посматрано у супротном смеру дешавања, свакако рад виртуелне машине меморијски може значајније црпети ресурсе физичког рачунара.

Међутим, овде се може поставити питање, у складу са претходним разматрањима, зашто се у обзир није узела и узрочно последична веза везана за заузимање простора на дисковима, односно партицијама, када се може наћи извесна аналогија са уоченим активностима везаним за рад са физичком и виртуелном радном меморијом. Свакако, одређени механизми везани за рад са партицијама, како физичким, тако и виртуелним, би се могли остварити кроз реализацију одређених сервиса, али овде се поставља питање да ли би реализација оваквих сервиса заиста било најбоље решење за крајњег корисника или не. Наиме, сви посматрани оперативни системи (комплетна Windows, Linux и macOS фамилија оперативних система) већ имају у себи уграђен низ механизама који врши одређене превентивне радње приликом рада са партицијама и истовремено врши и адекватно обавештавање корисника о инцидентиним ситуацијама и наступању потенцијално опасних ситуација. Ова понашања оперативних система једнако се испољавају без обзира да ли се ради о физичком или виртуелном систему. У складу са тим, имплементација додатног сервиса би била сувишна јер би суштински радила скоро идентичан процес оном који се већ реализује у оквиру системских процеса самих оперативних система.

Како би се ипак на одговарајући начин поткрепиле претходно изнете тврдње, спроведен је тест над физичком и виртуелном машином. За потребе теста на постојећу физичку конфигурацију, засновану на Windows оперативном систему, додат је SSD од 64 GB, и комплетан капацитет диска додељен је једној партицији. Затим је одговарајућим датотекама на датој партицији заузето више од две трећине простора (око 45 GB), односно направљена је ситуација у којој је преостало мање од 20 GB слободног простора на датој партицији. На дату партицију је ископиран виртуелни диск одговарајуће виртуелне машине који је подешен тако да се понаша по већ описаном динамичком принципу, односно приликом копирања виртуелни диск је произвео заузеће од 3.66 GB, а сам виртуелни диск поседује максимални капацитет од 20 GB. У оквиру оперативног система физичког система реализован је сервис који прати заузеће партиције скоро у реалном времену и након стартовања истог стартована је и виртуелна машина заснована на употреби виртуелног диска који је ископиран на предметну партицију. Потом се приступило копирању датотека на виртуелни диск виртуелне машине како би се повећавала заузетост виртуелног диска до граница максималног капацитета. При количини од око 10 GB ископираних података оперативни систем избацио је упозорење о попуњености партиције, а готово истовремено и у оквиру виртуелне машине добијена је порука о немогућности наставка започетог процеса копирања. Након краћег временског интервала (мање од једног минута) и имплементирани сервис избацио је адекватну поруку о недостатку простора на одговарајућој партицији. Ово кашњење

имплементираног сервиса резултат је подешености интервала узорковања података на 30 секунди како се не би додатно оптерећивали физички ресурси, док је код механизама интегрисаних у оквиру оперативног система време реакције значајно краће.

У оквиру теста реакција сервиса имплементираног изворно у оквиру оперативног система и кориснички имплементираног сервиса извршена је на скоро идентичан начин. Такође, систем имплементиран изворно у оквиру оперативног система увек ће имати краће време реакције јер ће своје активности увек спроводити на нижем нивоу и са мањим бројем остварених корака до приступа својствима хардвера од било ког имплементираног корисничког сервиса. Колика ће разлика у временима реакције бити зависи од више хардверских и софтверских параметара на којима се заснива рад посматраног физичког система. У појединим случајевима оно може бити и свега неколико секунди, али без обзира на то разлика у временима реакције ипак постоји. Оно што је примећено током теста, поред до сада наведеног, јесте да имплементација корисничког сервиса може изазвати одређену збуњеност код корисника, будући да се добијају две поруке, једна на нивоу оперативног система и једна на нивоу корисничког сервиса. Оваква појава може навести корисника да се ради о две различите појаве, да се морају предузети одвојене акције и може код корисника изазвати погрешну интерпретацију о томе шта се десило, а самим тим корисник може произвести и погрешне закључке о мерама које треба даље прдеузети.

Резултати теста потврдили су оправданост одлуке да се, у оквиру решења које се описује, не реализују сервиси који би пратили заузеће партиције која ће се користити за потребе смештања виртуелног диска коришћеног за рад виртуелне машине будући да су механизми реализовани у оквиру оперативних система сасвим довољни за рад и да увођење додатних сервиса не би било сврсисходно и додатно би збунило крајњег корисника. Иако је за потребе теста коришћен Windows оперативни систем, скоро идентични резултати добили би се и коришћењем неког другог оперативног система, па су закључци примењиви и за случај коришћења оперативног система заснованог на Linuxy или macOS оперативног система. У складу са тим праћење стања партиције у реалном времену поверава се стандардним системским сервисима реализованих оригинално у оквиру оперативног система на ком се користи одговарајући хипервизор, а самим тим и одговарајућа виртуелна машина.

4.3.2. Меморија

За разлику од разматрања везаних за проблем праћења својстава партиција, за меморију не постоје реализовани механизми у оквиру оперативног система који би се могли адекватно употребити у домену проблема који се разматра у докторској дисертацији. Зато се за меморију морају реализовати посебни кориснички сервиси како би се могло извршити праћење стања радне меморије како на физичком систему, тако и у оквиру виртуелне машине. У наставку ће се приказати специфичности реализације ових сервиса будући да се имплементација разликује за сваки од оперативних система који се разматрају, односно за Windows, Linux и macOS случајеве имплементације.

Укупна количина радне меморије физичког система је већ одређена приликом поступка иницијализације и њена вредност је снимљена у датотеку *vhinit_manifest*, која је дата у CSV формату, па се по потреби може прочитати на адекватан начин коришћењем команди за читање садржаја CSV датотеке. Будући да представља константну вредност нема потребе за понављањем њеног одређивања. Поновно одређивање количине радне меморије физичког система вршиће се једино у случају наступања промене вредности (на пример додавање новог меморијског модула, искључивање дефектног меморијског модула, повећање додељене меморије

интегрисаној графичкој карти и слични могући догађаји). Поновно одређивање спроводи се кроз претходно описане мере инцијализације које се неће овде поново описивати.

Будући да је за даље анализе потребан тачан датум и време, како бисмо добили одређене временске серије, мора се извршити бележење тачног датума и времена када је одређена вредност очитана из система. При томе ће се временска карактеристика бележити коришћењем следећег формата:

gg-mm-dd HH:MM:SS

при чему је

- gg – последње две цифре године из датума за који се врши бележење (на пример за 2023. годину биће 23),
- mm – редни број месеца из датума за који се врши бележење при чему се за једноцифрене бројеве додаје нула испред цифре (на пример месец августе је осми месец у години па ће бити 08),
- dd – редни број дана уз месецу из датума за који се врши бележење при чему се за једноцифрене бројеве додаје нула испред цифре (на пример за 5. ће бити 05),
- HH – број сати из времена за које се врши бележење, при чему се узима 24-часовни формат рачунања времена и при чему се за једноцифрене бројеве додаје нула испред цифре,
- MM - број минута из времена за које се врши бележење и при чему се за једноцифрене бројеве додаје нула испред цифре и
- SS - број секунди из времена за које се врши бележење и при чему се за једноцифрене бројеве додаје нула испред цифре.

Сходно томе на пример снимање извршено 5. августа 2023 године у 9 часова 3 минута и 7 секунди имаће временску карактеристику исказану као 23-08-05 09:03:07 . Овакав формат датума и времена није у духу српског језика, али је изабран заои што је лак за даље интерпретирање у оквиру функција за анализу података датих кроз разне библиотеке програмског језика Python.

Тренутни датум и време добијају се коришћењем библиотеке *datetime* која омогућава разне манипулације над приказом датума и времена. У оквиру поменуте библиотеке учитаће се објекат коришћењем израза *from datetime import datetime* и коришћењем функција *now()* и *strftime()* добијају се тренутни датум и време у складу са горе наведеним форматом коришћењем облика *datetime.now().strftime("%y-%m-%d %H:%M:%S))*.

Стање меморије у одређеном временском тренутку очитава се коришћењем функције *virtual_memory()* функције и њеног атрибута *free* из *psutil* библиотека па ће крајњи облик употребе бити у суштини *psutil.virtual_memory().available*. Поступак је више пута претходно објашњаван па се на овом месту неће давати додатна појашњења.

За разлику од претходних поступака где је вршено појединачно очитавање меморије, овде је потребно очитавања вршити у одређеним временским интервалима па се очитавање врши кроз одговарајућу програмску петљу. Међутим, потребно је да се итерације програмске петље врше у правилно дефинисаним временским размацима, како би се постигли једнаки интервали времена између свака два снимања у циљу добијања адекватне временске серије. Ово ће се постићи уметањем *sleep()* функције из *time* модула на крају тела програмске петље чиме се постиже чекање одговарајући број секунди пре извршења следеће наредбе. На пример *sleep(5)* ће произвести чекање од пет секунди пре него што се крене са извршавањем следеће наредбе.

Одговарајућим временским делта функцијама могуће је дефинисати период времена у коме ће се вршити снимање захтеваних параметара. Овде се одустало до таквог решења и примењено је снимање у континуитету током целокупног рада физичког рачунара, односно виртуелне машине. Овакав приступ је урађен како би се повећао волумен података који ће се користити у даљим анализама чиме се омогућава још бољи квалитет урађених анализа, а самим тим и добијање квалитетнијих закључака постаје извесније. Осим тога, оваквим континуалним приступом омогућава се конзистентност добијених података. Очитавање вредности меморије врши се на сваких 15 секунди чиме се постиже одговарајући временски интервал за завршетак свих радњи очитавања и смештај. Краћи временски интервал није изабран како би се смањила могућност појаве грешака, а процењено је да се са четири меморијске вредности по минуту може сасвим добро проценити целокупно понашање система посматрано са становишта конзумирања меморијских ресурса. Такође, овакво снимање неће оптеретити ни диск на коме се врши смештај података будући да се по једном снимању за смештај података утроши 23 бајта. То би практично значило да када се рачунар не би гасио годину дана и када би вршио непрекидно задато снимање за смештај података би се утрошило:

$$23\ \text{B} \cdot 4 \cdot 60 \cdot 24 \cdot 365 = 48.355.200\ \text{B}$$

односно ако се то преведе у мегабајте коришћењем бинарне метрике, у складу са оним што се користи у оквиру различитих модула за анализу података и ако се зна да је:

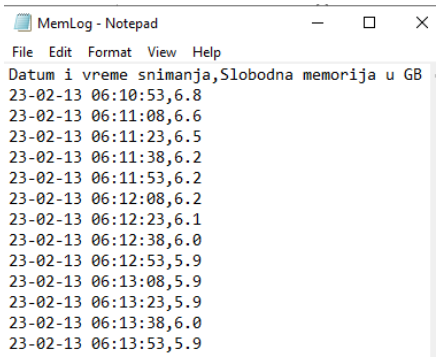
$$1\ \text{MB} = 1024\ \text{KB} = 1024 \cdot 1024\ \text{B} = 2^{20}\ \text{B}$$

биће:

$$\frac{48.355.200}{2^{20}}\ \text{MB} \approx 46{,}115\ \text{MB} \approx 46\ \text{MB}$$

што за данашње кацапитете за смештај података код савремених рачунара представља врло малу вредност, па је тиме доказано да неће доћи до значајнијег оптерећења партиције где се врши смештај очитаних података.

Чување података врши се у оквиру текстуалне датотеке *MemLog* која је дата у CSV формату, чиме се омогућава једноставан и брз приступ подацима намењеним за извођење одговарајућих анализа. Датотека се снима локално, односно у оквиру оних ресурса над којим се врши прикупљање одговарајућих података. Пример овако реализоване датотеке дат је на слици 56.



Слика 55. Пример датотеке *MemLog*

Поступак снимања текстуалне датотеке у CSV формату коришћењем *csv* модула је већ претходно објашњаван, па се на овом месту неће поново, додатно објашњавати.

Претходни поступци реализују се кроз одговарајуће корисничке сервисе у оквиру задатог оперативног система, развијених у оквиру докторске дисертације. Ти сервиси морају имати могућност аутоматског стартовања по укључивању рачунара чиме се интеракција са корисником своди на минимум и елиминишу грешке које настају услед људског фактора. Имајући у виду да су ови сервиси кључни за решење које се овде описује требало би онемогућити заустављање и онемогућавање истих од стране крајњег корисника. Међутим, будући да се ради о личној својини, односно личним рачунарима студената, ипак се мора дати потпуна контрола над сервисима крајњим корисницима, односно студентима, у противном би се могло рећи да долази до промене својстава личне својине на недозвољен начин, што се не сме урадити. У складу са тим, за сваки од реализованих предметних сервиса дозвољена је потпуна контрола од стране крајњег корисника, односно сваки корисник, без обзира на аутоматско понашање сервиса, исти може стартовати, поново покренути и искључити у потпуности, без обзира што омогућавање таквих контрола није добро по само решење и може проузроковати неке бочне ефекте у самом решењу.

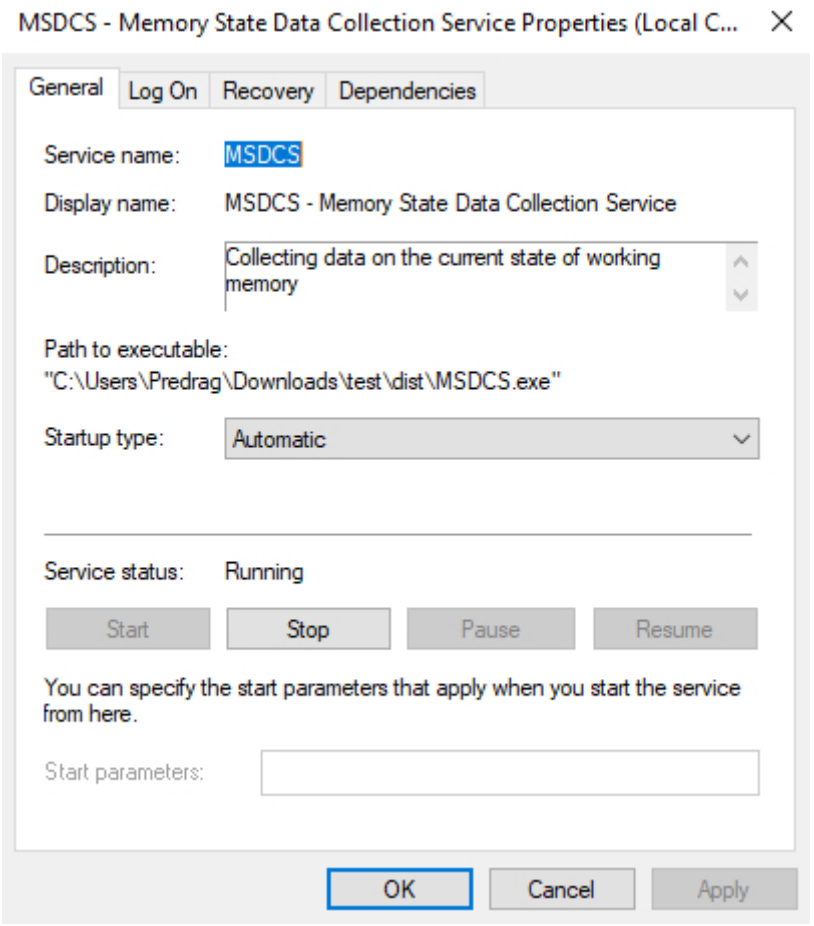
Имајући у виду да сваки оперативни систем има своје специфичне механизме за креирање сервиса и њихову контролу, не може се урадити свеобухватно и целовито решење за све оперативне системе, већ се за сваки примењени оперативни систем мора урадити посебно програмско решење. У складу са тим у наставку ће се описати реализација сервиса парцијално за Windows системе, Linux дистрибуције и macOS системе. Овде треба напоменути да код појединих оперативних система постоји више решења за изградњу сервиса. У решењу које се описује овом докторском дисертацијом, за сваки од оперативних система искоришћено је само по једно решење за изградњу сервиса, али то не значи да се до реализације не може доћи и другачијом изградњом сервиса, међутим овде нећемо залазити дубље у ову проблематику будући да то није окосница разматрања изнетих у дисертацији.

Изградња сервиса за Windows оперативне системе ослања се на коришћење *pywin32* који представља скуп Python проширења за Windows (PYthon extensions for WINdows) [153,154]. Будући да скуп поменутих проширења поседује широк скуп модула за реализацију различитих могућности у оквиру Windows фамилије оперативних система треба нагласити да су за конкретну изградњу сервиса у овом случају коришћени следећи модули из *pywin32*: *win32event, win32service, win32serviceutil* и *servicemanager*.

Изградња сервиса базира се на дефинисању класе која носи име сервиса. У случају који се презентује сервис је назван *MSDCS* (скраћеница од *Memory State Data Collection Service*), па ће се класа дефинисати као *class MSDCS(win32serviceutil.ServiceFramework)*. У оквиру реализоване класе дефинишу се константе које омогућавају приказивање основних информација о сервису у складу са [155] и то:

- *_svc_name_ = "MSDCS"* – име сервиса које ће се користити приликом инсталације истог и сличних поступака у оквиру задатог Windows окружења,
- *_svc_display_name_ = "MSDCS - Memory State Data Collection Service"* – описно име сервиса које ће се приказивати у листи сервиса у оквиру задатог Windows окружења и
- *_svc_description_ = "Collecting data on the current state of working memory"* - кратак опис сервиса.

Употреба ових информација јасно се види у приказу датом на слици 56 где је приказан прозор сервиса у ком се види употреба информација садржаних у горњим констатнтама.



Слика 56. Прозор у ком се врши управљање реализованим сервисом

Међутим окосницу сервиса не чине само прикази одређених информација, већ се мора омогућити покретање и заустављање сервиса, као и јасно дефинисање посла који ће сервис обављати све док је у покренутом стању. То се постиже дефинисањем одређеног броја функција у оквиру поменуте реализоване класе, чиме се остварују циљане функционалности сервиса. Поменуте реализоване функције су:

- *__init__(self,args)* – омогућава креирање објеката из дефинисане класе, односно иницијализацију атрибута,
- *SvcStop(self)* – омогућава заустављање рада сервиса,
- *SvcDoRun(self)* – омогућава покретање рада сервиса и
- *main(self)* – функција која садржи команде које омогућавају реализацију задатака које сервис извршава све док је у покренутом стању.

Као што се види из претходног описа најважнија функција јесте функција *main* будући да се у оквиру ње, коришћењем одговарајуће петље, у свакој итерацији врши одговарајуће очитавање стања меморије и потом се тај податак снима на адекватан начин. Садржај ове функције био би довољан за реализацију ако би се о извршењу програма старао сам корисник, међутим, као што је већ више пута наглашено жели се постићи што већи степен аутоматизације процеса, односно што већи степен минимизације неопсредног учешћа корисника. Зато поред ове функције постоје две додатне функције. Задатак *SvcDoRun* функције је да дефинише шта ће се десити када се сервис покрене и то се мора експлицитно нагласити, односно у оквиру ове функције се

врши позив *main* функције. Насупрот њој *SvcStop* дефинише које све активности треба предузети од оног момента када се добије сигнал за заустављање сервиса па до стварног заустављања сервиса. У оквиру ове функције мења се стање контролне променљиве из *True* у *False* која омогућава заустављање својих активности које се реализују у оквиру main функције. У противном сервис би добио сигнал за заустављање, али се процеси који се реализују у оквиру сервиса не би никада зауставили јер *main* функција не би била свесна да је дошло до заустављања рада, па је неопходно увођење контролне променљиве у циљу омогућавања адекватног рада свих компоненти сервиса.

Као што се види из приложеног, све компоненте Windows сервиса дефинишу се интерно у оквиру извршне датотеке. Код Linux и macOS заснованих система извршна датотека је састављена само од оних делова кода који омогућавају реализацију задатака за које је сервис намењен, док су остале компоненте сервиса дефинисане екстерно, односно дефинишу се у оквиру неких других датотека, ван извршне датотеке. Извршна датотека их не укључује ни на који начин, већ напротив екстерне датотеке укључују извршну како би реализовале пуну функционалност предвиђеног сервиса.

Развој сервиса у Linux окружењу базираће се на употреби *systemd* који представља софтвер који управља системом и сервисима у оквиру Linux оперативних система [156,157]. Његов рад је организован кроз одговарајуће јединице (systemd units), при чему је свака јединица представљена одговарајућим конфигурационим датотекама јединице смештеним на одговарајућим унапред дефинисаним локацијама унутар организације самог Linux система. У складу са тим, развој сервиса врши се у сервисној јединици (service unit) при чему релевантне конфигурационе датотеке имају наставак *.service* и снимају се на локацији која се налази на путањи */etc/systemd/system*, па ће сходно томе за сервис под именом *MSDCS,* који се развија за потребе решења представљеног у докторској дисертацији име конфигурационе датотеке са одговарајућом путањом бити дато као */etc/systemd/system/MSDCS.service.* Пример изгледа наведене конфигурационе датотеке дат је на слици 57.

```
[predrag@a10 ~] $ cat /etc/systemd/system/MSDCS.service
[Unit]
Description=MSDCS - Memory State Data Collection Service

[Service]
User=root
WorkingDirectory=/home/predrag
ExecStart=/home/predrag/Documents/dist/MSDCS
Restart=always

[Install]
WantedBy=multi-user.target
[predrag@a10 ~] $
```

Слика 57. Пример конфигурационе датотеке MSDCS.service

Конфигурациона датотека састављена је од одговарајућих секција, при чему је почетак сваке секције задат именом секције приказаном у оквиру угластих заграда ([]). У том смислу конфигурациона датотека која се овде користи састављена је од три секције: *Unit, Service* и *Install*. *Unit* секција се користи за дефинисање метаподатака и у оквиру ње може бити садржан велики број директива, међутим у случају који се овде разматра постоји потреба за исказивањем само једне директиве а то је директива *Description* у којој се даје кратак опис намене сервиса за који се креира предметна конфигурациона датотека. Секција *Service* егзистира само у конфигурационим

датотекама које се реализују за потребе сервиса и она поседује мноштво директива које се могу имплементирати у зависности од реалне потребе. У конкретном случају који је овде приказан идентификоване су и реализоване следеће директиве:

- *User* – везивање покретања сервиса за одређеног корисника дефинисаног у оквиру система на ком се сервис покреће,
- *WorkingDirectory* – директоријум над којим се врши покретање сервиса и који ће се користити за предузете радње у оквиру сервиса,
- *ExecStart* – команда која ће се извршити по покретању сервиса, у овом случају извршна датотека која садржи све неопходне елементе намењене прикупљању података о стању меморије и њиховом одговарајућем снимању за даљу употребу и неопходне анализе и
- *Restart* – дефинише могућност сервиса да изврши самостално рестартовање у случају да дође до неког неочекиваног прекида процеса који је покренут од стране сервиса (не односи се на ситуацију када се примени одговарајућа *stop* команда за заустављање сервиса).

У овој секцији треба посебну пажњу обратити на добро дефинисање вредности директива, јер се могу јавити ефекти који доводе до неправилног рада сервиса и поред тога што се сама процедура стартовања и рада сервиса спроводи без појаве грешака. На пример ако се не изабере адекватно корисник који ће се користити за покретање сервиса могу се јавити проблеми са дозволама за упис података и слично. И последња, *Install* секција може имати више директива. Ова секција намењена је реализацији аутоматског покретања сервиса приликом подизања оперативног система и дефинише одређено понашање сервиса које се реализује том приликом. У оквиру ове секције искоришћена је једино директива *WantedBy* која означава стања система за која ће бити извршено аутоматско покретање датог сервиса. У овом конкретном случају наведена је вредност *multi-user.target* која се неће овде посебно разматрати будући да треба дубље ући у разматрање конструкције самог оперативног система што није предмет разматрања ове докторске дисертације. Указаће се само да се коришћењем наведене вредности покривају сви случајеви коришћења оперативног система од стране корисника који су од интереса за подручје рада докторске дисертације, односно, адекватно аутоматско покретање сервиса биће извршено у свим разматраним случајевима.

Комплетна манипулација сервисом обавља се коришћењем *systemctl* наредбе која у овом случају има облик *systemctl akcija MSDCS.service* при чему *akcija* може бити следећа:

- *enable* - омогућава аутоматско стартовање сервиса приликом подизања оперативног система,
- *disable* - укида аутоматско стартовање сервиса приликом подизања оперативног система,
- *start* - омогућава мануелно стартовање сервиса,
- *stop* - омогућава мануелно заустављање сервиса,
- *restart* - омогућава мануелно рестартовање сервиса,
- *status* - омогућава проверу тренутног стања сервиса.

Пример реализовања *systemctl* наредбе за реализовани сервис, као и извештаја о тренутном стању реализованог сервиса, дат је на слици 58.

```
[predrag@a10 ~] $ sudo systemctl status MSDCS.service
● MSDCS.service - MSDCS - Memory State Data Collection Service
     Loaded: loaded (/etc/systemd/system/MSDCS.service; enabled; preset: disabled)
     Active: active (running) since Tue 2023-08-22 23:14:34 CEST; 8s ago
   Main PID: 3571 (MSDCS)
      Tasks: 2 (limit: 17665)
     Memory: 38.5M
        CPU: 485ms
     CGroup: /system.slice/MSDCS.service
             ├─3571 /home/predrag/Documents/dist/MSDCS
             └─3576 /home/predrag/Documents/dist/MSDCS

Aug 22 23:14:34 a10 systemd[1]: Started MSDCS.service - MSDCS - Memory State Data Collection Service.
[predrag@a10 ~] $
```

Слика 58. Пример стартованог сервиса у Linux окружењу

Слични принципи изградње сервиса оним реализованим у Linux окружењу срећу се и приликом реализације сервиса у оквиру macOS система. У случају који се овде разматра, реализација сервиса у оквиру macOS оперативног система извршиће се употребом *launchd* помоћног системског софтвера [158,159]. И у оквиру macOS система важи правило да се све дефиниције везане за реализацију самог сервиса врше екстерно, ван извршне датотеке, у посебној наменској датотеци у којој се декларишу својства сервиса и из које ће се извршити позивање поменуте извршне датотеке. Та посебна наменска датотека у оквиру *launchd* дефинисана је као датотека са списком својстава (property list file), означава се *.plist* екстензијом и смешта се на унапред дефинисане путање у оквиру система. Пример такве датотеке дат је на слици 59.

MSDCS.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Disabled</key>  <false/>
    <key>RunAtLoad</key> <true/>
    <key>KeepAlive</key> <true/>
    <key>Label</key>     <string>MSDCS</string>
    <key>Program</key>   <string>/Users/predrag/Documents/dist/MSDCS</string>
</dict>
</plist>
```

Слика 59. Пример датотеке са листом својстава реализоване за потребе покретања сервиса у оквиру launchd

Разликују се два типа сервиса, сервис типа агента и сервис типа демона. У разматраном случају развијен је сервис типа демона под именом *MSDCS* и који, у складу са системским захтевима macOS система, има одговарајућу датотеку са листом својстава *MSDCS.plist* смештену у оквиру путање */Library/LaunchDaemons* која је подразумевана за смештање демона развијаних од стране трећих лица који се покрећу у оквиру оперативног система. Поменути сервис је могуће реализовати и као демона у оквиру постојећег корисничког налога, међутим одлука је да реализација буде у оквиру демона на нивоу целокупног оперативног система из разлога што се од корисника неће захтевати никаква додатна подешавања, док би се у случају реализације у оквиру поменутог корисничког налога морала од стране корисника извршити додатна безбедоносна подешања што је супротно захтеваном смањењу директног учешћа корисника у целокупном процесу имплементације. Такође, уколико би се корисник улоговао у оквиру неког другог корисничког налога морала би се извршити поновна процедура инсталације и подешавање демона, што опет захтева интеракцију са корисником, као и напредна

знања из области системске администрације macOS система што се не може сматрати универзалним знањем сваког корисника. Као што је већ речено демони развијени од стране трећих лица смештају се у оквиру путање */Library/LaunchDaemons*, па ће цела путања заједно са именом поменуте датотеке која садржи декларисана својства бити */Library/LaunchDaemons/MSDCS.plist*. Сама датотека реализује се употребом XML структуре коришћењем елемената својствених намени датотеке исказаних у виду кључева и њихових вредности.

У оквиру .plst датотеке постоји могућност дефинисања врло великог броја својстава у зависности од захтева који се постављају пред сам сервис. Комплексност реализованог сервиса који се описује није високог нивоа будући да покреће извршни датотеку која нема никаквих додатних параметара и све предвиђене активности се остварују самом извршном датотеком (очитавање параметара и њихово снимање). У складу са ниским нивоом комплексности, одабран је минимални скуп својстава који је неопходан за адекватно описивање сервиса. Будући да је свако својство, као што је већ претходно указано, дефинисано кључевима и њиховим вредностима, даје се кратак опис коришћених кључева у случају реализоване датотеке:

- *RunAtLoad* – омогућавање стартовања сервиса аутоматски одмах по његовом учитавању, не захтева се додатна интеракција у виду било какве команде за стартовање,
- *KeepAlive* – омогућавање рада сервиса без обзира на наступање неких нових околности у оквиру система, при чему ће у таквој ситуацији сервис бити аутоматски рестартован и наставити са даљим радом,
- *Label* – име сервиса које ће се приказивати у листи учитаних и активних сервиса, при чему име сервиса мора бити идентично имену одговарајуће .plist датотеке (уколико се разликују рад сервиса неће бити могућ) и
- *Program* – име извршне датотеке, заједно са њеном целокупном путањом на постојећем систему, која се извршава по покретању сервиса.

Манипулација радом сервиса обавља се коришћењем *launchctl* команде која омогућава спровођење одређених дефинисаних акција над *launchd* процесима. Уопштено говорећи команда има следећи облик *launchctl akcija putanja/datoteka.plist*, односно конкретизовано за претходна разматрања *launchctl akcija /Library/LaunchDaemons/MSDCS.plist*, при чему *akcija* може бити:

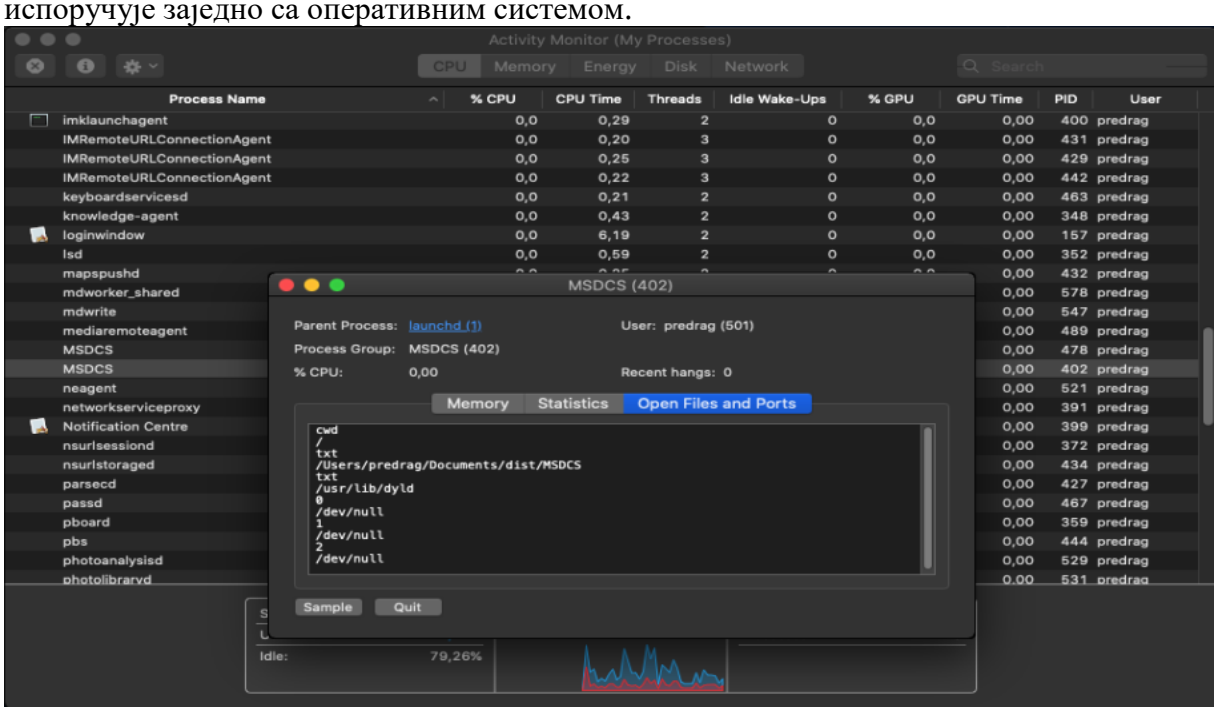
- load – учитавање сервиса и
- unload – брисање сервиса.

Како би сервис био перманентно учитан, односно како се дати процес не би понављао приликом сваког стартовања система, потребно је додати –w опцију пре навођења путање, па ће тако команда која ће омогућити да разматрани сервис буде стално учитан имати облик *launchctl load –w /Library/LaunchDaemons/MSDCS.plist*. Преглед свих учитаних сервиса остварује се командом *launchctl list* којом се може извршити провера да ли је дати сервис учитан или не. Учитаним сервисима се може даље манипулисати користећи облик команде *launchctl akcija ime_servisa*, односно у конкретно посматраном случају ће бити *launchctl akcija MSDCS*, при чему *akcija* у овом случају може бити:

- start – покретање сервиса и
- stop – заустављање рада сервиса.

Овде треба напоменути да се ове акције могу спроводити само над учитаним сервисима.

На слици 60 показан је пример учитаног и стартованог сервиса у оквиру macOS система. Приказ је остварен коришћењем апликације ActivityMonitor која се стандардно испоручује заједно са оперативним системом.



Слика 60. Пример стартованог сервиса у оквиру launchd

Извршне датотеке за сваки од разматраних оперативних система (Windows, Linux, macOS) креиране су употребом *pyinstaller* софтвера на одговарајућем оперативном систему. Овде треба напоменути да је у случају Linux и macOS оперативних система код који је претваран у извршну датотеку идентичан и састоји се од еопходних команди за прибављање података о тренутно стању радне меморије, док је у случају Windows система тај код проширен одговарајућим командама за потребе реализације сервиса јер као што је већ споменуто, једино се у Windows систему команде за изградњу сервиса налазе интерно у оквиру извршне датотеке.

Без обзира на посматрани систем, у свим случајевима остварује се очитавање тренутних вредности доступне радне меморије за даљи рад и снимање тих вредности у одговарајућу текстуалну датотеку CSV формата. Тиме се омогућава доступност података за даљу анализу на један једноставан и флексибилан начин, а сами сервиси омогућавају да се подаци адекватно прикупе без неопходне интеракције са самим корисником, чиме се могућа појава грешака приликом прикупљања података своди на прихватљиви минимум.

На крају треба напоменути и то да се све инсталационе процедуре остварују одговарајућим скриптовањем за сваки од оперативних система чиме је омогућено аутоматизовање поступака неопходних за спровођење инсталације, чиме се обезбеђује висок степен тачности спровођења самог инсталационог поступка. Учешће крајњег корисника, односно студента сведено је само на преузимање одговарајуће скрипт датотекке и на њено покретање у оквиру циљаног оперативног система. Претходно се односи искључиво на физички хардвер, док је у случају виртуелне машине која се обезбеђује за студента имплементација одговарајућег сервиса и софтвера у оквиру виртуелне машине већ извршена, односно од студента се не захтева била каква интеракција у том погледу.

Сви подаци се прикупљају локално и складиште се локално у оквиру постојећих капацитета личних рачунара студената. Ни у једном тренутку не постоји никаква интеракција са било каквим сервисом на интернету, чиме се ово решење добро уклапа и у правне норме, нарочито у разним сегментима безбедности података и заштите личних података.

4.4. Анализа постојећих и предикција будућих стања

Претходно је описана реализација решења која, кроз имплементацију одговарајућих сервиса, омогућавају снимање тренутног стања радне меморије, читањем вредности тренутно слободног капацитета радне меморије и снимањем те вредности у одговарајућу текстуалну датотеку (*MemLog*). Та текстуална датотека садржи све забележене вредности од момента стартовања сервиса до момента гашења. Поновним стартовањем сервиса не врши се додавање нових вредности у постојећу датотеку, већ се поново формира поменута датотека и почиње се упис вредности од прве позиције у датотеци. Овај поступак се примењује увек након поновног покретања сервиса како би се адекватно објединиле све вредности током једне сесије коришћења рачунара. Међутим, пре поновног формирања датотеке за упис поменутих тренутних вредности, врши се одговарајући поступак анализе претходно уписаних података у датотеку, врши се упис добијених резултата у другу текстуалну датотеку која чува податке те анализе и тек након окончања тог поступка формира се нова текстуална датотека за упис тренутних вредности и целокупни процес читања и уписа тренутних вредности креће из почетка.

Приликом покретања сервиса, а пре кретања у поступак читања стања радне меморије, врши се прорачун средње вредности количине слободне радне меморије за претходну сесију, односно за вредности забележене у том тренутку у оквиру датотеке *MemLog*. Прорачун средње вредности врши се у складу са формулом (1) за израчунавање средње вредности узорка. Први корак у израчунавању поменуте средње вредности јесте учитавање вредности забележених у оквиру датотеке *MemLog*. Будући да се за статистичка израчунавања користити модул *pandas*, исти ће се искористити и за налажење средње вредности у овом случају. Сходно томе, користиће се функција *read_csv()* која омогућава читање података из датотеке засноване на CSV формату, каква је *MemLog* и директну конверзију прочитаних података у одговарајући dataframe. Из постојећег dataframea издвојиће се колона која садржи само одговарајуће меморијске вредности и извршити над њима одговарајући даљи поступци конверзије, будући да су подаци сачувани у текстуалном формату, а за даље прорачуне потребно је добити одговарајуће бројчане вредности. Поред текстуалних конверзија извршиће се и крајња конверзија коришћењем функције *astype(float)* како би се добиле одговарајуће вредности са тачношћу од једне децимале и тако правилно формиране вредности за даљи прорачун сачуваће се у нови dataframe. Над новоформираним dataframeом извршиће се функција *mean()*, која израчунава средњу вредност за све меморијске вредности лоцираних у предметном dataframeу. Овако добијена средња вредност се потом, заједно са датумом и временом извршеног прорачуна датим у већ поменутом облику gg-mm-dd HH:MM:SS, уписује у текстуалну датотеку *MemLogMean*. Датотека *MemLogMean* дата је у CSV формату, па ће се сходно томе и у овом случају, за све потребне манипулације приликом уписа, користити функције модула *csv* на начин као што је већ било описивано у претходним поглављима ове докторске дисертације.

Тек након извршеног уписа одговарајуће вредности у *MemLogMean* датотеку, претходно описани сервиси могу предузети даље радње. Датотека *LogMean* се ресетује, односно претходно уписани подаци у исту се бесповратно уништавају и креће се са уписом нових података актуелне сесије. Као што је већ речено овај поступак се понавља

сваким новим стратовањем сервиса, било да је он остварен приликом стартовања оперативног система рачунара, било да се ради о ручном рестартовању сервиса од стране корисника или некој другој околности покретања сервиса. Овакав начин руковања подацима чини податке једноставнијим за даље процесирање, добијају се квалитетнији сетови података и смисленији подаци, подиже се степен конзистентности и остварују се многе друге погодности.

Поред статистичких пројекција, за успешност примене решења које се описује у овој докторској дисертацији битно је остварити и одговарајуће предикције везане за меморијска стања која се остварују како на нивоу физичког рачунара, тако и на нивоу саме виртуелне машине.

Разни облици предвиђања и дејства на основу тих добијених предвиђања већ су вршени у домену рада виртуелних машина како би се оптимизовало коришћење меморијских ресурса. Једна од најзаступљенијих техника ове врсте је тзв. „меморијско балонирање“ (memory ballooning) које омогућава расподелу одређених меморијских ресурса међу више виртуелних машина [160-162] и која се у пракси често користи на наменским серверима који опслужују већи или велики број виртуелних машина. Суштина овог решења је могућност да се одређена количина меморије „одузме“ од виртуелне машине која има смањену активност (deflate) и та количина преусмери и „дода“ другој виртуелној машини која услед повећане активности захтева додатне меморијске ресурсе (inflate). Целокупан процес балонирања обавља се без потребе да се током поступка виртуелне машине рестартују што је једна од добрих страна овог процеса. При томе се не захтева додатна количина меморије од самих физичких ресурса будући да се целокупан процес одвија у оквирима већ додељене количине меморије од стране физичких ресурса потребном виртуелном хардверу. Овакав приступ је присутан у многим савременим хипервизорима који се данас користе у различитим поступцима виртуелизације и препоручљиво је омогућити примену датог поступка када се ради о већем броју реализованих виртуелних машина у оквиру једног хипервизора у циљу омогућавања бољег располагања постојећим физичким ресурсима, боље оптимизације виртуелних машина, као и остварења бољих перформанси истих.

Међутим, овакав вид решења је неприменљив у случају реализације решења које се остварује у оквиру предметне докторске дисертације из неколико разлога. Овде приказано решење се остварује у већини случајева над далеко скромнијим хардверским ресурсима будући да се ради о личним рачунарима студената, те су и сами ресурси који се додељују виртуелном хардверу далеко скромнијег капацитета у односу на случај када се ради о серверским решењима које спроводе виртуелизацију. Такође, ти рачунари у оквиру својих свакодневних задатака спроводе више различитих активности у односу на наменске сервере који су у већини случајева фокусирани на неку специјализовану врсту активности. У случају који се овде разматра не користи се више виртуелних машина, већ се користи само једна чија је намена да створи једно унифицирано окружење за рад студената и остварење задатака у оквиру предвиђених наставних активности које се остварују у високошколском образовању. Дакле, не постоји више виртуелних машина, већ само једна, па се овде не могу испољавати односи између виртуелних машина већ је израженије фокусирање на односе који владају између система домаћина (host) и гостујућег система (guest). Овде треба додати и циљност предвиђеног решења на мултиплатформску применљивост, односно, као што је већ указано, решење треба применити у потпуности на Windows, Linux и macOS окружење коришћењем задатог хипервизора, а познато је да на пример за macOS окружење у оквиру задатог хипервизора технике балонирања нису подржане [163].

Имајући напред наведено у виду, развијене су нове технике у циљу задовољавања циљева које спроводи решење приказано у оквиру докторске дисертације. Будући да се

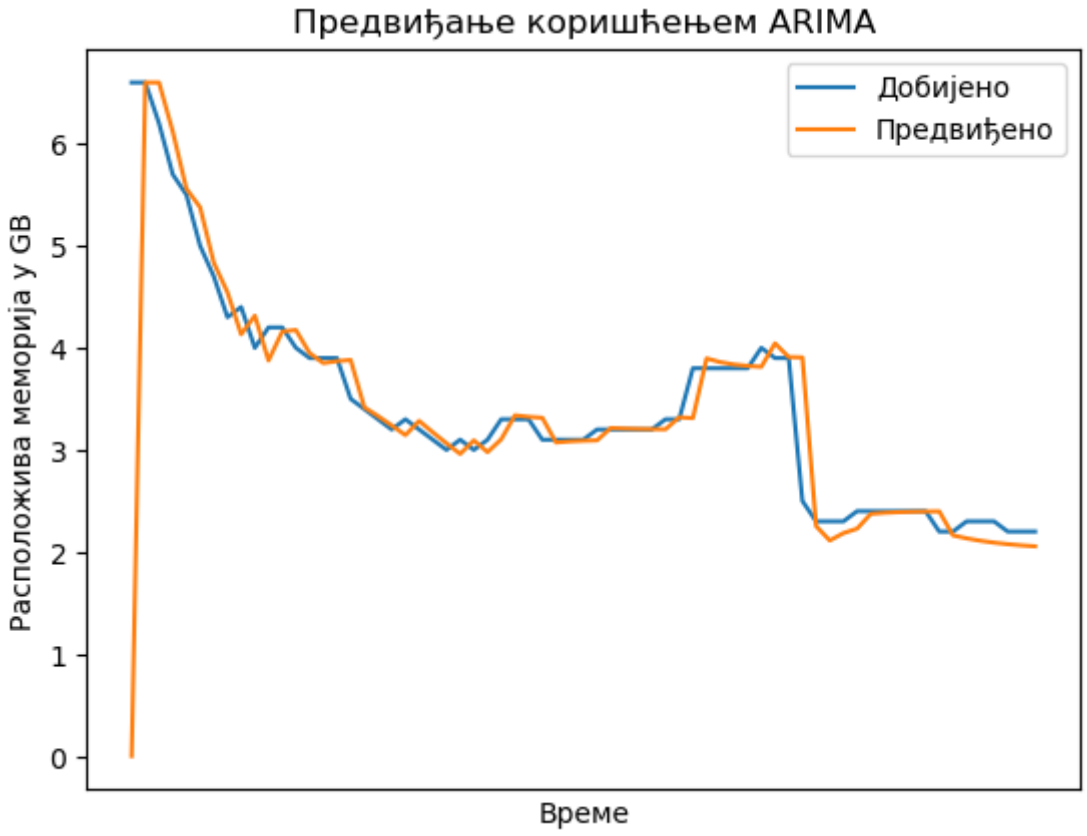
користе новоформиране технике, формирају се и нови облици предвиђања који се уклапају у домен рада тих новоформираних техника. Приказ конструкције механизама задужених за остваривање задатака из сфере предиктивне аналитике у оквиру предвиђеног решења даје се у наставку.

Како би одговарајућа предикција била остварена, мора се употребити одговарајући начин за њено реализовање заснован на неком моделу предикције. Постоји доста различитих алгоритама којима се може остварити предиктивна анализа и добити одговарајуће прогнозиране вредности. Избор алгоритма углавном се базира на два фактора, питању које се поставља и подацима који су доступни у сврху тражења одговора на то постављено питање [164]. Међутим, који ће конкретно алгоритам и модел базиран на том алгоритму бити искоришћен у домену решавања неког проблема не може зависити искључиво од претходно наведених фактора без обзира на примарност тих фактора. Треба узети и у обзир перформансе, могућност интегрисања у одговарајуће решење и читав низ других фактора. Који ће се фактори посматрати приликом одлуке о примени неког алгоритма и модела за остваривање одређене прогнозе зависи од случаја до случаја, односно од природе самог проблема који се настоји решити базирањем на овим алгоритмима и моделима. Алгоритам и модел заснован на одређеном алгоритму примењен на неком конкретном решењу не значи да ће се аутоматски добро уклопити у трагању за неким другим решењем, без обзира на сродност проблема.

Зато ће се у трагању за конкретним решењем у домену проблема који се обрађује у докторској дисертацији анализирати три модела како би се изабрао међу њима најподесинији за примену у конкретизацији решења над проблемом који се истражује. Како би се омогућила међусобна упоредивост анализираних модела, сви модели свој рад заснивају над истим скупом података. Поменути улазни скуп података за моделе реализован је као реални скуп података и представља једну инстанцу описане *MemLog* датотеке генерисане на једном рачунару у току једне радне сесије приликом реалних услова рада корисника. Тиме се омогућава реална упоредивост рада поменутих модела будући да се користе очитане меморијске вредности у стварним условима рада са свим својим карактеристикама и особеностима. Овакав принцип је примењен јер се сматра да код вештачки генерисаних података, без обзира на квалитет извршене симулације, може доћи до одсуства неких кључних момената који се појављују у реалном раду. На пример реални рад може подразумевати више кључних тачака у којима се дешавају значајније промене, док код симулације овај број тачака може бити значајније мањи, а и сам нагиб угла остварене промене меморијских вредности може значајније бити различит у реалним условима од оних добијених симулацијом. Зато је овде предност дата стварним подацима над оним који би се добили вештачким путем. Сви посматрани модели, дакле, користиће исти скуп података будући да се подаци могу припремити једном и користити у сва три случаја, односно за сва три модела чиме се омогућава смисленост компарације рада посматраних модела [165]. У оквиру докторске дисертације упоређени су следећи модели: интегрисани ауторегресиони модел покретних просека - ARIMA (Auto-Regressive Integrated Moving Average) [166], случајна шума (Random Forest) [167] и Prophet [168].

ARIMA модели представљају једне од најчешће употребљаваних модела при решавању проблема који се заснивају на предвиђањима реализованим над временским серијама података [169]. Будући да се проблем који се анализира на овом месту заснива на одређеној временској серији података, одлучено је да један од анализираних приступа буде базиран на коришћењу ARIMA заснованог модела над претходно припремљеним скупом података. Припрему података за одговарајуће моделирање извршено је употребом одговарајућих функција из *pandas* модула чиме је омогућен увоз одговарајућих вредности из *MemLog* датотеке, дате у CSV формату, у одговарајући

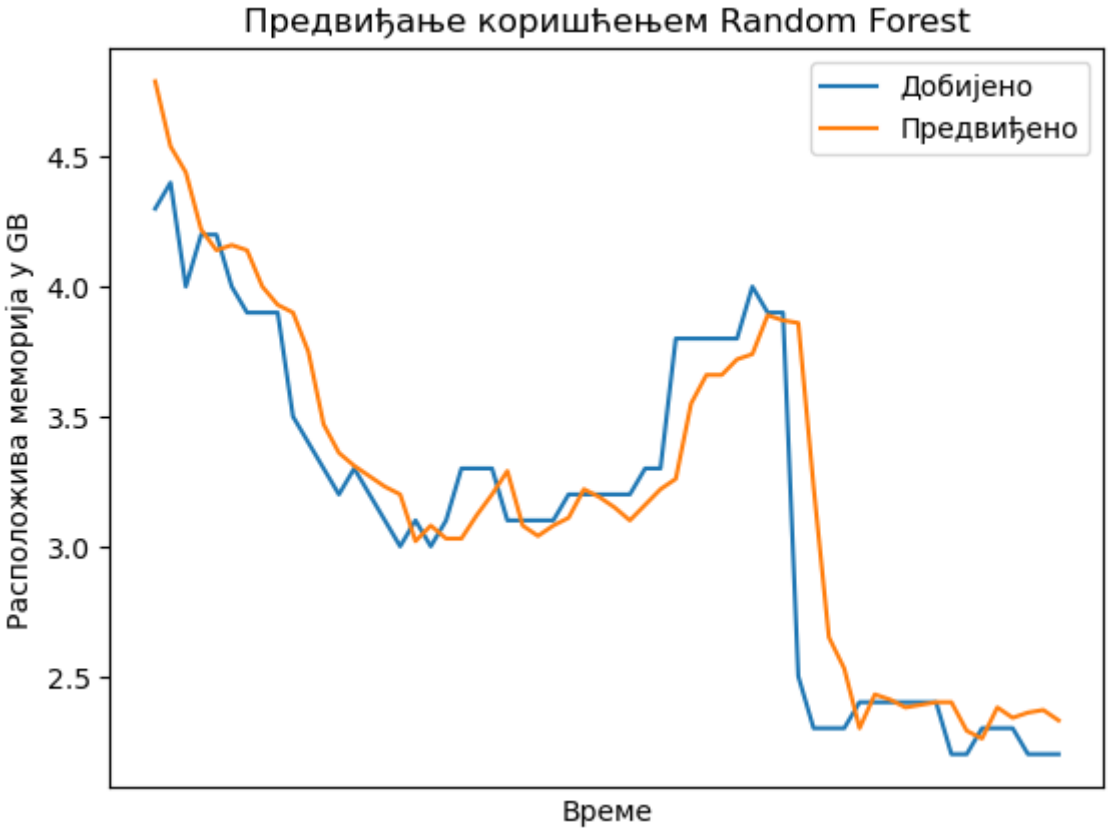
dataframe, а потом су извршене одговарајуће конверзије над подацима како бисмо добили потребне бројчане вредности за даљи процес. Претходни поступак је већ претходно објашњаван, па се на овом месту неће поново објашњавати. Овако добијени подаци подељени су функцијом *iloc* у односу 80:20, односно 80 % података опредељено је за тренирање модела, док је 20 % опредељено за тестирање модела. Над овако формираним скуповима података приступило се ARIMA моделирању коришћењем *statsmodels.tsa.arima.model* модула и његове функције *ARIMA(data,order=(p,d,q))*, где је p ред ауторегресивног модела, d степен диференцирања и q ред покретних просека, док *data* представља скуп података намењен тренирању модела. Након неколико тестирања модела, утврђено је да се најбољи резултати остварују за *(p,d,q)=(1,1,1)*, мада се мора напоменути да се изузетно добри резултати добијају и за *(p,d,q)=(1,0,0)* када у суштини ARIMA модел постаје AR (Auto-Regression) модел, односно ауторегресиони модел првог реда (често у литератури означаван и као AR(1)). Над дефинисаним моделом примењена је *fit()* функција како би се извршила обука модела над прецизираним скупом података, односно у овом кораку врши се прилагођавање дефинисаног модела подацима. Након извршених претходних корака приступа се налажењу прогнозираних вредности употребом функције predict() која предвиђа вредности на основу претходне извршене обуке задатог модела. Приказ прогнозираних вредности у односу на добијене вредности у оквиру *MemLog* датотеке графички је приказан на слици 61. Графичка интерпретација резултата остварена је употребом *Pyplot* интерфејса, модула *Matplotlib* [170], који у себи обједињује функције графичког приказа налик оном који се остварује у оквиру програмског пакета MATLAB [171].



Слика 61. Приказ резултата добијених коришћењем ARIMA модела

Следећи модел који је тестиран јесте модел заснован на примени случајних (насумичних) шума (Random Forest). Алгоритам случајних шума заснива се на стаблима

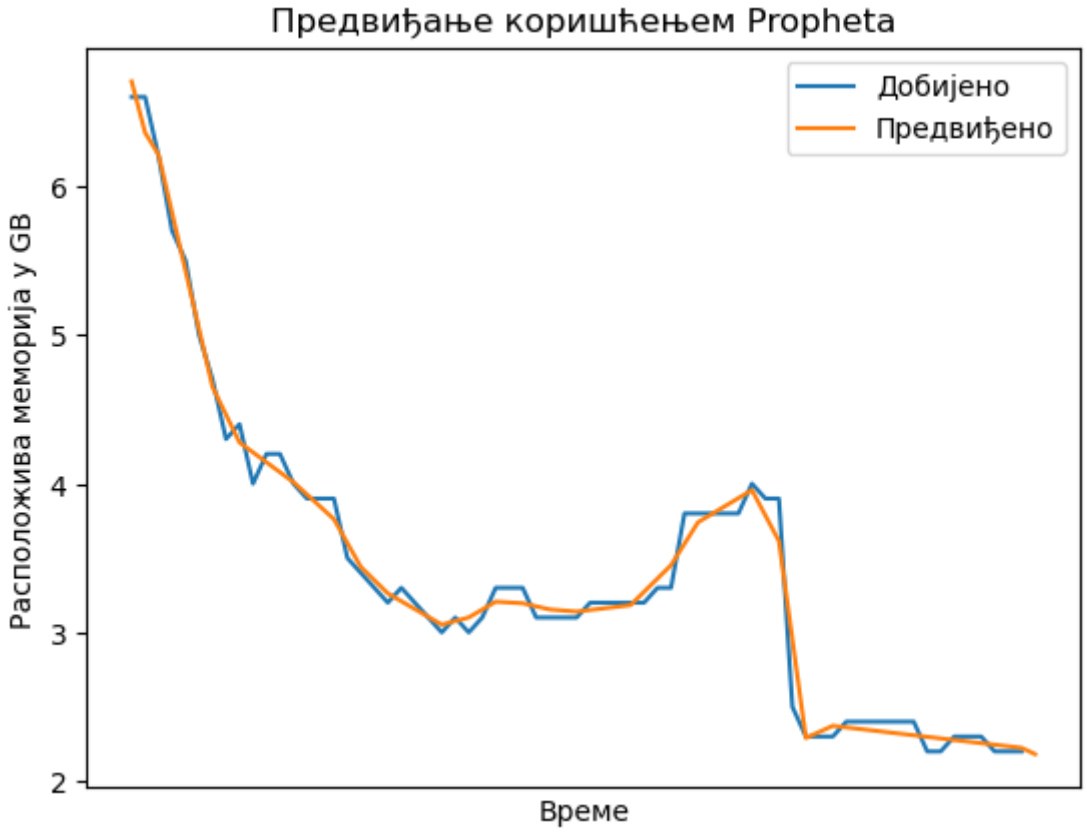
одлука, односно он представља једну ансамбл (ensemble) технику која омогућава комбиновање више стабала одлучивања [172]. Случајне шуме ширу примену налазе у проблемима који захтевају примену класификације, али се у домену временских серија алгоритам случајних шума може искористити и за задатке који подразумевају употребу регресије. Случајне шуме не захтевају обимнија подешавања и у већини случајева довољно је само прилагодити број стабала за употребу при решавању конкретног проблема [173]. У највећем броју случајева за број стабала (*n_estimators*) се узима 10 [167], па је сходно томе приликом моделирања усвојена та вредност и крајњи модел је реализован са том вредношћу, али су вршене пробе модела и са 100 и са 1000 стабала како би се утврдило понашање модела за далеко већи број стабала. Као и код претходно анализираног модела и овде је потребно прилагодити податке *MemLog* датотеке у складу са захтеваним начином самог модела. При коришћењу случајних шума предпроцесирање података добија мало другачији облик. Наравно први кораци ће бити идентични оним који су коришћени за припрему података у случају ARIMA моделирања, односно и овде ће се извршити учитавање података у складу са CSV конструкцијом датотеке, а потом и извршити одређене корективне радње као у случају претходног модела. Међутим, овде ће припрема података имати још један значајан корак који се односи на реструктуирање података који имају облик временске серије у податке намењене надгледаном учењу (supervised learning). То практично значи да ће забележена меморијске вредности у једном тренутку времена бити основ за предвиђање меморијске вредности забележене у наредном временском тренутку. Прва и последња забележена вредност ће се одбацити као неупотребљиве, будући да првој забележеној меморијској вредности не претходи ниједан податак, као и да после последње забележене меморијске вредности не следи ниједан податак. Одбацују се и све вредности које нису дате бројчано, односно тзв. NaN (Not a Number) вредности. За ове операције поред поменутог *pandas* модула, користе се и функције *numpy* модула [174] како би добијени подаци били трансформисани у одговарајући n-димензионални низ (*ndarray*). Добијени сет података се, потом, раздваја на два дела, односно два сета података који ће се користити за обуку модела, односно колоне које садрже улазе и излазе постају засебни сетови података, поштујући при томе наведено правило о изузимању прве и последње забележене вредности. Дефинисање модела врши се употребом *RandomForestRegressor()* функције која је дефинисана у оквиру *sklearn.ensemble* модула [175], при чему се функцији као параметар прослеђује број стабала (*n_estimators*). Овде треба напоменути да је прослеђивање броја стабала вршено употребом одговарајућег параметра у свим случајевима, укључујући и случајеве када се ради о подразумеваним (default) вредностима, будући да се подразумеване вредности могу разликовати у зависности од верзије *sklearn* модула који се користи приликом превођења [176], па се овим осигурава да се, без обзира на коришћену верзију модула, увек користи предвиђени број стабала. И у случају моделирања заснованог на алгоритму случајних шума, обука модела остварује се употребом *fit()* функције, с тим што се сад за остваривање обуке користе два сета података, један са улазним и један са излазним подацима. Након завршене обуке модела, конструише се сет улазних података који су, у складу са напред наведеним начином рада предметног модела, *ndarray* типа и над којим ће се вршити предикција. Само прогнозирање се и у овом случају успоставља позивом *predict()* функције. Приказ прогнозираних вредности у односу на конструисане улазне вредности заснованих на вредностима у оквиру *MemLog* датотеке, илустрован је графиком приказаним на слици 62. Идентично претходно разматраном моделу и овде се графичка интерпретација резултата ослања на функције *Pyplot* интерфејса *Matplotlib* модула.



Слика 62. Приказ резултата добијених коришћењем Random Forest модела

Последњи модел чија ће реализација и резултат бити разматрани јесте Prophet прогностичарски модел (Prophet Forecasting Model). Prophet модел и целокупно решење отвореног кода које га окружује, развијени су од стране Facebook компаније (сада компанија Meta) [168] као одговор на потребе обимних и учесталих анализа временских серија у оквиру задатака који се реализују за потребе друштвене мреже Facebook. Идеја овог модела је да се минимизује учешће корисника у подешавању великог броја својстава које се јављају приликом моделирања, односно да се постигне висок степен аутоматизације самог модела, како би се одговорило на реалне сценарије употребе када се реализује велики број предвиђања. Модел је базиран на адитивном моделу [177]. Prophet поседује развијене механизме који омогућавају да се модел прилагоди на појаву недостајућих података (missing data), а оно што га посебно карактерише јесте изузетна прилагодљивост на промене у тренду и појаву нестандардних вредности приликом анализа и предвиђања временских серија. И у случају рада са Prophet моделом, основно полазиште приликом примене модела јесте припрема података над којим ће се модел извршавати. Припрема података у овом случају извршена је коришћењем функционалности датих у оквиру *pandas* модула. Стандардно подаци из задате *MemLog* датотеке уносе се у одговарајући dataframe (користећи особености CSV формата у коме је датотека задата), а након извршеног уноса, примењују се стандардни поступци конверзије, који су већ претходно објашњавани, како би одговарајуће вредности, коришћене приликом процеса предикције, биле бројчаног типа. У односу на процес конверзије какав је коришћен код ARIMA модела, овде ће бити успостављен још један додатни корак а то је и конверзија датума и времена уноса података у датотеку како би временска карактеристика била у облику који користи Prophet. Ово се једноставно реализује коришћењем функције *to_datetime(data, format)* садржане у оквиру pandas

модула, где *data* представља податке о забележеном датуму и времену, а *format* облик у коме су ти подаци представљени. Када се дискутовало о MemLog датотеци објашњен је и начин уноса података о датуму и времену уписа, па ће сходно томе прослеђена информација бити *format='%y-%m-%d %H:%M:%S'*. Припрема модела врши се коришћењем *Prophet()* функције која се налази у оквиру *prophet* модула. Као што је већ наведено, сам модел поседује висок степен аутоматизације, па се припрема модела може обавити и без додатне параметризације. У овом конкретном случају моделу је ипак прослеђен параметар *interval_width = 0.95* како би се моделу конкретизовало да ради са ширином интервала несигурности од 95 % уместо стандардних 80 % . Као и у претходним моделима и у овом моделу обука модела омогућена је позивом *fit()* функције. Број остварених предвиђања у оквиру дефинисаног Prophet модела остварени су позивом функције *make_future_dataframe(periods,freq)*, при чему параметар *periods* дефинише колико прогноза ће бити остварено, док *freq* дефинише у којим временским интервалима ће бити предвиђања реализована (минутно, сатно, дневно, месечно, годишње и слично). Приликом тестирања модела у овом случају коришћено је минутно предвиђање (*freq="T"*). Након свих претходно извршених операција, предвиђања се остварују употребом функције *predict()*. Упоредни приказ предвиђања и реално добијених вредности из *MemLog* датотеке дат је графиконом приказаним на слици 63 реализованим функционалностима *Pyplot* интерфејса *Matplotlib* модула на идентичан начин оним коришћеним код претходна два приказана модела.



Слика 63. Приказ резултата добијених коришћењем Prophet модела

Како је већ наведено, одлука о коначном избору модела који ће се даље примењивати у реализацији решења за дати проблем може зависити од великог броја параметра који се дефинишу за сваки посматрани проблем појединачно. Пре доношења коначне одлуке о избору конкретног модела за примену над дефинисаним проблемом у оквиру докторске

дисертације, потребно је дати и одређене опсервације начињене током коришћења претходним модела и ширих тестирања која су вршена над њима, а која овде неће бити приказана како се не би непотребно ширила постојећа дискусија.

Приликом употребе ARIMA модела примећено је да се могу јавити одређени проблеми приликом предвиђања у случајевима наглих промена забележених вредности, док је примећено да модел случајних шума далеко боље реагује на ове нагле промене и даје реалистичније прогнозе. Некада је потребно много експериментисања са моделом док се не пронађу адекватне (p, d, q) вредности (order вредности) за које модел најбоље врши предвиђање што може унети додатну комплексност у самом решењу где се поменути модел примењује. Код модела случајних шума примећено је да се у појединим случајевима могу појавити проблеми са перформансама. Примећени су случајеви када је модел случајних шума захтевао значајно више времена за реализацију својих предиктивних радњи у односу на исти процес који се обављао коришћењем друга два модела. Овакве варијације у перформансама могу представљати значајан проблем када се предиктивне радње засноване на одређеном моделу интегришу у остатак решења. Код prophet модела примећено је да, иако постоји висок степен аутоматизације и да у већини случајева модел даје изузетно употребљиве резултате коришћењем подразумеваних вредности, ипак се бољи резултати постижу ако се поједини параметри модела додатно подесе мануелним путем. Овде треба напоменути да је подешавање параметара и трагање за најбољим вредностима далеко једноставније и брже него што је то случај са ARIMA моделом.

У претходним анализама коришћени су исти параметри графичког представљања добијених прогнозираних редултата сва три модела како би се лакше могли међусобно сагледати и упоредити. Како се може уочити са реализованих графика, сва три модела дају добре и употребљиве прогнозиране резултате, међутим, гледајући графиконе стиче се утисак да се најбоља и најсмисленија предикција врши коришћењем prophet модела.

Постоје различити приступи у оцењивању модела. У пракси је највише заступљено оцењивање модела на основу израчунавања неке од следећих вредности: средње квадратне грешке (Mean of the Square of Errors - MSE), или уместо ње на појединим местима коришћењем и корена средње квадратне грешке (Root of the Mean of the Square of Errors - RMSE), средње апсолутне грешке (Mean Absolute Error - MAE) и коефицијента детерминације (R-Squared score - R^2) [178]. Код поменутих модела вршен је прорачун средње апсолутне грешке, изражен као:

$$\mathrm{MAE} = \frac{1}{n} \cdot \sum_{i=1}^{n} |y_i - \hat{y}_i| \qquad (8)$$

где је

- n - број редова, односно извршених опсервација,
- y_i - стварна вредност и
- $\hat{y}_i$ - прогнозирана вредност.

Прорачун средње апсолутне грешке реализован је коришћењем функције *mean_absolute_error(y_true,y_predicted)*, дате у оквиру *sklearn.metrics* модула [179], при чему *y_true* представља стварне вредности, док *y_predicted* представља прогнозиране вредности. Израчунавање је извршено идентичним поступком за сва три приказана модела, а резултати су приказани у табели 16.

Табела 16. Вредности средње апсолутне грешке за анализиране моделе

Модел	Вредност MAE заокружена на три децимале
ARIMA	0,144
Случајне шуме	0,152
Prophet	0,079

За средњу апсолутну грешку важи правило да што је њена вредност приближнија нули то је модел тачнији. У складу са тим, посматрајући вредности средње апсолутне грешке приказане у табели, можемо рећи да су сва три модела испољила изузетну добру тачност. Међутим, ове вредности нису међусобно упоредиве будући да се ради о различитим моделима на основу којих је вршено предвиђање, а средња апсолутна грешка није упоредива над различитим моделима.

За потребе упоредивости модела може се искористи друга мера означена као средња апсолутна процентуална грешка (Mean Absolute Percentage Error - MAPE) која има облик дат на следећи начин:

$$\text{MAPE} = \frac{1}{\text{n}} \cdot \sum_{\text{i=1}}^{\text{n}} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \qquad (9)$$

Прорачун средње апсолутне процентуалне грешке реализован је коришћењем функције *mean_absolute_percentage_error(y_true,y_predicted),* дате у оквиру *sklearn.metrics* модула, при чему *y_true* представља стварне вредности, док *y_predicted* представља прогнозиране вредности. Израчунавање је извршено идентичним поступком за сва три приказана модела, а резултати су приказани у табели 17.

Табела 17. Вредности средње апсолутне процентуалне грешке за анализиране моделе

Модел	Вредност MAPE заокружена на три децимале	Вредност MAPE у % заокружена на три децимале
ARIMA	0,064	6,4
Случајне шуме	0,058	5,8
Prophet	0,024	2,4

И за средњу апсолутну процентуалну грешку важи правило да што је њена вредност приближнија нули то је модел тачнији. Уколико посматрамо процентуалну вредност, важи правило да уколико је вредност испод 50 %, модел је тачан, с тим да су најтачнији модели са вредношћу средње апсолутне процентуалне грешке мањом од 10 %, уколико је ова вредност између 10 % и 20 %, то су тачни модели, док су модели код којих је вредност између 20 % и 50 % прихватљиви. У складу са тим, посматрајући вредности средње апсолутне процентуалне грешке приказане у табели, можемо рећи да сва три модела испољавају изузетно високу тачност. При томе најтачнији се показао prophet модел.

Како би се омогућио још један увид у прогнозу коју остварују претходно приказани модели, навешће се још један специфичан податак карактеристичан за скуп података који је коришћен у овом случају, односно за податке садржане у оквиру *MemLog* датотеке. Неколико последњих уноса у оквиру поменуте датотеке имају идентичне забележене вредности од 2,2 GB . Без употребе модела и предикције остварене на основу ових модела, на основу пружених података, очекивало би се да прва наредна вредност која би се очитала и уписала у оквиру предметне датотеке буде око 2,2 GB, односно или једнака,

или нешто мало виша, или нешто мало нижа вредност. Употребом модела, остварене су прогнозе дате у табели 18.

Табела 18. Прогнозиране вредности анализираних модела

Модел	Прогнозирана вредност заокружена на две децимале
ARIMA	2,05
Случајне шуме	2,26
Prophet	2,18

Као што се из табеле види, prophet модел је прохнозирао вредност која је најближа 2,2 GB, мада ни други модели нису много одступили од те вредности. Ово се може сматрати још једном потврдом да се модели добро уклапају у очекиване резултате.

Узимајући у обзир све напред наведено, у реализацију решења које се представља овом докторском дисертацијом могао би се интегрисати било који од претходна три модела. Међутим, одлука је да се изврши интеграција prophet модела у предвиђено решење јер поред тога што резултати показују да је prophet модел пружио најбоље резултате над посматраним скупом података, током тестирања се показало да је prophet модел добар и по питању времена потребног за предвиђање, да даје јако добре прогнозе чак и са подразумеваним вредностима, да га није потребно често преподешавати, као и да када постоји потреба за додатним подешавањима, додатна подешавања се изводе на крајње једноставан начин. Такође, овај модел има изузетну добру софтверску подржаност реализовану као решење отвореног кода, а будући да је обезбеђен модул за интеграцију модела у решења заснована на Python програмском језику, то га чини подесним за интеграцију у решење развијено у оквиру предметне докторске дисертације. У складу са свим наведеним, све предиктивне радње у оквиру докторске дисертације засноване су на prophet моделу.

4.5. Прорачун вредности количине виртуелне радне меморије

Прорачун вредности количине виртуелне радне меморије настао је као резултат истраживања реализованих у оквиру докторске дисертације. Прво се дефинише коефицијент иницијализације као:

$$k_I = \frac{M_{hw}}{M_{gt}} - 1 \qquad (10)$$

где је:

- M_{hw} - гранична вредност количине слободне радне меморије дефинисана у односу на укупну количину физичке радне меморије и
- M_{gt} - количина виртуелне радне меморије која се додељује виртуелној машини приликом поступка иницијализације.

Вредности M_{hw} и M_{gt} дефинисане су у односу на детектовану укупну количину физичке радне меморије (доступна радна меморија рачунара) означене као M_{ht} . У складу са подацима приказаним у табели 14, коефицијент k_I ће узимати различите вредности приказане у табели 19 у зависности од вредности M_{ht} .

Табела 19. Могуће вредности k_I у зависности од вредности M_{ht}

M_{ht}	k_I
$M_{ht} \leq 4$ GB	0,5
4 GB < $M_{ht} \leq 6$ GB	0,25
6 GB < $M_{ht} \leq 8$ GB	0,167
8 GB < $M_{ht} \leq 12$ GB	0,125
12 GB < M_{ht} < 16 GB	0,083
$M_{ht} \geq 16$ GB	0,0625

Како би се одредила нова вредност количине виртуелне радне меморије (M_{gt}) која ће бити додељена виртуелној машини, морају се одредити одређене меморијске вредности везане за физички хардвер. Вредности које се одређују засноване су на претходно описаном предиктивном моделу на основу кога се налазе прогнозиране вредности засноване на подацима из претходно описаних датотека *MemLog* и *MemLogMean*. Одређивање ових вредности врши се у тренутку издавања команде за покретање виртуелне машине. Када се таква команда изда, пре стварног покретања виртуелне машине, извршиће се окончање тренутне сесије бележења и снимања одговарајућих података, односно заустављање одговарајућег сервиса који омогућава бележење тих података. Након заустављања сервиса, извршиће се одговарајући прорачун прогнозиране вредности засноване на подацима из *MemLog* датотеке (садржи појединачне вредности стања слободне радне меморије) и таква вредност биће сачувана као m_{MLpr}. Потом ће се извршити и одговарајући прорачун прогнозиране вредности засноване на посацима из *MemLogMean* датотеке (садржи средње вредности стања слободне радне меморије) и таква вредност биће сачувана као m_{MLMpr}.

На основу односа претходних двеју прорачунатих вредности, врши се прорачун могућег будућег стања слободне радне меморије m_{hap} као:

$$m_{hap} = \begin{cases} \frac{m_{MLpr} + m_{MLMpr}}{2} - 0,5 & \text{, ако је } m_{MLpr} > m_{MLMpr} \\ m_{MLpr} - 0,5 & \text{, ако је } m_{MLpr} \leq m_{MLMpr} \end{cases} \qquad (11)$$

Сада се могућа количина виртуелне радне меморије m_{gt} која се може доделити виртуелној машини може исказати као:

$$m_{gt} = \frac{m_{hap}}{k_I + 1} \qquad (12)$$

Као што је већ било речи раније и у оквиру виртуелне машине постоји сервис који бележи тренутна стања слободне радне меморије по истим принципима као што то ради и на стварном рачунару. Сервис се стартује одмах по подизању оперативног система на виртуелној машини, а рад сервиса завршава се оног момента када се изда команда за гашење виртуелне машине пре него што се почне са стварним процесом гашења. Када сервис добије команду за прекид рада, налази се минимална вредност m_{gamin} од свих забележених вредности у оквиру *MemLog* датотеке на виртуелној машини. Пре гашења виртуелне машине та вредност се преноси одговарајућом методом на физички рачунар и снима на исти у оквиру *MemLogVM* датотеке. Поменута вредност биће искоришћена у наредној сесији коришћења виртуелне машине како би се кориговала вредност количине

виртуелне радне меморије која ће се доделити виртуелној машини у циљу избегавања случајева предимензионисања система.

Ако се са M'_{gt} означи вредност додељене количине виртуелне радне меморије виртуелној машини током претходне сесије, за наступајућу сесију рада виртуелне машине доделиће се вредност количине виртуелне радне меморије M_{gt} дефинисане као:

$$M_{gt} = \begin{cases} \dfrac{m_{gt} + (M'_{gt} - m_{gamin})}{2} & \text{, ако је } m_{gt} > M'_{gt} - m_{gamin} \\ m_{gt} & \text{, ако је } m_{gt} \leq M'_{gt} - m_{gamin} \end{cases} \qquad (13)$$

Све величине које улазе у прорачун вредности количине виртуелне радне меморије која се додељује виртуелној машини чине меморијски буџет виртуелне машине. Меморијски буџет виртуелне машине исказује се табеларно, а у оквиру буџета уносе се ознаке, вредности и опис. Пример једног меморијског буџета виртуелне машине дат је у оквиру табеле 20.

Табела 20. Пример меморијског буџета виртуелне машине

Величина	Вредност	Опис
M_{ht}	7,77 GB	укупна радна меморија рачунара
k_I	0,167	коефицијент иницијализације
m_{MLpr}	2,18 GB	прогнозирана тренутна вредност доступне радне меморије рачунара
m_{MLMpr}	4,26 GB	прогнозирана средња вредност доступне радне меморије рачунара
m_{hap}	1,68 GB	прогнозирана вредност доступне радне меморије рачунара
m_{gt}	1,44 GB	количина виртуелне радне меморије која се може доделити виртуелној машини
M'_{gt}	3,19 GB	вредност додељене количине виртуелне радне меморије виртуелној машини током претходне сесије
m_{gamin}	1,88 GB	минимална вредност забележених стања доступне количине виртуелне радне меморије у претходној сесији виртуелне машине
M_{gt}	1,375 GB	количина виртуелне радне меморије која ће се доделити виртуелној машини за наступајућу сесију
		НАПОМЕНА: Количина виртуелне радне меморије која ће се доделити виртуелној машини биће 1,375 · 1024 MB = 1408 MB узимајући у обзир да виртуелна машина исказује вредности радне меморије у мегабајтима и да важи однос 1 GB = 1024 MB.

Приликом поступка иницијализације (виртуелна машина се стартује први пут, или је наступила битна измена у хардверској структури физичког рачунара) узимају се вредности приказане у табели 14. У свим осталим случајевима стартовања виртуелне машине вредности које се додељују базирају се на претходним прорачунима.

5. РЕАЛИЗАЦИЈА САМОАДАПТИРАЈУЋЕ ВИРТУЕЛНЕ МАШИНЕ

5.1. Припрема основне виртуелне машине

Претходна поглавља описала су неке техничке аспекте предложеног решења које се реализује у оквиру докторске дисертације. Описана парцијална решења потребно је интегрисати у једно целовито решење које ће се користити од стране студената за реализацију задатака предвиђених у оквиру наставних активности који се реализују у оквиру студијског програма који одређени студент похађа. Применљивост решења у домену високошколског образовања зависи, не само од добре реализације претходно описаних поступака, већ и од свих осталих пратећих активности као и саме интеграције добијених поступака у једно целовито применљиво решење које омогућава добијање задатих функционалности.

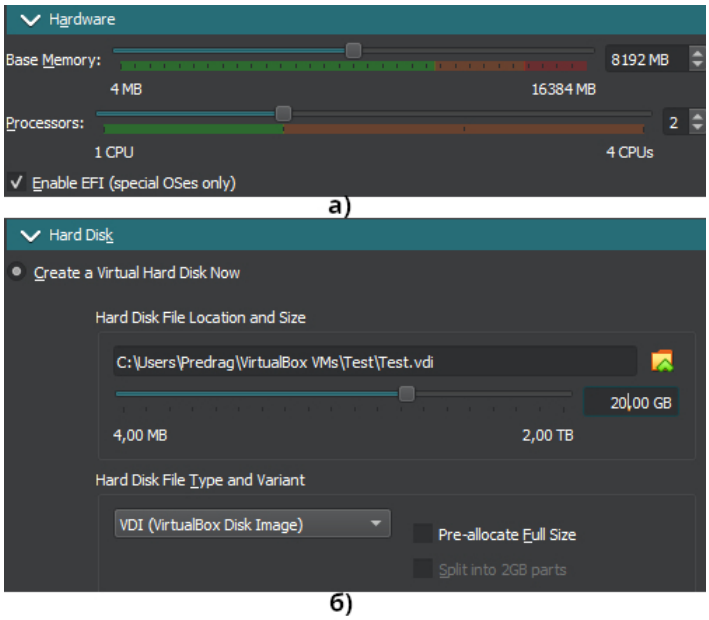
Како би се добило решење које испуњава задате критеријуме за сваког студената појединачно, основни корак који је потребно применити јесте припрема базичне виртуелне машине од стране особља високошколске институције која ће се касније преузимати од стране студената и за сваког студента адаптирати према његовим стварним потребама за остваривање учешћа у наставном процесу. При томе треба нагласити да припрема базичне машине може бити остварена како од ненаставног особља (на пример одговарајућа информатичка или техничка служба институције), или наставног особља (сарадници у настави, асистенти, наставници), а у појединим ситуацијама и од помоћног особља у настави (лаборанти, стручни сарадници, демонстратори). Заједнички именитељ, без обзира на то ко у наставном процесу реализује предметну виртуелну машину, јесте да особа која врши припрему исте мора поседовати широк обим знања из области рачунарске технике (виртуелизација, оперативни системи, рачунарски хардвер и многе друге области). Иако припрема базичне виртуелне машине изгледа као једноставан процес, он ипак испољава одређена специфична својства која треба узети у обзир како би се смањили каснији значајнији утицаји на целокупно решење.

Први задатак који треба остварити у целокупном процесу јесте избор оперативног система на коме ће рад виртуелне машине бити базиран. Како је већ речено у једном од претходних поглавља, основу виртуелне машине чиниће оперативни систем базиран на Linux окружењу из разлога што треба омогућити одговарајуће редистрибуирање виртуелне машине, а концепција Linuxa се уклапа у потпуности будући да је у оквиру лиценце омогућена његова редистрибуција без накнаде или додатних скривених трошкова. Постоји много Linux дистрибуција које се могу искористити за постизање задатих функционалности виртуелне машине. У оквиру рада на докторској дисертацији одлучено је да се виртуелна машина заснива на Linux Fedora Workstation верзије 38 [180]. Ова дистрибуција примењиваће се због чињенице да је подржана од једне од најстаријих и водећих компанија која се бави развојем Linux решења, а то је Red Hat, добро је документована, редовно ажурирана, има велику распрострањеност и заједницу корисника. Међутим и сама одабрана дистрибуција има више подверзија које се могу једнако добро искористити и поставља се питање коју одабрати за примену. Све подверзије су декстоп верзије са јасним, интуитивним и једноставним корисничким интерфејсима који ће омогућити једноставно сналажење студената. Зато се одлука на избор подверзије свела на постизање најбољих могућих перформанси целокупног система. У складу са тим анализирана је меморијска захтевност за најзаступљеније подверзије Linux Fedora Workstation. При анализи коришћени су подаци обајвљени у

[181] из којих се види да најмању меморијску захтевност показује Fedora LXDE [182] која користи само 318 MB радне меморије. Иако је приликом тестирања у оквиру виртуелне машине конзумација меморије износила нешто више, 386 MB радне меморије (вероватно из разлога што је тестирање у [181] вршено пре неколико година, а околности су се у међувремену промениле), та вредност је и даље најнижа у односу на остале тестиране подверзије, па је одлучено да се виртуелна машина у оквиру докторске дисертације заснива на Linux LXDE (Lightweight X11 Desktop Environment) 38. Ова верзија оперативног система уклапа се у циљеве докторске дисертације будући да поседује изузетне перформансе јер је замишљена да буде верзија која неће бити гломазна, већ базична, али врло употребљива и брза. Приликом тестирања у оквиру виртуелне машине брзина укупног стартовања оперативног система износила је одличних 12,885 s (резултати анализа обављених systemd-analyze алатом).

Када се говори о реализацији виртуелне машине, овде треба поменути да су сви параметри виртуелне машине сврстани у две основне категорије: променљиви и непроменљиви. Променљиви парамтери су они параметри који ће се променити приликом прве иницијализације виртуелне машине на крајњој дестинацији, односно на рачунару корисника исте (студента), о чему је већ било речи у претходном поглављу. Такви су на пример већ споменути параметри у претходном поглављу везани за рад виртуелне централне процесорске јединице, виртуелне радне меморије и виртуелног чипсета. Непроменљиви параметри су они који се неће мењати током рада виртуелне машине на крајњем одредишту и они остају са истим вредностима како су задати приликом креирања базне виртуелне машине од стране високошколске институције. Као пример могу се навести одређени параметри везани за реализацију виртуелног диска или виртуелног мрежног интерфејса. Овде треба напоменути да су одређени непроменљиви параметри могли бити сврстани у променљиве и њихово подешавање се могло поверити процесу прве иницијализације виртуелне машине на одредишту, међутим ово није урађено из неколико разлога. Прво, желело се скраћење процеса иницијализације на одредишном рачунару, тако да су строго изабрани параметри за којим постоји реална потреба промене на одредишту. Друго, креирањем врло уског скупа променљивих параметара тежи се ка минимазицији вероватноће појаве грешке у целокупном процесу, а сасвим сигурно уколико би се вршила иницијализација великог броја параметара на одредишту вероватноћа појаве грешке би драстично порасла. На крају треба поменути да се оваквим приступом постижу и одређене погодности за саме реализаторе базичних виртуелних машина иако можда делује да се напротив оваквим поступцима ствара додатно оптерећење. Особа која креира базичну виртуелну машину када постави непроменљиве параметре зна у сваком моменту вредност тих параметара без обзира у ком окружењу се у неком моменту у будућности та виртуелна машина буде користила. То је од великог значаја приликом дијагностичких поступака у случају настанка било какве спорне ситуације јер се скраћују дијагностичке процедуре услед константних вредности. Такође, евентуалне надоградње је врло лако извести уз константност ових параметара јер се смањује комплексност инсталација истих. У неким ретким случајевима уколико дође до неке неовлашћене или случајне промене ових параметара, услед њихове константне вредности враћање на подразумеване вредности ће бити много једноставнији процес за реализацију. Променљиви параметри поред тога што омогућавају постизање прилагођења виртуелне машине на реалне услове средине у којој она остварује свој рад, омогућавају и комфорнији процес креирања базичне верзије исте. На пример приликом креирања базичне виртуелне машине, уколико се тај процес обавља на рачунару који има 8 језгара и 64 GB радне меморије, могло би се виртуелној машини доделити свих 8 језгара и 32 GB меморије чиме би лице задужено за сам процес креирања имало врло комфоран рад на припремању те виртуелне машине не бринући о тим параметрима касније када је

потребно виртуелну машину спустити на ниво крајњег корисника јер ће се свакако касније извршити иницијализација ових параметара на услове крајњег одредишта. Пример променљивих и непроменљивих параметара дат је на слици 64 док је пример разврставања ових параметара приказан у табели 21.



Слика 64. Пример дефинисања а) променљивих и б) непроменљивих параметара виртуелне машине

Табела 21. Разврставање параметара виртуелне машине

Променљиви параметри	Непроменљиви параметри
- Количина радне меморије - Процесор - Чипсет	- Сви општи параметри (General) - Редослед и статус boot уређаја - TPM - Показивач - Проширене опције матичне плоче - Execution Cap - Проширене опције процесора - Сви параметри акцелерације (Acceleration) - Све опције приказа (Display) - Све опције уређаја за смештање података (Storage) - Све опције аудио уређаја (Audio) - Све опције мреже (Network) - Све опције серијски портова (Serial Ports) - Све опције USB - Све опције дељених фасцикли (Shared Folders) - Све опције корисничког интерфејса (User Interface)

Два кључна непроменљива скупа параметара која су доминантна приликом употребе виртуелне машине од стране студената за реализацију задатака везаних за реализацију наставног процеса у високошколском образовању јесу параметри виртуелног диска и параметри везани за виртуелни мрежни интерфејс. Они ће бити детаљније размотрени и анализирани у наставку.

Скуп параметара виртуелног диска у овом случају обухватиће три параметра: додељену укупну величину диска, избор типа виртуелног диска и одлуку да ли ће унапред доделити пуна величина диска или ће се та додела вршити динамичким путем, односно по потреби до достизања максималне вредности. Ови параметри се могу лако уочити у оквиру приказа датог под б) на слици 64. Прво питање које се овде поставља јесте како проценити величину виртуелног диска, односно како избећи две потенцијално проблематичне ситуације, прву када се виртуелни диск дефинише са јако скромним капацитетом и убрзо дође до његове попуњености и другу када се диск предимензионише и својим предимензионисањем у појединим ситуацијама може довести до угрожавања стварних ресурса на рачунару у оквиру ког виртуелна машина функционише. У складу са претходним, неопходно је прво урадити процену потребног капацитета диска, па тек онда дефинисати максимални капацитет који ће се доделити виртуелном диску. Процена се сагледава на основу неколико реалних чинилаца који утичу на заузетост сваког диска, било да се ради о физичком или виртуелном. Највећи део заузетости сваког системског диска огледа се у простору који је заузет од стране самог оперативног система, корисничких апликација и корисничких података. Међутим, овде треба имати природу употребе капацитета која није статичка, већ може значајно варирати у временском периоду. На пример може доћи до значајније надоградње оперативног система која може значајно додатно увећати простор који оперативни систем заузима на диску. Слично се може десити и са инсталираним апликацијама. Зато се, поред наведене иницијалне заузетости диска од стране оперативног система, корисничких апликација и корисничких података, мора увести и додатна корекција на исту у складу са претходно описаним могућим ситуацијама. Када се у обзир узме све напред наведено, одлучено је да се капацитет виртуелног диска, у оквиру ове докторске дисертације дефинише као:

$$KVD = (PZOS + PZA + PZKP) \cdot k \quad (14)$$

где је:

- KVD - укупни (максимални) капацитет виртуелног диска, исказан у гигабајтима
- PZOS - процењено заузеће диска од стране оперативног система, исказано у гигабајтима
- PZKA - процењено заузеће диска од стране корисничких апликација, исказано у гигабајтима,
- PZKP - процењено заузеће диска од стране корисничких података, исказано у гигабајтима и
- k - коефицијент корекције.

У овде разматраном случају процена је да ће оперативни систем заузети око 5 GB простора, корисничке апликације око 3 GB, док је процена да ће за корисничке податке бити довољно заузеће од 2 GB простора на посматраном диску. За коефицијент корекције узета је вредност 2. У складу са (14) биће:

$$KVD = (5\ GB + 3\ GB + 2GB) \cdot 2$$

$$KVD = 10\ GB \cdot 2$$

$$KVD = 20\ GB$$

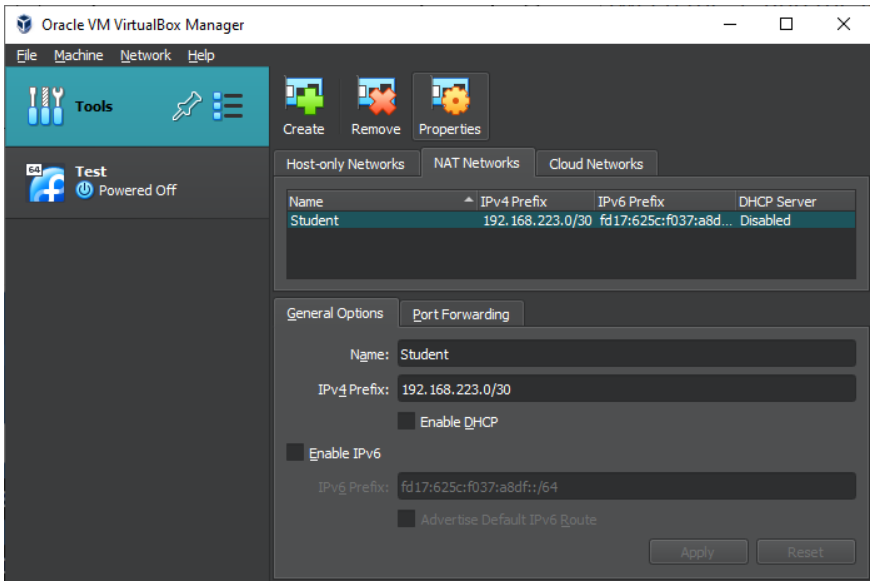
У складу са добијеним вредностима виртуелном диску ће је додељена вредност максималног капацитета од 20 GB. Следећи параметар који је потребно подесити јесте тип виртуелног диска. У оквиру хипервизора који се користи приликом реализације предметног решења описаног у докторској дисертацији на располагању су следећи типови виртуелног диска: VDI, VHD, VMDK, HDD, QCOW и QED. Без улажења у дубље анализе и објашњавања за сваки понуђени тип посебно, за тип хард диска овде је изабран VDI (Virtual Disk Image), будући да представља решење наменски развијано за употребу у окбиру хипервизора који се користи (VirtualBox). Остала решења примарно су развијана за неке друге хипервизоре, па се сматра да се најбољи степен интеграције постиже управо са изабраним VDI. Последњи параметар представља одлуку да ли ће се примењивати динамички или статички начин рада виртуелног диска, који је у претходном поглављу већ био објашњаван. Уколико се означи опција Pre-allocate Full Size на физичкој партицији ће VDI датотека која суштински представља виртуелни диск зазузети 20 GB простора, што у посматраном случају не чини адекватно решење из неколико разлога. Прво, беспотребно се заузима простор на физичкој партицији јер чак и у случају да се користи на пример само 10 GB простора на виртуелном диску, он ће ипак на физичкој партицији заузети 20 GB. То даље повлачи и импликацију да ће се приликом преузимања базичне виртуелне машине увек преузимати 20 GB података, односно крајњи корисник ће беспотребно преузимати виртуелну машину много веће величине него што је то у датом тренутку заиста и потребно. У динамичком систему рада, постоји значајније растерећење ресурса јер се не врши одмах алокација простора који одговара пуном капацитету виртуелног диска, већ се алоцирање врши по потреби у складу са стварним условима рада уз напомену да не постоји повратни процес. То практично значи да ако се неки простор алоцира једном, он се никада неће моћи више деалоцирати, односно смањење употребљеног простора на виртуелном диску не имплицира смањење физичке заузетости капацитета партиције. Динамички систем нуди мање оптерећење партиције, а при преузимању виртуелне машине од крајњих корисника чини да количина преузетих података буде значајно мања. Зато у овом случају опција Pre-allocate Full Size неће бити одабрана. већ ће се користити динамичка структура виртуелног диска.

Сам процес инсталације оперативног система у оквиру виртуелне машине неће бити посебно разматран и објашњаван. Поступак инсталације је идентичан поступку инсталације оперативног система на физичком рачунару, при чему је током процеса инсталације извршено аутоматско партиционисање диска, регионални параметри подешени су на Europe/Belgrade, дефинисано је корисничко име student за приступ систему и сам приступ се остварује без употребе лозинке. Напоменуће се само да је након зљавршеног поступка инсталације извршено комплетно ажурирање оперативног система на последњу верзију свих пакета који су укључени у оквиру истог, као и потпуно искључивање SELinux (Security Enhanced Linux) [183] заштитног система, будући да је за остваривање задатака који се стављају пред предметну виртуелну машину у потпуности довољан укључени firewall систем у оквиру оперативног система, док SELinux може проузроковати одређене проблеме и сметње приликом реализације задатака како у припреми саме виртуелне машине, тако и током њеног коришћења. Употреба SELinux заштитног система захтевала би напредна знања коришћења и администрације Linux система од стране свих учесника у процесу, како овлашћених лица високошколске институције, тако и крајњих корисника (студената).

Претходно је поменуто да је други кључан скуп непроменљивих параметара онај који је везан за подешавање виртуелног мрежног интерфејса. Остваривање мрежних функционалности виртуелне машине може се извршити на више доступних начина у оквиру хипервизора, али се они неће посебно разматрати. Потребно је нагласити да би се остварила адекватна мрежна повезаност виртуелне машине неопходно је извршити адекватна подешавања како са стране хипервизора, тако и са стране саме виртуелне машине. Са стране хипервизора за остваривање мрежних функција користиће се опција NAT Networks која свој рад на додељивању адреса заснива на принципима превођења мрежних адреса (Network Address Translation - NAT) [184]. Систем превођења функционише идентично систему превођења приватних у јавне адресе приликом реализације приступа интернету [185], с тим што ће се овде приватне адресе односити на домен виртуелне машине, док ће се јавне адресе односити на домен физичког хардвера, односно рачунара у оквиру ког се реализује посматрана виртуелна машина. Ово је могуће реализовати на два начина, динамичким путем, коришћењем DHCP (Dynamic Host Configuration Protocol) који омогућава аутоматско конфигурисање у оквиру IP (Internet Protocol) мреже, или статичким путем, где се користи мапирање 1-1 („један на један“), односно где се неће вршити аутоматско конфигурисање већ ће додељивање бити мануелним путем [186]. У овом случају, будући да се ради о једној виртуелној машини која ће бити имплементирана у оквиру одређеног одредишног рачунара, примениће се статички NAT. Такође, сво адресирање базираће се на коришћењу IPv4 мрежа. У складу са претходним, како би се обезбедиле неопходне мрежне функционалности на виртуелној машини, у оквиру хипервизора дефинисана је приватна мрежа са следећим параметрима:

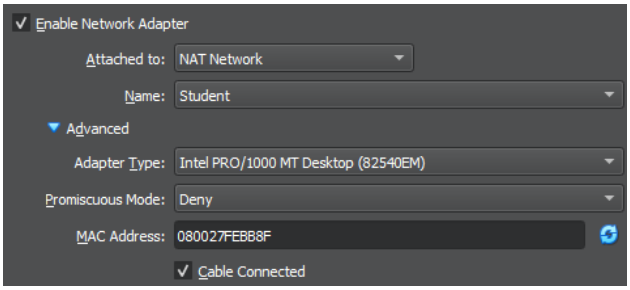
- име мреже: Student
- адреса: 192.168.223.0
- подмрежна маска: 255.255.255.252
- CIDR вредност: /30
- DHCP сервер: онемогућен
- IPv6 адресирање: онемогућено

Приказ овако задатих параметара у оквиру употребљеног VirtualBox хипервизора дат је на слици 65.



Слика 65. Задата NAT мрежа у оквиру VirtualBox хипервизора

Након обављених операције дефинисања мреже и њених параметара у оквиру хипервизора, потребно је дефинисати виртуелни мрежни адаптер. Постоји могућност избора неколико виртуелних мрежних адаптера, као и могућност дефинисања више виртуелних мрежних адаптера у оквиру једне виртуелне машине. У разматраном случају коришћења, у оквиру виртуелне машине дефинисаће се само један виртуелни мрежни адаптер и при томе користиће се подразумевани виртуелни мрежни адаптер (виртуелизација Intel PRO/1000 MT Desktop мрежног адаптера). Након омогућавања одговарајућег виртуелног мрежног адаптера он се повезује са креираном мрежом у оквиру хипервизора, у овом случају са дефинисаном NAT мрежом под именом Student. Приказ подешавања ових параметара у оквиру хипервизора приказан је на слици 66.

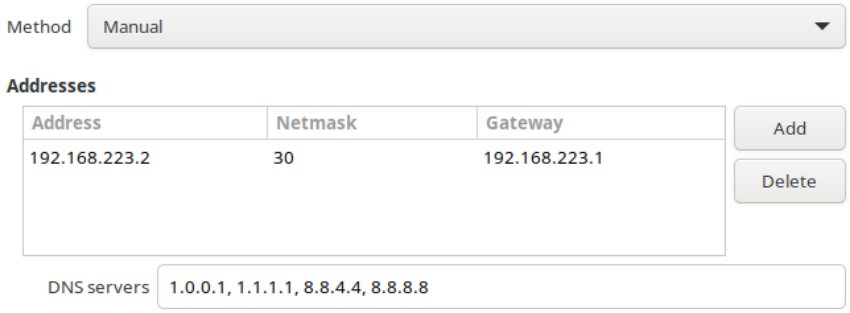


Слика 66. Подешавање виртуелног мрежног адаптера

Даља подешавања остварују се у оквиру оперативног система који је инсталиран у оквиру виртуелне машине. Приликом инсталације оперативног система, сама инсталација препознаје дефинисани виртуелни мрежни адаптер и укључује га у даљу инсталациону процедуру. Овде постоје два могућа приступа за подешавање виртуелног мрежног адаптера. Будући да је, како је већ речено, сам виртуелни мрежни адаптер препознат и укључен у процес инсталације оперативног система, подешавање се може остварити у самом инсталационом процесу. Уколико се подешавање не реализује у самом инсталационом процесу, могуће га је реализовати и накнадно, након извршене инсталације оперативног система. У случају који се разматра овде, подешавања параметара виртуелног мрежног адаптера остварена су након инсталације оперативног система и састоје се у избору начина подешавања, избору да ли ће се остваривати IPv6 адресирање, подешавања IPv4 адресе, подмрежне маске или CIDR, адресе мрежног пролаза, адресе DNS сервера који ће бити употребљавани и сличног.

У складу са претходним разматрањима, реализованом виртуелном мрежном адаптеру додељене су следеће вредности, приказане илустративно на слици 67:

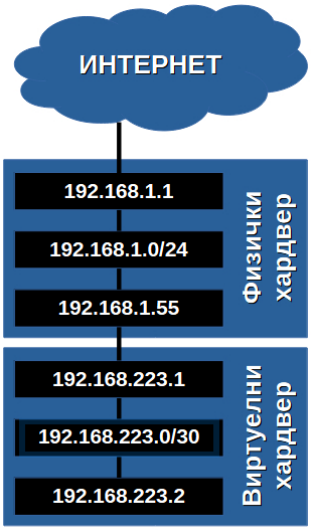
- метода додељивања адреса: мануелна (Manual)
- додељена адреса: 192.168.223.2
- подмрежна маска: 255.255.255.252
- CIDR вредност: /30
- адреса мрежног пролаза: 192.168.223.1
- IPv6 адресирање: онемогућено
- примарни DNS сервер: 1.0.0.1
- секундарни DNS сервер: 1.1.1.1
- терцијарни DNS сервер: 8.8.4.4
- квартарни DNS сервер: 8.8.8.8



Слика 67. Подешавање виртуелног мрежног адаптера у оквиру инсталираног оперативног система на виртуелној машини

Претходно описана реализација мрежних сервиса заснива се на коришћењу NAT сервиса који је могуће користити у оквиру задатог хипервизора. Овакав приступ, у односу на остале приступе који су доступни у оквиру хипервизора, одабран је из сигурносних разлога, будући да у посматраном решењу у оквиру ове докторске дисертације пружа највећу могућу сигурност. При томе та сигурност не значи да се губи нека од функционалности решења које се разматра. Виртуелна машина може по потреби комуницирати са одређеним сервисом покренутим над мрежним интерфејсом физичког рачунара, односно може приступити дефинисаном мрежном сервису на одређеној IP адреси физичког мрежног интерфејса. Дозвољена је оваква врста комуникације јер се може десити случај да на физичком рачунару буде покренут web сервер, систем за управљање базама података, или неки слични други систем коме треба приступити, па је сходно томе остављена могућност комуникације виртуелне машине са физичким рачунаром. Обрнути смер комуникације, односно комуникација физичког рачунара са виртуелном машином, могућ је искључиво уз дефинисање одговарајућих портова за прослеђивање (port forwarding). Комуникација између виртуелних машина је дозвољена, као и комуникација виртуелне машине са остатком мреже и комуникација виртуелне машине са Интернетом. Обрнуто, комуникација остатка мреже и Интернета са виртуелном машином могућа је искључиво уз дефинисање одговарајућих портова за прослеђивање (port forwarding).

Пример IP адресирања реализованог у претходно описаним разматрањима дат је на слици 68.



Слика 68. Пример реализованог IP адресирања виртуелне машине у односу на физички рачунар на коме се виртуелна машина налази

У конкретном примеру који се овде разматра, IP Адреса виртуелног мрежног адаптера задата је као статичка IP адреса 192.168.223.2 која се налази у оквиру приватне мреже дефинисане као 192.168.223.0/30. На адреси 192.168.223.1 врши се превођење адреса за претходно дефинисани мрежни опсег у приватну адресу 192.168.1.55 која представља статичку IP адресу физичког рачунара на ком је реализована виртуелна машина. Ова адреса представља адресу дефинисану у оквиру приватне мреже 192.168.1.0/24. Превођење претходне статичке IP адресе у јавну IP адресу која ће се користии приликом приступа интернету врши се на адреси 192.168.1.1. Овим поступком виртуелна машина остварује повезаност са Интернетом. Ту повезаност је битно остварити како би се могли реализовати сервиси неопходни за остваривање самоадаптивности саме виртуелне машине који се описују у оквиру предметне докторске дисертације, а такође, битни су и за процедуре инсталације софтвера, као и за остале неопходне процедуре које захтевају повезаност са Интернетом, а битне су за целокупно остваривање задатака у оквиру наставних процеса високошколског образовања. Овде се због илустрације приказа користила конкретна IP адреса физичког рачунара (192.168.1.55). У стварности ова IP адреса варираће од случаја до случаја у зависности од конфигурације одредишног рачунара, као и конфигурације саме локалне мреже и интернет приступа на одредиште. Без обзира на то принцип повезивања остаје исти јер ће увек физички рачунар поседовати неку приватну IP адресу и увек ће се морати вршити превођење те приватне адресе у јавну услед остваривања приступа интернету.

Како би се реализовале проширене могућности виртуелне машине које омогућавају реализацију механизама неопходних за постизање дефинисаних поступака адаптације, у саму виртуелну машину пре дистрибуције до крајњег одредишта морају бити уграђени сви неопходни софтверски елементи. Сав неопходан софтвер који се уграђује у виртуелну машину и који се користи у адаптивним поступцима, сакривен је од крајњег корисника. Софтвер се копира на виртуелни диск предметне виртуелне машине на путањи */VMAdaptive/* са које се врши његово стартовање без обзира да ли се ради о интерном стартовању (стартовање командом са виртуелне машине), или је команда за покретање софтвера издата екстерним путем (команда прослеђена са физичког рачунара). У оквиру ове фасцикле на виртуелном диску налазе се следећи садржаји:

- извршна датотека неопходна за рад сервиса за праћење стања расположиве виртуелне радне меморије и документовање истих у оквиру MemLog датотеке (сервис MSDCS),
- MemLog датотека која се креира од стране MSDCS сервиса,
- софтвер за омогућавање рада са развијеном онтологијом,
- помоћне скрипте.

Сви садржаји, сем помоћних скрипти, су детаљније разматрани у претходним поглављима. Такође, разматран је и начин имплементације сервиса у оквиру Linux окружења, који зависи искључиво од самог оперативног система, а не и од врсте хардвера над којим је тај оперативни систем реализован, па је претходно описан поступак идентичан и једнако примењив и на физички рачунар и на виртуелну машину. Како би се избегло понављање одређених дискусија изнети у оквиру докторске дисертације, у складу са претходним, овде ће се дати само краће појашњење у вези садржаја поменутих помоћних скрипти.

У овом случају помоћне скрипте представљају shell скрипте Linux базираног оперативног система. Ове скрипте имају могућност покретања разних апликација, аутоматизације активности које се периодично извршавају, администрирања система и применљивост у многим другим активностима где је потребно реализовати неки

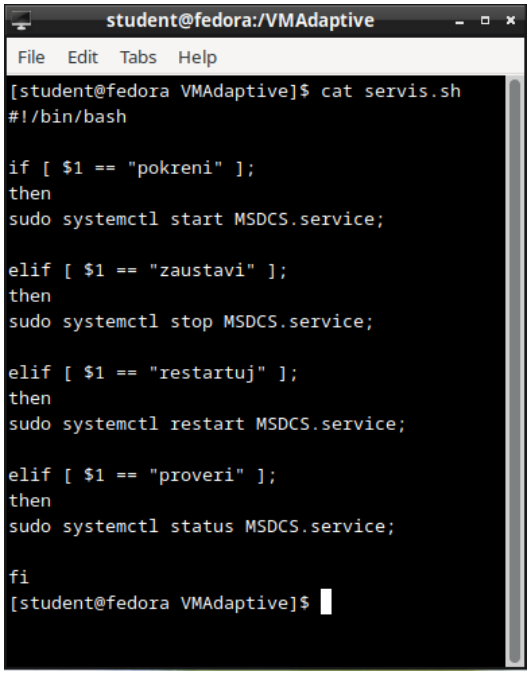
специфичан скуп команди Linux система. Скрипте се покрећу у оквиру shella, будући да shell омогућава директну интеракцију са оперативним системом кроз разноврсне активности попут читања текста и рашчлањивања унетих команди, обраде улазно-излазних преусмеравања, усмеравања, позадинске обраде, обраде сигнала, покретања програма за извршење и сличног [187]. Скрипте реализоване у оквиру овог доктората писане су за Bash (Bourne Again SHell) [188], који у коришћеној Linux дистрибуцији представља основни shell оперативног система и који представља sh-компатибилан shell. Скрипте написане за извршавање у оквиру поменутог shella представљају датотеке означене екстензијом .sh и у оквиру поменуте Linux дистрибуције могу се покретати реализацијом команде *bash ime_skript_datoteke.sh [argumenti]* или *./ime_datoteke.sh [argumenti]*. Овде ће се користити други облик за покретање, будући да ће команда за покретање скрипти бити издавана екстерно, односно са рачунара на ком је покренута виртуелна машина, па се тежи ка смањивању броја аргумената који ће бити прослеђиван приликом покретања, а у првом облику ће се име скрипт датотеке рачунати као још један додатни аргумент. У оквиру предметне докторске дисертације скрипте су реализоване због уочених проблема издавања команди екстерним путем. Под Linuxом постоје команде које у једној команди комбинују већи број извршних датотека са већим бројем аргумената што не представља проблем када се команде реализују у оквиру терминала самог оперативног система, али постаје проблем када се команда прослеђује екстерно, односно када се облик команде задаје са физичког рачунара, а потом ту команду задатог облика треба применити у оквиру оперативног система виртуелне машине. Како би се превазишли постојећи проблеми, написане су одговарајуће скрипте које такве команде редукују и своде их на следећи облик позивања *putanja/./ime_skript_datoteke.sh argument*. Као пример редуковања команди путем скрипта навешћемо реализацију команди за покретање, заустављање, рестартовање и проверу статуса MSDCS сервиса који је реализован у оквиру виртуелне машине за потребе документовања поменутих стања радне меморије. У табели 22 дат је упоредни приказ команди које би се стандардно користиле у терминалу виртуелне машине и команди које су реализоване за исти поступак одговарајућом скриптом.

Табела 22. Преглед команди за управљање радом реализованог MSDCS сервиса

Опис	Стандардна команда	Команда уз употребу скрипте
покретање сервиса	sudo systemctl start MSDCS.service	/VMAdaptive/./servis.sh pokreni
заустављање сервиса	sudo systemctl stop MSDCS.service	/VMAdaptive/./servis.sh zaustavi
рестартовање сервиса	sudo systemctl restart MSDCS.service	/VMAdaptive/./servis.sh restartuj
статус сервиса	sudo systemctl status MSDCS.service	/VMAdaptive/./servis.sh proveri

Када би покретали екстерним путем, на пример, команду за покретање сервиса, тада би први део команде sudo био програм који се позива, односно био би наведен са својом комплетном путањом и позван као /bin/sudo, systemctl програм би био означен као први аргумент, команда за покретање сервиса start као други аргумент и сама ознака сервиса MSDCS.service као трећи аргумент што је јако комплексан приступ и зна понекада због великог броја аргумената наићи на грешке приликом обраде овако формулисаног захтева. Употребом скрипта то се редукује на позив програма и једног његовог аргумента, односно /VMAdaptive/./servis.sh би био програм који се покреће, а pokreni би

био аргумент. Овакав приступ је далеко делотворнији за екстерно деловање приликом покретања одговарајућих процеса над самом виртуелном машином. Скрипт датотеке нису гломазне, ефективне су, не захтевају прављење извршне датотеке, лако се покрећу и разумљиве су као што говори пример употребљене скрипт датотеке servis.sh приказане на слици 69.

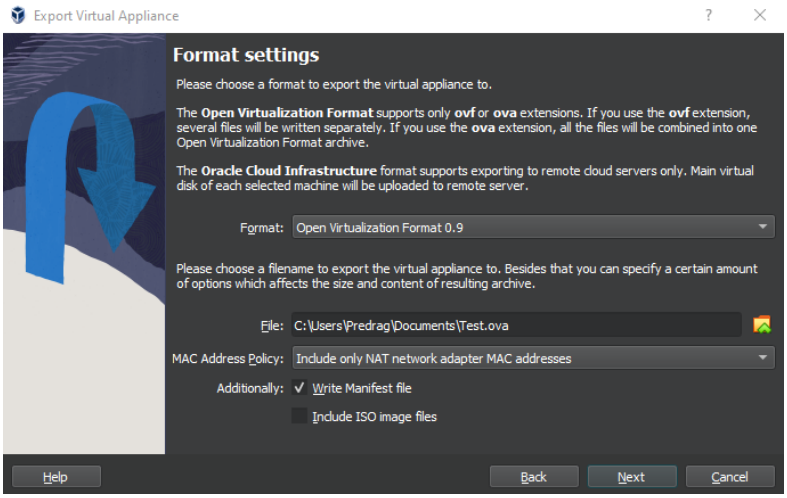


Слика 69. Терминалски приказ скрипт датотеке servis.sh

Базна виртуелна машина сматра се припремљеном за даљу редистрибуцију до крајњих дестинација оног момента када је су на њој подешени сви параметри, инсталиран и ажуриран одговарајући оперативни систем, покренути одговарајући сервиси и постављени сви програми и све скрипте неопходне за даље поступке обебеђивања њеног нормалног рада и адаптивности приликом рада на одредишту. Треба напоменути да је овде приказан један скраћени процес припремања базне виртуелне машине како не би разматрања приказана у оквиру докторске дисертације била преобимна. Нису разматрани многи администраторски поступци који се морају спровести како би се обезбедила пуна функционалност виртуелне машине, али то не значи да се они смеју изоставити. Свакако, додељивање одређених привилегија над одређеним датотекама и фасциклама, подешавање одговарајућих параметара десктоп окружења, подешавање firewalla, локализација, додавање српске ћирилице и латинице у опције избора тастатуре и многи други поступци и подешавања се морају спровести како би се студентима омогућило добијање крајње функционалног окружења. Реализовање базне виртуелне машине представља један задатак у који су укључени многи процеси, задаци и процедуре, а овде су наведени само најрелевантнији за приказ у оквиру теме докторске дисертације.

Након завршетка припреме базне виртуелне машине, поступак њене припреме за преузимање на крајњим одредиштима није окончан. Сама виртуелна машина састоји се од неколико датотека које представљају виртуелни диск, опис виртуелне машине, стање појединих параметара и остала својства. Дистрибуција појединачних датотека не би била пожељна будући да би се морала преузимати појединачно свака датотека која улази у састав виртуелне машине. Ово би се могло решити обједињавањем предметних датотека

у оквиру једне архиве употребом неког архивера и распакивањем преузете архиве на одредишту, али се над овако преузетом виртуелном машином не би могли спровести аутоматизовани поступци њеног увоза у хипервизор инсталиран на одредишном рачунару. У оквиру хипервизора постоји дефинисана процедура извоза виртуелне машине и применом те процедуре виртуелна машина се може дистрибуирати као једна датотека и тако извезена виртуелна машина на крајњем одредишту се може на једноставан начин увести у хипервизор коришћењем интегрисаних аутоматизованих процедура. За редистрибуцију виртуелне машине користи се посебан Open Virtualization Format (OVF) [189] који подржава рад са .ovf и .ova екстензијама. Уколико се користи .ovf екстензија неколико датотека се ипак формирају засебно, док се коришћењем .ova екстензије све датотеке комбинују у оквиру једне датотеке која представља Open Virtualization Format (OVF) архиву. Зато ће се у оквиру овде докторске дисертације за процедуре извоза, увоза, као и редистрибуирања виртуелне машине користити искључиво .ova екстензија. Сама процедура извоза виртуелне машине је крајње поједностављена у оквиру самог хипервизора као што се види из приказа датог на слици 70.



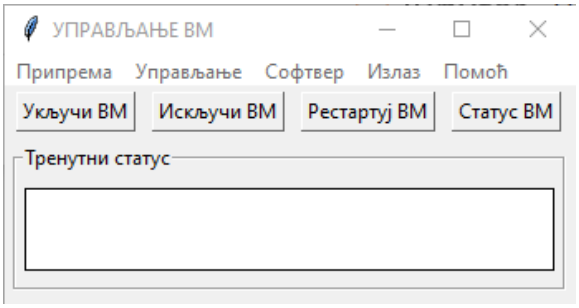
Слика 70. Поступак извоза виртуелне машине

Датотеку која представља извезену виртуелну машину треба сместити на место које је доступно путем Интернета за крајњег корисника, како би виртуелна машина могла бити преузета од крајњег корисника и омогућен њен трансфер на одређени одредишни рачунар.

5.2. Софтвер за управљање виртуелном машином

На одредишном рачунару где ће виртуелна машина бити покретана и где ће се користити приликом обављања задатака из наставног процеса високошколског образовања од стране студента, мора постојати интегрисано решење за управљање виртуелном машином. Ово не значи да на одредишном рачунару не мора постојати инсталиран хипервизор. Напротив, као што је речено, без присуства хипервизора не може се направити одговарајућа повезаност физичког рачунара и виртуелне машине. Међутим, сам хипервизор нема одговарајуће адаптивне механизме који су развијени и приказани у оквиру ове докторске дисертације, па сходно томе не може се вршити покретање и примена тих адаптивних механизама интерно у оквиру хипервизора већ се они примењују екстерним путем. У оквиру хипервизора се могу реализовати основне операције манипулације са виртуелном машином, стартовање, заустављање, увоз/извоз,

промена параметара виртуелног хардвера, али би то укључивало додатни напор за самог корисника, односно студента. Студент би морао бити упознат детаљно са концептом виртуелизације и виртуелних машина, радом и коришћењем хипервизора, подешавањем виртуелног хардвера, отклањањем грешака у раду и сличним процедурама што студента може поприлично удаљити од позитивног остваривања предвиђених исхода учења. Поред тога, студент би морао да познаје тачне процедуре коришћења виртуелне машине, када треба применити који корак приликом рада, када се примењује хипервизор, када се примењују екстерна решења, којим редоследом, да ли се у појединим ситуацијама неки од корака примењује или не. Овакво деловање би повећало комплексност самог решења, довело би до појаве збуњености код студента приликом коришћења истог, као и до појаве грешака приликом употребе виртуелне машине. Честе би биле ситуације у којима би студент прескочио неки од корака, или би их реализовао погрешним редоследом, а често студент не би био ни свестан да је направио неку од грешака приликом спровођења одређене процедуре. Зато се управљање задацима које обављају реализована екстерна решења, било да се налазе у оквиру виртуелне машине, или у оквиру самог физичког рачунара, као и управљање основним задацима самог хипервизора, интегришу у оквиру једног централизованог софтвера у оквиру физичког рачунара чији је задатак интегрисано управљање виртуелном машином. Основни изглед поменутог реализованог софтвера приказан је на слици 71. Приказан софтвер развијен је за сва три водећа оперативна система (Windows, Linux и macOS) и на сваком оперативном систему има скоро идентичан изглед (постоје благе варијације сходно самом графичком окружењу које се користи од стране оперативног система), спроводи идентичне процедуре и реализује идентичне функционалности.

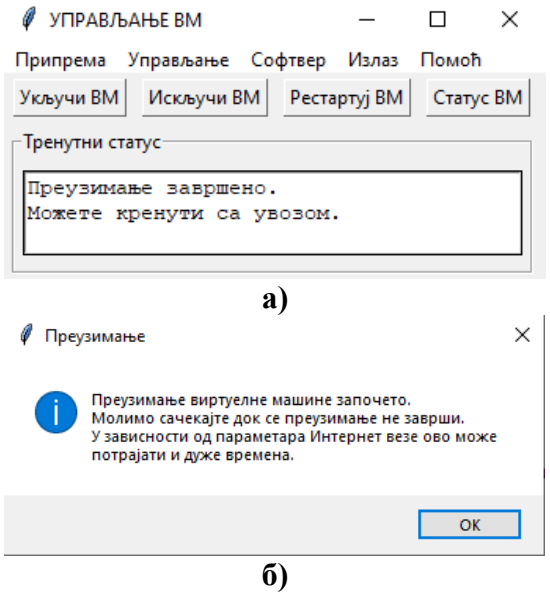


Слика 71. Основни изглед софтвера за интегрисано управљање виртуелном машином.

Као што се види из приказа датог на горњој слици, софтвер се визуелно састоји од три компоненте. Прва компонента је падајући мени који се налази у гоњем делу прозора и пружа приступ свим функцијама софтвера, како основним, тако и проширеним. У средишњем делу смештена је дугмад која реализује најчешће коришћене операције над виртуелном машином. Доњи део прозора намењен је за приказ различитих статусних порука у зависности од примењене операције у самом софтверу. Овде треба додати да софтвер у појединим операцијама користи и искачуће прозоре са обавештењима када се кориснику треба указати на неки специфични детаљ неке операције која је покренута у оквиру разматраног софтвера. Пример приказивања статусне поруке приказан је под а) , док је пример приказивања поруке применом искачућег прозора приказан под б) на слици 72.

Приликом развоја графичког корисничког интерфејса примењен је минималистички приступ како би употреба развијеног софтвера била крајње једноставна, разумљива и функционална у домену употребе виртуелних машина. Развој графичког корисничког интерфејса у потпуности је урађен коришћењем *tkinter* пакета који представља интерфејс за Tcl/Tk GUI омогућен на Windows, Linux и macOS платформама [190]. Иако се изглед

можда не уклапа у савремене трендове графичког дизајна примењених код оперативних система и њихових апликација, претходни интерфејс изабран је за коришћење у развоју предметног софтверског алата због своје доказане универзалности, стабилности, једноставности, стандардизованости у оквиру Python програмског језика и због својих добрих перформанси јер је намењен искључиво дефинисању графичког корисничког интерфејса [191]. Поред тога, предност се огледа и у могућности мултиплатформског развоја, који је битна карактеристика неопходна за задовољавање свих аспеката понуђеног решења.



Слика 72. Пример приказивања порука путем
а) статусних порука и б) искачућих прозора

Величина прозора апликације реализована је на 345x130 пиксела (подешено *geometry()* функцијом) са могућносшћу проширења до максималних 345x155 пиксела (подешено *maxsize()* функцијом) што је сасвим довољно за јасан приказ свих компоненти. Оваква величина прозора одабрана је и како би се добио јасан приказ и на системима који користе за приказ ниже резолуције, чији број није занемарљив нарочито међу популацијом студената. Креирање основног менија извршено је употребом функције *Menu()*. У оквиру главног менија реализовани су падајући подменији употребом функције *add_cascade()*, док су команде, било да су реализоване на највишем нивоу у главном менију, било да су реализоване у оквиру падајућих подменија, креиране употребом функције *add_command()*. Како би се добила одговарајућа визуелна допадљивост, у оквиру падајућих подменија извршено је раздвајање команди употребом предефинисаног сепаратора функцијом *add_separator()*. Реализација дугмади извршена је коришћењем *Button()* функције. Статусне поруке приказују се употребом *Text()* функције, док се поруке које се приказују искачућим прозорима приказују *messagebox.showinfo()* функцијом будући да су све поруке информативног садржаја. Распоређивање елемената у оквиру прозора извршено је употребом управљача распоредом уз помоћ мреже (Grid Layout Manager) који прави размештај свих елемената у прозору користећи дводимензионалну табелу (мрежу). Положај елемента дефинисан је бројем реда и колоне, при чему је положај прве ћелије табеле која се користи за смештај елемента дефинисана као (0,0). Треба нагласити да један елеменат може заузети и више ћелија при чему се дефинише положај прве ћелије од које почиње заузимање положаја,

као и број ћелија у реду и колони које ће наведени елемент заузети. Распоред се остварује кроз позив *grid()* метода. Поред употребе управљача распоредом елемената, у циљу боље организације приказа елемената извршено је груписање сродних елемената употребом frame контејнера који се остварују кроз *Frame()* позиве. На овај начин остварена је боља прегледност и лакше сналажење корисника унутар саме апликације. Одговарајуће команде менија и подменија, као и дугмади остварују се кроз задате корисничке функције.

Основне функционалности које се остварују развијеним софтвером подељене у три главне категорије дате су као:

- Припремне активности (подмени *Припрема*):
  - преузимање виртуелне машине (команда *Преузимање ВМ*),
  - увоз виртуелне машине (команда *Увоз ВМ*) и
  - иницијализација виртуелне машине (команда *Инцијализација ВМ*),
  - иницијализација виртуелне мреже (команда *Иницијализација мреже*);
- Управљање виртуелном машином (подмени *Управљање* и дугмад):
  - стартовање виртуелне машине (команда *Укључи* и дугме *Укључи ВМ*),
  - гашење виртуелне машине (команда *Искључи* и дугме *Искључи ВМ*),
  - рестартовање виртуелне машине (команда *Рестартуј* и дугме *Рестартуј ВМ*) и
  - провера тренутног статуса виртуелне машине (команда *Статус* и дугме *Статус ВМ*);
- Управљање инсталацијом софтвера на виртуелној машини (подмени *Софтвер*):
  - идентификација студента путем броја индекса (команда *Унос броја индекса*) и
  - инсталација софтвера коришћењем развијене онтологије (команда *Инсталација софтвера*);
- Напуштање програма (команда *Излаз*) и
- Кратке информације о програму (команда *Помоћ*).

Делови функционалности који се остварују као функције хипервизора остварују се коришћењем програма VBoxManage који се испоручује у саставу VirtualBox хипервизора, коришћеног у докторској дисертацији. VBoxManage се налази у саставу VirtualBoxа без обзира да ли се ради о Windows, Linux или macOS инсталацији. Међутим, овде треба напоменути да се начин стартовања разликује међу оперативним системима сходно именовању извршне датотеке у оквиру оперативног система. У Windows системима то именовање је VBoxManage.exe, док је у Linux и macOS системима то именовање дато као vboxmanage. У даљем тексту користиће се облик команди намењен Linux системима, који је идентичан и за употребу на macOS системима, док за Windows системе треба променити именовање извршне датотеке сходно претходном запажању.

Преузимање виртуелне машине остварује се кроз подмени *Припрема* избором команде *Преузимање ВМ*. Како је већ претходно наведено, базна виртуелна машина се на крају поступка поставља на одређено доступно место на Интернету одакле је могуће њено преузимање и трансфер до одредишта, односно до рачунара студента. Ове активности реализоване су употребом *urllib.request* модула [192]. Пре почетка процеса преузимања, врши се позив функције *urlcleanup()* како би се осигурало да не постоје неке

привремене датотеке од претходних преузимања које могу на пример остати у систему у ситуацији да је претходно дошло до неуспешног преузимања и да се преузимање поново стартује. Само преузимање реализује се кроз функцију *urlretrieve(src,dst)*, где аргумент *src* означава локацију базне виртуелне машине за преузимање (комплетна интернет путања заједно са именом OVF архиве), док аргуменат *dst* означава одредиште где ће преузета виртуелна машина бити сачувана по окончању процеса преузимања (локална путања заједно са именом OVF архиве). Приликом стартовања процеса преузимања базне виртуелне машине, кориснику се приказује адекватна информација о започетом процесу путем искачућег прозора, док се по окончању процеса преузимања, кориснику приказује адекватна информација о окончању процеса кроз статусну поруку, у за то предвиђеном простору, у самом прозору софтвера.

Увоз виртуелне машине остварује се кроз подмени *Припрема* избором команде *Увоз ВМ*. Овај процес спроводи се над преузетом базном виртуелном машином што значи да се може покренути тек по успешном окончању процеса преузимања. Процес је намењен увозу преузете базне виртуелне машине у хипервизор на одредишном рачунару како би се могли спроводити даљи поступци над виртуелном машином у оквиру хипервизора. Процес увоза реализује се позивом *run()* функције *subprocess* модула којој се параметарски прослеђује следећи облик vboxmanage команде на извршење:

vboxmanage import OVF_arhiva.ova

Приликом стартовања процеса увоза виртуелне машине корисник се информише о покретању и својствима истог кроз садржај искачућег прозора, док се по окончању целокупног процеса корисник информише о истом кроз одговарајућу статусну поруку.

Иницијализација виртуелне машине остварује се кроз подмени *Припрема* избором команде *Иницијализација ВМ*. Ова команда покреже извршну датотеку *vhinit* и користи се без обзира да ли се ради о примарној или секундарној иницијализацији виртуелне машине. Сам процес иницијализације, као и употреба извршне датотеке *vhinit*, објашњени су у претходном поглављу. По завршетку иницијализације приказује се одговарајућа статусна порука.

Иницијализација виртуелне мреже остварује се кроз подмени *Припрема* избором команде *Иницијализација мреже*. Иницијализација мреже није обухваћена програмом vhinit, будући да се врши само једном, односно постоји само примарна иницијализација виртуелне мреже. За секундарном иницијализацијом не постоји реална потреба јер чак и случају промене физичког хардвера не постоји променљивост параметара задате виртуелне мреже. У складу са овим својствима, иницијализација виртуелне мреже је издвојена као посебан процес. Извршење се врши следећим скупом vboxmanage команди:

vboxmanage natnetwork add --netname Student --network 192.168.223.0/30 --enable --dhcp off --ipv6=off

VBoxManage.exe natnetwork start --netname Student

које се као аргумент прослеђују *subprocess.run()* функцији. Прва команда врши иницијализацију параметара виртуелне мреже, док друга команда задату мрежу чини доступном за коришћење у оквиру хипервизора, а самим тим и задате виртуелне машине. По окончању целокупног процеса приказује се одговарајућа статусна порука о успостављању виртуелне мреже задатог имена.

Стартовање виртуелне машине представља процес који је могуће извршити једино у случају да је поступак увоза виртуелне машине у сам хипервизор окончан правилно и у потпуности. Покретање виртуелне машине може се обавити на два начина, коришћењем команде *Укључи* из подменија *Управљање*, или коришћењем дугмета *Укључи ВМ*. Оба поступка су међусобно равноправна, са истим процедурама и идентичним остварењем крајњег резултата. Стартовање виртуелне машине у оквиру команди које се задају путем реализованог софтвера обухвата већи број корака који омогућавају адаптивност виртуелне машине, пре него што се зада последњи корак у оквиру процедуре а то је стварно стартовање виртуелне машине из задатог хипервизора.

Први корак који се задаје јесте заустављање сервиса који прати тренутне вредности доступне радне меморије. У зависности од оперативног система физичког рачунара то се постиже процесирањем неке од следећих команди у оквиру оперативног система:

- *MSDCS stop* уколико се ради о Windows систему,
- *sudo systemctl stop MSDCS.service* уколико се ради о Linux систему и
- *launchctl stop MSDCS* уколико се ради о macOS систему.

За Windows системе процесирање команде реализује се коришћењем функције *windll.shell32.ShellExecuteW()* из *ctypes* модула [193] како би се коришћењем аргумента runas обезбедиле потребне администраторске привилегије за заустављање рада сервиса. За Linux и macOS системе ово процесирање извршава се коришћењем *subprocess.run()* функције.

Након заустављања поменутог сервиса врши се читање укупне количине радне меморије рачунара и на основу додељене вредности врши се формирање коефицијента k_I . Потом се врши налажење прогнозираних података на основу података садржаних у датотекама *MemLog* и *MemLogMean*. На основу добијених вредности врши се прорачун могућег будућег стања слободне радне меморије и добијени резултат се укључује у прорачун могуће количине виртуелне радне меморије која се може доделити виртуелној машини. Сада се врши читање вредности из *MemLogVM* датотеке и прорачунава се вредност количине виртуелне радне меморије која ће се доделити предметној виртуелној машини. Целокупан поступак прорачуна се на овом месту не разматра будући да је дат у претходном поглављу.

Пошто је прорачуната вредност количине виртуелне радне меморије која ће се доделити виртуелној машини, поступак додељивања се реализује као:

vboxmanage modifyvm "naziv_virtuelne_masine" --memory kolicina_virtuelne_radne_memorije

уз прослеђивање путем *subprocess.run()* функције. Путем претходне функције спроводи се и последњи корак у коме се задаје команда покретања виртуелне машине хипервизору у следећем облику:

vboxmanage startvm "naziv_virtuelne_masine" .

О завршетку реализације целокупне процедуре корисник се обавештава путем одговарајуће статусне поруке.

Гашење виртуелне машине се, као и претходни поступка, може обавити на два начина, коришћењем команде *Искључи* из подменија *Управљање*, или коришћењем дугмета *Искључи ВМ*. Слично претходном случају и у случају гашења виртуелне машине постоји низ поступака које треба спровести пре коначног прослеђивања команде

хипервизору која ће искључити виртуелну машину, али, за разлику од претходног случаја, овде ћемо имати и одређене процедуре које ће уследити и након прослеђивања команде искључења хипервизору.

И у овом случају у првом кораку предвиђено је заустављање сервиса који прати тренутне вредности доступне радне меморије али овог пута се ради о имплементацији тог сервиса у оквиру виртуелне машине. Виртуелна машина реализована је на бази Linux система па решења за остале платформе није потребно узимати у разматрање. Претходно је већ речено да се руковање споменутим сервисом на виртуелној машини остварује употребом одговарајуће shell скрипте како би се избегло прослеђивање великог броја параметара са физичког рачунара ка виртуелној машини. Сходно томе облик команде која ће се реализовати кроз *subprocess.run()* функцију је:

vboxmanage guestcontrol “naziv_virtuelne_masine” --username student run --exe /VMAdaptive/./servis.sh --wait-stdout -- zaustavi .

Након заустављања сервиса на виртуелној машини, кроз поменуту функцију реализује се команда:

vboxmanage guestcontrol "naziv_virtuelne_masine" --username student run --exe /VMAdaptive/./memtransfer.sh --wait-stdout > MemLogVM

која врши трансфер садржаја *MemLog* датотеке са виртуелне машине у датотеку *MemLogVM* на физичком рачунару. На основу података из *MemLogVM* датотеке прорачунава се средња вредност и након извршеног прорачуна та вредност се уписује у *MemLogVM* датотеку, уз брисање претходног садржаја датотекем, како би била доступна за наредни процес покретања виртуелне машине. Сада *subprocess.run()* функција реализује команду:

vboxmanage controlvm “naziv_virtuelne_masine” acpipowerbutton

на основу које хипервизор почиње поступак гашења виртуелне машине. Међутим самим гашењем виртуелне машине од стране хипервизора не окончава се целокупни поступак, јер се морају спровести одређене додатне радње како би се следећа сесија стартовања виртуелне машине могла спровести процедурално тачно и у потпуности. Потребно је угашени сервис за праћење тренутних вредности доступне радне меморије поново покренути како би се актуелне вредности бележиле у *MemLog* датотеку и како би могле бити искоришћене у процесу следећег покретања виртуелне машине. У противном, следеће покретање виртуелне машине не би било засновано на реалним подацима и могле би наступити непредвиђене околности у виду већег оптерећења физичких ресурса или чак немогућности покретања виртуелне машине. Како би се избегла потенцијална нежељеан дејства спроводи се поступак покретања сервиса сходно оперативном систему који се користи:

- *MSDCS start* уколико се ради о Windows систему,
- *sudo systemctl start MSDCS.service* уколико се ради о Linux систему и
- *launchctl start MSDCS* уколико се ради о macOS систему,

при чему се за Windows системе процесирање реализује коришћењем функције *ctypes.windll.shell32.ShellExecuteW()*, док се за Linux и macOS системе процесирање реализује коришћењем *subprocess.run()* функције из разлога који су претходно већ

наведени. Стартовањем сервиса окончан је целокупни поступак гашења виртуелне машине и корисник се о томе обавештава пригодном статусном поруком.

Рестартовање виртуелне машине се, као и у претходно описаним случајевима стартовања и гашења виртуелне машине, може обавити на два начина, коришћењем команде *Рестартуј* из подменија *Управљање*, или коришћењем дугмета *Рестартуј ВМ*. У оквиру хипервизора постоји предвиђена могућност рестартовања виртуелне машине и она би се могла реализовати коришћењем команде:

vboxmanage controlvm "naziv_virtuelne_masine" reset

али се на овом месту овакво решење неће користити. Ово се чини из разлога постојања оправдане сумње да ће се у већини случајева рестартовање вршити приликом наступања неких непредвиђених или отежавајућих околности приликом коришћења виртуелне машине. Како би се добила објективна слика понашања виртуелне машине и њене околине ваљало би и оваква стања укључити у процену, а претходно поменутом командом оваква стања би се изгубила. На пример можда се рестартовање чини из разлога отежаног рада услед преостале мале количине слободне виртуелне радне меморије. У том случају било би јако добро знати одређена стања на основу којих би се у следећој сесији доделила друга количина меморије и извршила адаптивност на новонастале услове. Из тог разлога се горе поменута команда неће користити за рестартовање виртуелне машине, већ ће се рестартовање реализовати обједињавањем команди које чине процедуру гашења виртуелне машине и команди које чине процедуру стартовања виртуелне машине у једном процедуралном току. Прво ће се реализовати све команде гашења, без приказивања статусне поруке, а по завршетку процедуре гашења, аутоматски ће бити започета процедура стартовања виртуелне машине, с тим што се на крају неће реализовати статусна порука стартовања, већ ће се приказати на крају целокупног процеса статусна порука везана за рестартовање. Оваквим приступом обезбеђује се правилан и сигуран рад виртуелне машине, као и њена адекватна адаптивност.

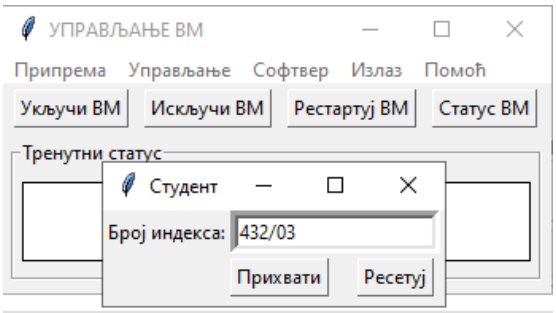
Тренутни статус виртуелне машине може се проверити позивањем команде *Статус* из подменија *Управљање*, или коришћењем дугмета *Статус ВМ*. Команда облика:

vboxmanage showvminfo "naziv_virtuelne_masine"

реализује се коришћењем *subprocess.run()* функције, уз дефинисање излаза параметрима *stdout=subprocess.PIPE* и *text=True*. Коришћењем функције *stdout.splitlines()* над добијеним излазом добија се скуп различитих информација о разним параметрима виртуелне машине. Информације су дате у облику листе коју чини преко 120 елемената. Коришћењем одговарајућих идентификованих индекса, приступ се ограничава на пет елемената дате листе релевантних за приказ кориснику а то су: количина виртуелне радне меморије додељене виртуелној машини, ознака виртуелног чипсета, број виртуелних процесора, тренутно стање виртуелне машине и резолуција графичког приказа. Виртуелна радна меморија која се приказује у мегабајтима, ознака чипсета, број виртуелних процесора и резолуција графичког приказа реализују се у оквиру искачућег прозора. На основу идентификованог стања виртуелне машине у оквиру статусне поруке приказује се да ли је виртуелна машина покренута или је тренутно угашена. Ова опција је додата како би корисник могао имати увид у тренутно додељене параметре виртуелној машини будући да су механизми адаптације виртуелне машине скривени и корисник нема увид у њих. Ови параметри се могу прочитати и у оквиру тренутних подешавања виртуелне машине у оквиру хипервизора, али се контакт корисника са хипервизором жели избећи како не би дошло до случајног нарушавања конзистентности виртуелне

машине од стране корисника. Такође, сматра се да се оваквим увидом у ограничени скуп тренутних параметара виртуелне машине постиже нижи ниво комплексности што се уклапа у тенденцију да фокус корисника буде усмерен ка остваривању позитивних резултата везаних за исходе учења.

Унос броја индекса, који представља јединy улазну тачку у софтверу где ће се затражити идентификација студента, врши се позивањем команде *Унос броја индекса* из подменија *Софтвер*. То је једино место где се мора затражити конкретна интеракција са студентом, сви остали елементи софтвера су базирани на аутоматизованим процедурама и интеракција са студентом се своди само на дефинисање тренутка у коме ће неко својство софтвера бити покренуто. Реализација уноса извршава се у посебном прозору где је унос омогућен кроз одговарајуће *Entry* поље које прихвата дати унос, као што је приказано на слици 73. Притиском дугмета *Прихвати*, функцијом *get()* вредност уноса се смешта у посебну глобалну променљиву *indeks* како би могла постати доступна за потребе процеса који ће користити унети број индекса у даљем раду. Прозор за унос броја индекса ће се тада аутоматски затворити а у оквиру статусне поруке биће приказана порука са унетим бројем индекса како би се остварио увид у евентуални погрешни унос броја индекса и оставила могућност да се у том случају поново покрене процедура уноса.



Слика 73. Идентификација студента

Број индекса једнозначно и недвосмислено одређује студента, па је у том смислу неопходан у процесу инсталације софтвера. На основу њега се егзактно утврђују потребни параметри за прослеђивање онтологији на основу којих ће се реализацијом упита над онтологијом извршити конструкција свих релеватних података неопходних за даље поступање приликом инсталације софтвера.

Инсталација софтвера потребног за остваривање задатака у оквиру наставних процеса покреће се позивањем команде *Инсталирање софтвера* из подменија *Софтвер*. У оквиру фолдера */VMAdaptive* задате виртуелне машине смештен је програм за инсталацију софтвера, на бази развијене онтологије, под именом *ontoinstall*. Овај програм коришћењем функције облика *sys.argv[1]* модула *sys* прихвата као улазни аргумент при покретању број индекса из променљиве *indeks*. Сходно томе прослеђивањем команде

vboxmanage guestcontrol "naziv_virtuelne_masine" --username student run --exe /VMAdaptive/./ontoinstall --wait-stdout broj_indeksa

функцији *subprocess.run()*, где је *broj_indeksa* садржај преузет из променљиве *indeks*, покренуће се инсталација свог неопходног софтвера у оквиру предметне виртуелне машине за потребе реализације наставних активности студента. Овакав позив програма за инсталацију уз прослеђивање вредности променљиве неопходан је како би се на основу броја индекса програм *ontoinstall* са виртуелне машине повезао са информационим системом високошколске институције и прикупио релевантне податке везано за студента (врста студија, студијски програм, година студија, предмети који се

слушају) и на основу њих направио улазне податке у виду листе. На основу тих улазних података, програм ће формирати адекватан упит над онтологијом, синтетисати неопходне информације потребне за инсталацију и спровести инсталациони процес за све захтеване софтвере неопходне за реализацију наставног процеса. По окончању целокупног поступка приказаће се одговарајућа статусна порука. Овде треба напоменути да, пошто се сав неопходан софтвер инсталира у оквиру виртуелне машине, пре почетка самог процеса биће извршена провера статуса виртуелне машине и ако се добије позитиван одговор на питање да ли је покренут и функционалан оперативни систем виртуелне машине, покренуће се целокупна поменута процедура. У противном приказаће се статусна порука да процедуру није могуће покренути. Провера стања оперативног система извршиће се покретањем shell скрипта *booted.sh* у оквиру виртуелне машине командом:

vboxmanage guestcontrol "naziv_virtuelne_masine" --username student run --exe /VMAdaptive/./booted.sh --wait-stdout

кроз *subprocess.run()* који даје одговор *Pokrenut* ако је оперативни систем стартован у целости, док у противном враћа празан стринг.

Излаз из програма омогућен је, поред стандардних могућности излаза дефинисаних самим опетативним системом у оквиру кога је програм покренут, коришћењем команде *Излаз* која се налази у главном менију. Излаз из програма реализован је позивом функције destroy() из tkinter модула и функције exit() из sys модула.

Помоћ се добија приступом команди Помоћ главног менија. Позивом ове команде отвара се искачући прозор који пружа кратко објашњење главних функционалности самог програма како би се корисник информисао које су му могућности коришћења на располагању.

5.3. Инсталационе процедуре

Неопходни софтвер који се користи приликом реализације решења и циљева описаних у оквиру докторске дисертације могу се поделити у две основне категорије: софтвер који ће бити инсталиран на рачунару корисника, односно на физичком рачунару и софтвер који ће бити инсталиран у оквиру предметне виртуелне машине. Даље се подела у свакој од ових категорија може извршити на интерни и екстерни софтвер. Интерни софтвер представља сав софтвер развијен у оквиру рада на докторској дисертацији и који је описан у оквиру докторске дисертације било у претходним поглављима, или поглављима која следе. Екстерни софтвер представља софтвер развијен од стране других произвођача, односно сав софтвер који није типски развијан за потребе и у оквиру предметне докторске дисертације. Оно што је заједничко за сав софтвер, без обзира којој категорији припадали, јесте да се мора извршити обезбеђивање правилних инсталационих процедура како би се обезбедило правилно функционисање сваког појединачног софтвера, као и софтвера као дела решења које се представља. У складу са тим развијене су одређене инсталационе процедуре које ће кратко бити размотрене у наставку.

Ако се разматра софтвер који ће бити инсталиран на рачунару корисника, односно на физичком рачунару реализација инсталационих процедура се мора извршити за сваки од заступљених оперативни система посебно. То значи да ће се посебно развити инсталационе процедуре за Windows, посебно за Linux и посебно за macOS системе како би се обухватила својства и посебни захтеви имплементације у сваком од ових система, будући да сваки оперативни систем има своје дозволе, своју организацију датотека, свој

начин имплементације сервиса и многе друге специфичности. Инсталационе процедуре за све оперативне системе развијене су у оквиру Python програмског језика, без графичког корисничког интерфејса како се не би повећавао ниво сложености самих процедура. У развоју инсталационих процедура нису коришћени наменски програми за развој инсталационих пакета који се могу наћи на тржишту будући да не могу одговорити у потпуности на специфичности које постоје у самим инсталационим поступцима а које се захтевају од стране развијеног решења.

За потребе инсталације сами инсталациони програми и датотеке које се инсталирају морају бити доступни одредишном рачунару, а то се постиже тако што се датотеке смештају на одређено место на Интернету коме ће се моћи приступити преко URL или IP адресе. Како би се датотекама могло приступити на једноставан и флексибилан начин потребно је да та интернет локација суштински буде дефинисана као web локација са које ће бити могуће преузимање. У ту сврху формиран је одговарајући web сервер који је у овом случају био заснован на Linux Fedora Server оперативном систему и nginx web серверу али се у конкретној реализацији може користити било каква реализација web сервера будући да је намена искључива проста доступност датотека и не захтевају се неки посебни додатни услови. У оквиру web сервера, односно одговарајуће web адресе формиране су три фасцикле (фолдера) означене као Windows, Linux и macOS у које се смештају све релевантне датотеке, за сваки оперативни систем посебно, потребне за спровођење инсталационог поступка. У зависности од оперативног система, програми за спровођење инсталационог поступка преузимају се са одговарајућих Интернет адреса на следећи начин:

- за Windows оперативне системе:
  http://<URL_adresa | IP_adresa>/Windows/wininstall.exe
- за Linux оперативне системе:
  http://<URL_adresa | IP_adresa>/Linux/lininstall
- за macOS оперативне системе:
  http://<URL_adresa | IP_adresa>/macOS/macinstall

Након преузимања одговарајуће датотеке, иста се стартује локално на корисничком рачунару и стартовањем почиње процедура инсталације. Сви инсталациони програми у себи обједињавају процедуре добијања администраторских права за поступке инсталације, али је препорука да уколико корисник поседује одговарајући ниво знања и вештина и сам инсталациони програм стартује са администраторским привилегијама (на пример у Windows системима коришћењем опције *Run as administrator,* у Linux и macOS системима коришћењем *sudo* команде).

Након стартовања одговарајућег инсталационог програма у зависности од оперативног система на коме се врши инсталација, преузимају се одговарајуће датотеке намењене том оперативном систему. Без обзира на оперативни систем у ком се врши инсталација преузимање датотека врши се употребом *urllib.request* модула кроз функцију *urlretrieve(src,dst)*. Параметар *src* у зависности од оперативног система на ком се врши инсталација има један од следећих облика:

- за Windows оперативне системе:
  http://<URL_adresa | IP_adresa>/Windows/ime_datoteke
- за Linux оперативне системе:
  http://<URL_adresa | IP_adresa>/Linux/ime_datoteke
- за macOS оперативне системе:
  http://<URL_adresa | IP_adresa>/macOS/ime_datoteke

док ће параметар *dst* имати један од следећих облика:

- за Windows оперативне системе:
  C:\\VMAdaptive\\ime_datoteke
- за Linux оперативне системе:
  /VMAdaptive/ime_datoteke
- за macOS оперативне системе:
  /opt/VMAdaptive/ime_datoteke

За сваку од датотека, пре почетка процеса преузимања, врши се позив функције *urlcleanup()* у циљу отклањања свих привремених датотека насталих приликом преузимања, уколико такве датотеке постоје. Поменута функција позива се и на крају процеса преузимања датотека. Списак датотека које се преузимају и путања на којој се смештају по преузимању, у зависности од оперативног система, дат је у табели 23.

Табела 23. Преглед датотека које се преузимају у поступку инсталације

	Windows	**Linux**	**macOS**
Путања	C:\VMAdaptive	/VMAdaptive	/opt/VMAdaptive
Датотеке	vhinit.exe MSDCS.exe Upravljac.exe	vhinit MSDCS MSDCS.service Upravljac	vhinit MSDCS MSDCS.plist Upravljac

По окончаном преузимању датотека, врши се инсталација неопходног сервиса у оквиру постојећег оперативног система уз коришћење администраторских привилегија на систему. Наредбе које се извршавају у зависности од оперативног система приликом инсталације су:

- за Windows системе:
  C:\VMAdaptive\MSDCS --startup auto install
- за Linux системе:

sudo chown root:root /VMAdaptive/MSDCS.service
sudo chmod 644 /VMAdaptive/MSDCS.service
sudo cp /VMAdaptive/MSDCS.service /etc/systemd/system/MSDCS.service
sudo systemctl daemon-reload
sudo systemctl enable MSDCS.service

- за macOS системе:

sudo chown root:wheel /opt/VMAdaptive/MSDCS.plist
sudo chmod 644 /opt/VMAdaptive/MSDCS.plist
sudo cp /opt/VMAdaptive/MSDCS.plist /Library/LaunchDaemons/MSDCS.plist
sudo launchctl load - w /Library/LaunchDaemons/MSDCS.plist

Овим кораком је завршена инсталација свих неопходних елемената који су реализовани у оквиру докторске дисертације.

Међутим, као што је већ напред напоменуто у реализацији целокупног решења учествују и софтверски производи који нису развијени у оквиру докторске дисертације и који представљају производе трећих лица. Такви софтверски производи имају своје

инсталационе датотеке и своје поступке инсталације над којим није дозвољено вршити промене па се морају користити у изворном облику. Такав случај представља и хипервизор VirtualBox који се користи као део описаног решења. Овде није могуће избећи интеракцију са корисником и сам корисник ће морати проћи све кораке саме инсталације и инсталирати производ без додатне помоћи. Међутим, како би се спречиле евентуалне грешке у преузимању хипервизора, у оквиру развијеног инсталационог програма у оквиру дисертације су реализоване команде за преузимање одговарајуће инсталације хипервизора и покретање исте, тако да корисник преузима интеракцију тек када започне процес инсталације. Ово је могуће извести захваљујући чињеници да производ VirtualBox има јако добро организован систем преузимања инсталација. Све инсталације VirtualBoxa доступне су на адреси download.virtualbox.org/virtualbox/ oznaka_verzije, а датотеке су јасно и недвосмислено означене и не може се направити грешка у избору инсталационе датотеке за одговарајући оперативни систем. Сходно томе, сам процес преузимања инсталационих датотека и покретање истих могуће је имплементирати у код инсталационих програма који је развијен овом приликом. Након завршетка инсталације хипервизора окончава се сегмент инсталације софтвера на физичком рачунару.

Корисници, односно студенти, не учествују у процесу инсталације софтвера на предметној виртуелној машини, односно не остварују учешће у инсталационим процедурама директним путем. Њихово учешће у инсталацији софтвера на виртуелној машини заснива се на индиректном учешћу. Студенти ће инсталирати софтвер за реализацију наставних задатака, али ће то чинити, као што је већ описано, стартовањем одговарајуће опције у развијеном софтверу, а потом ће виртуелна машина коришћењем својих дефинисаних механизама даље инсталирати софтвер аутоматским путем. Међутим да би то било остварив приликом припреме базне виртуелне машине у оквиру високообразовне институције морају се, такође, реализовати одређени инсталациони процеси.

У складу са претходним и за особље високообразовне институције које реализује базну виртуелну машину такође су припремљене одговарајуће инсталационе процедуре. Инсталациони програм који ће се користити за инсталацију на базној виртуелној машини постављен је такође у оквиру реализованог web сервера и преузима се са адресе:

http://<URL_adresa | IP_adresa>/VMLinux/VMinstall .

Виртуелна машина базирана је искључиво на једној Linux дистрибуцији, па ће преузимање датотека употребом *urllib.request* модула кроз функцију *urlretrieve(src,dst)* имати облик параметра *src* задат на следећи начин:

http://<URL_adresa | IP_adresa>/VMLinux/ime_datoteke

док ће параметар *dst* облик бити дат као:

/VMAdaptive/ime_datoteke .

Као и у претходним разматрањима и овде важи да ће се за сваку од датотека, пре почетка процеса преузимања, вршити позив функције *urlcleanup()* из напред већ описаних разлога. У виртуелној машини све датотеке се смештају искључиво на путањи */VMAdaptive*, а врши се преузимање следећих датотека:

- MSDCS

- MSDCS.service
- ontoinstall
- servis.sh
- booted.sh
- memtransfer.sh

Због приступа датотекама кроз одговарајуће команде на страни физичког рачунара како је већ претходно описивано, у току процеса инсталације се на овом месту реализују и команде за мењање дозвола приступа над */VMAdaptive* као и над свим датотекама које се налазе у оквиру наведене путање:

sudo chmod 777 /VMAdaptive
*sudo chmod 757 /VMAdaptive/**

Будући да се ради о виртуелној машини, као што се види из приложеног, овде не постоји инсталација датотека за управљање виртуелном машином, као ни хипервизора, који се користе на физичком рачунару. Међутим и овде се мора успоставити одговарајући сервис по принципима идентичним оним као код физичког рачунара. Будући да је, као што је већ напоменуто, виртуелна машина базирана искључиво на Linux систему, у складу са претходно наведеним код разматрања везано за физички рачунар, скуп команди за покретање сервиса биће дат у следећем облику:

sudo chown root:root /VMAdaptive/MSDCS.service
sudo chmod 644 /VMAdaptive/MSDCS.service
sudo cp /VMAdaptive/MSDCS.service /etc/systemd/system/MSDCS.service
sudo systemctl daemon-reload
sudo systemctl enable MSDCS.service

Након реализације овог скупа команди процес инсталирања софтвера на базној виртуелној машини је завршен. Даља инсталација софтвера које ће корисници користити извршава се након трансфера виртуелне машине у хипервизор физичког рачунара, поступака који следе након тога у циљу успостављања одређених подешавања, уноса броја индекса студента и покретања програма ontoinstall кроз одговарајућу команду програма за управљање виртуелном машином.

Након извршених инсталационих процедура, од стране свих укључених страна у сам процес, може се рећи да је систем спреман у потпуности да приступи реализацији задатака везаних за остваривање наставних процеса у високошколском образовању за које је планиран.

6. РЕЗУЛТАТИ ТЕСТИРАЊА

6.1. Уводна разматрања

У претходним поглављима приказани су различити аспекти реализованог решења. Међутим, потребно је омогућити доказивост представљеног концепта, односно спровести одговарајуће тестирање над целокупним решењем како би се сагледала реална употребљивост целокупног решења. За један систем, какав је представљен у овој докторској дисертацији, може се рећи да је верификован само ако је претходно тестиран [194]. Правило да добру архитектуру одликује лакоћа тестирања [195], генерално је применљиво на целокупан систем, односно добро осмишљен систем би требало да прати и одређена лакоћа тестирања.

Постоји више врста тестирања које се могу спроводити над неким системом попут оног чији је опис дат у претходним поглављима. Да ли ће се неко тестирање спровести или неће може зависити од низа околности, као што су постојање одређених услова како би се спровело тестирање, да ли је тестирање потребно у датом моменту спровести, да ли има смисла спроводити дато тестирање и многи други. Од тестова који овде неће бити спроведени наводимо тестирање прихватљивости [196] и тестирање сигурности [197].

За тестирање прихватљивости би неопходно било учешће конкретне високообразовне институције као партнера у тестирању. Морао би се формирати тим сачињен од одговарајућег особља високообразовне институције како би се креирале одговарајуће тестне процедуре и на основу њих потом креирали одговарајући тестови, што у тренутку писања докторске дисертације није било могуће реализовати из разних техничких, правних и економских разлога. Слична ситуација је и са тестирањем сигурности које се, такође, не може спровести на адекватан начин. За адекватно тестирање сигурности потребно би било створити услове идентичне условима који стварно владају у оквиру високошколске институције. То би практично захтевало упознавање са деловима или информационим системом високошколске институције у целости, увид у документацију, сервисе, начине прикупљања, обраде и чувања података. Ово је неопходно како би се тестирало да ли повезивање са информационим системом високошколске институције на било који начин може компромитовати целокупан систем кроз евентуални неовлашћени приступ, недозвољени увид у податке, или неовлашћену промену података. Поменуто, нажалост, није тренутно остварив услед строгих безбедоносних правила које се тичу приступа информационом систему високошколске институције. Честа је појава да информационе системе високошколских институција реализују партнерске компаније или неке друге институције што додатно уноси комплексност у овај вид тестирања будући да се морају испоштовати безбедносни аспекти прописаних и захтеваних од више учесника, па је и то један од разлога зашто у овом тренутку није било могуће спровести овакав вид тестирања. Међутим, како не би дошло до погрешне интерпретације тренутног стања у погледу ових тестирања, треба напоменути да уколико би се решење пуштало у продукцијски рад, односно уколико би се решење описано у докторској дисертацији користило у реалним процесима извођења наставног процеса у високошколској установи, спровођење поменутих тестова било би од врло велике важности за стабилан и поуздан рад целокупног система.

У складу са [197] за тестирања која су спроведена у оквиру докторске дисертације изабрани су следећи видови тестирања:

- функционално тестирање,
- тестирање оптерећења

- тестирање конфигурације и
- тестирање корисничког интерфејса.

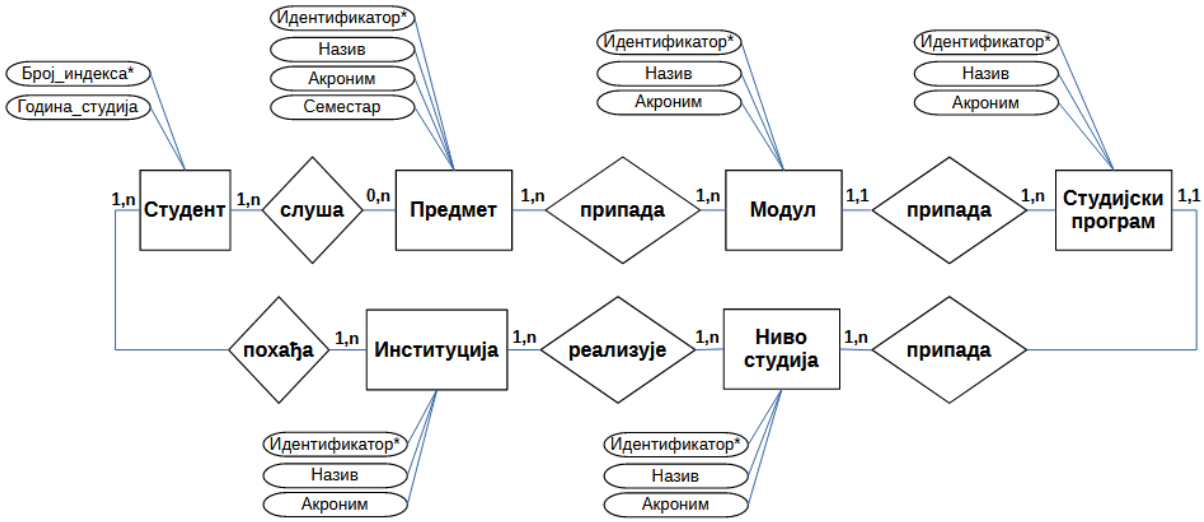
У оквиру функционалног тестирања провериће се да ли систем ради оно за шта је намењен, односно да ли се за одређене задате улазе добијају жељени излази. Овде важи и супротно, односно фукционално тестирање провериће да систем не реализује неке акције супротно његовој намени. У оквиру тестирања оптерећења, поред самог теста оптерећења система, извршиће се и тестирање перформанси. У оквиру тестирања перформанси провериће се време реакције система у нормалним условима, док ће се у оквиру тестирања напрезања проверити време реакције система у ванредним околностима. Тестирање конфигурације провериће да ли систем ради у различитим окружењима, односно на различитим хардверским конфигурацијама, различитим оперативним системима, различитим подешавањима конфигурације и сличном. При тестирању корисничког интерфејса провериће се да ли је кориснички интерфејс функционалан и да ли је једноставан за употребу.

Овде треба напоменути две кључне карактеристике тестирања које се често погрешно интерпретирају. Сам процес тестирања није усмерен ка проналажењу грешака у коду [198], већ процес треба пружити увид у остварење функционалности самог система, односно да ли систем при извршеном тестирању ради у очекиваним и жељеним нормативима, или не. Оно што је такође битно истаћи јесте да неуспешан тест не иницира промену теста, односно самих процедура тестирања [199]. Он указује на постојање неке грешке у систему коју треба анализирати и исправити, а потом након уочене и исправљене грешке, такав измењен систем треба поново подвргнути истоветном тестирању заснованом на идентичним тестним процедурама

6.2. Функционално тестирање

Како је већ напред наведено, тренутно не постоји могућност приступа самом информационом систему, његовим сервисима, као и документацији везаној за реализацију информационог система високошколске установе. Наведено решење би се применом у пракси морало ослонити на синергију са информационим системом високошколске установе како би се, на основу унетог броја индекса, могли прибавити релевантни подаци неопходни за рад система, а који се тичу самог студента: година коју студент похађа, ниво студија, студијски програм, модул. На основу претходног одређује се које предмете студент слуша, па се на основу тога одређује и потреба за инсталирањем одређеног софтвера на виртуелној машини које ће студент користити у реализацији задатака предвиђених наставним процесом. У недостатку могућности за приступ реалним сервисима информационог система високошколске установе, а како би се функционално тестирање система спровело у потпуности, функционалности информационог система високошколске установе су симулиране на адекватан начин. Реализована је адекватна релациона база података која симулира смештај горе наведених података у информационом систему високошколске установе. При томе узет је минималан скуп података потребан за тестирање функционалности система. Модел објекти-везе реализоване релационе базе података приказан је на слици 74. Сервис преко кога ће решење које се описује у докторској дисертацији за потребе функционалног тестирања приступати овим подацима заснива се на одговарајућем програму који, за одговарајући улаз дат у виду броја индекса студента, реализује одговарајућу серију SQL упита над базом, прибавља одговарајуће податке и смешта их у оквиру одговарајуће листе како би се даље користили у процесу од стране SPARQL упита над одређеном онтологијом. Овако симулиран приступ информационом систему високошколске

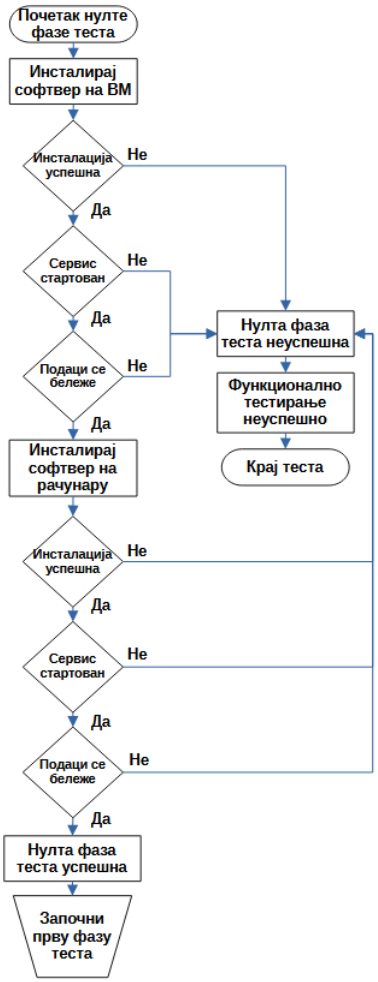
институције је адекватан за предметно функционално тестирање јер обезбеђује генерисање неопходних података дефинисаних као улазни подаци у оквиру система који се тестира. Будући да би се подаци у стварности генерисали од стране екстерног система, овакав приступ не ремети функционалне тестирање и не може довести до погрешних закључака у вези анализе функционалности самог система јер не мења ни на који начин интерну структуру посматраног система.



Слика 74. Модел објекти-везе симулираног екстерног решења

За спровођење функционалног тестирања у посматраном случају није битан размештај компоненти, будући да ће бити обављена посебна тестирања која укључују и овај аспект посматрања система. Зато је за спровођење овог тестирања коришћен један сервер, два рачунара и одговарајућа мрежна инфраструктура за обезбеђивање међусобне повезаности. На серверу су смештене све неопходне датотеке за преузимање током инсталационог процеса, укључујући и саму инсталациону датотеку. Са истог сервера се преузима и базна виртуелна машина. На истом серверу се налази и онтологија којој ће се приступати у процесу инсталације софтвера неопходног за реализацију наставног процеса. За потребе спровођења тестирања сервер ће, такође, опслуживати и малопре поменуту релациону базу података. Са друге стране, један рачунар ће током тестирања имати улогу рачунара корисника, односно студента, док ће други рачунар имати улогу рачунара овлашћеног лица високошколске институције. Други рачунар је потребан како би се спровело тестирање инсталационог процеса везаног за базну виртуелну машину о чему је било речи у претходном поглављу. Систем тестирања у овом случају конципиран је кроз неколико фаза тестирања. Да би систем успешно прошао функционално тестирање морају се успешно реализовати све његове фазе.

Прво је реализовано тзв. „нулто тестирање“ којим је извршено тестирање инсталационог процеса, како за виртуелну машину, тако и за физички рачунар. Поступак тестирања је обухватао преузимање инсталационе датотеке са сервера и њено покретање, проверу да ли су све датотеке успешно преузете и смештене на одговарајућу путању. Потом је проверено да ли је сервис за снимање одговарајућих меморијских стања покренут и да ли се успешно реализује снимање циљаних података. Процес тестирања је прво покренут на виртуелној машини, а потом је покренут и на физичком рачунару. Процедура тестирања је графички представљена на слици 75.



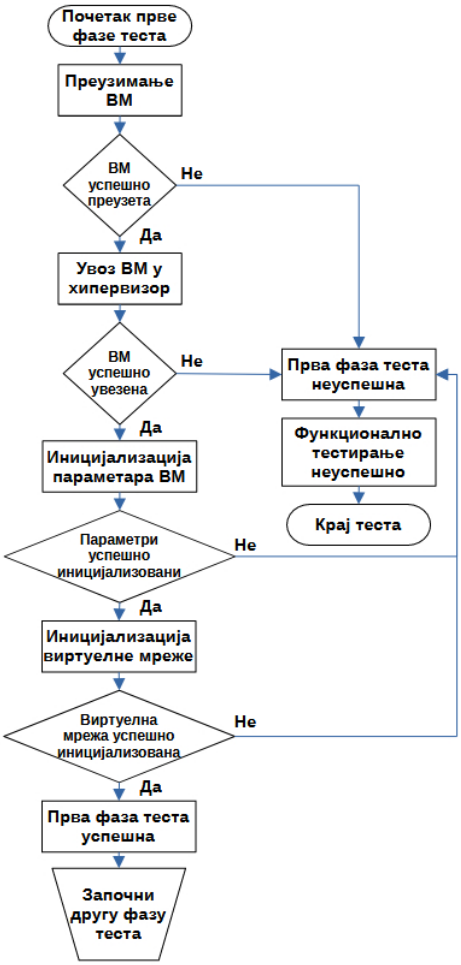
Слика 75. Процедура нулте фазе функционалног тестирања

Током процеса тестирања сви кораци у тестирању су оцењени као позитивни, што значи да је нулта фаза тестирања успешно окончана без појаве грешака и уочених проблема. Сходно захтевима тестирања, прелази се на следећу фазу тестирања.

У следећој фази тестирања проверава се процес преузимања виртуелне машине са сервера, њен увоз у хипервизор, примарна иницијализација параметара виртуелне машине у односу на окружење у коме ће она бити покренута, као и иницијализација виртуелне мреже у оквиру хипервизора у коме ће виртуелна машина остваривати свој рад. Процедура тестирања које се извршава у оквиру прве фазе илустрована је графичким приказом датим на слици 76. Овај корак тестирања се поклапа са провером исправности процеса који се реализују избором опција из подменија *Припрема* у оквиру корисничког графичког интерфејса саме апликације.

Током спровођења корака дефинисаних у тестној процедури, виртуелна машина је успешно преузета са сервера и извршен је њен увоз у хипервизор без икаквих уочених проблема. Примарна иницијализација је, такође, успешно извршена, параметри окружења су успешно препознати у складу са затеченим хардвером и вредности су исправно додељене одговарајућим параметрима виртуелне машине у складу са дефинисаним правилима доделе. Виртуелна мрежа у оквиру хипервизора је успешно стартована са одговарајућим параметрима.

У складу са претходним оцењено је да је прва фаза тестирања успешно окончана без уочених проблема и грешака и да се може прећи на следећу фазу у оквиру предметног функционалног тестирања.

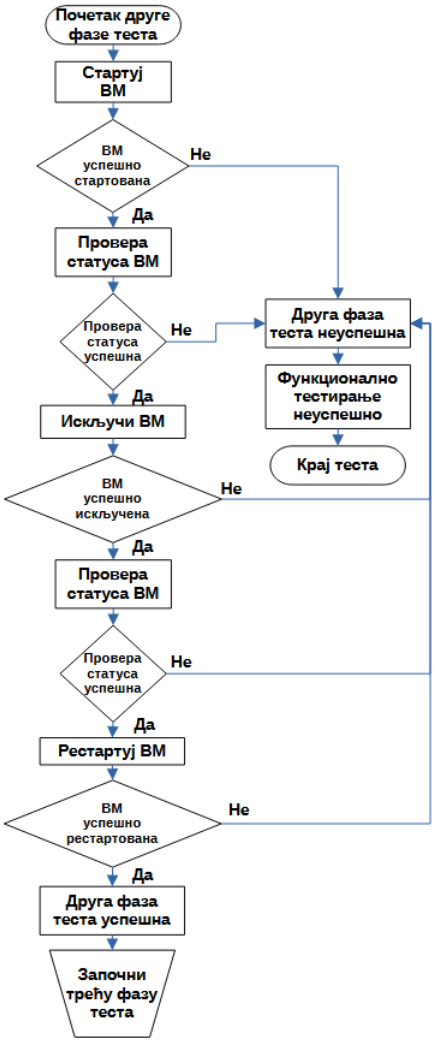


Слика 76. Процедура прве фазе функционалног тестирања

У другој фази функционалног тестирања оцењују се поступци стартовања, рестартовања, гашења, као и провере статуса виртуелне машине у реалном раду у оквиру задатог хипервизора. Овај корак подразумева проверу исправности процеса који се реализују коришћењем опција садржаних у подменију *Управљање*, као и коришћењем опција дугмади садржаних у корисничком графичком интерфејсу саме апликације. Процедура тестирања у овој фази дата је на слици 77.

Овде треба појаснити шта подразумева у овом случају стартовање, рестартовање и гашење виртуелне машине, будући да су ови поступци далеко комплекснији у счучају тестирања предметног решења од стартовања, рестартовања и гашења виртуелне машине из самог хипервизора. Процедура стартовања подразумева заустављање одговарајућег сервиса који прати тренутне вредности доступне радне меморије, прорачун меморијског буџета, додељивање количине виртуелне радне меморије виртуелној машини на основу прорачунатог меморијског буџета и на крају само стартовање виртуелне машине из хипервизора. Процедура гашења обухвата заустављање одговарајућег сервиса који прати тренутне вредности доступне радне меморије али у овом случају на виртуелној машини, трансфер забележених вредности доступне радне меморије, путем поменутог сервиса, са виртуелне машине на физички рачунар, гашење виртуелне машине из хипервизора и стартовање сервиса који прати тренутне вредности доступне радне меморије али у овом случају на физичком рачунару. Процедура рестартовања обухвата обједињавање процедуре гашења и процедуре стартовања виртуелне машине у оквиру једне процедуре

с тим што ће се прво реализовати сви аспекти процедуре гашења, а потом и сви аспекти процедуре стартовања виртуелне машине.



Слика 77. Процедура друге фазе функционалног тестирања

Приликом спровођења теста стартовања машине машина је успешно стартована у оквиру хипервизора. Међутим, то није довољан показатељ успешности, па је проверен и статус сервиса на физичком рачунару који је показао да је сервис привремено угашен. Остало је само да се провери усаглашеност додељене вредности радне меморије виртуелној машини са прорачунатим меморијским буџетом. Како би ово било могуће реализовати, за потребе теста направљен је посебан програм који ће комплетан меморијски буџет снимити у посебну датотеку што ће омогућити упоредивост са додељеним параметрима. У тесту виртуелној машини је додељена количина радне меморије 2.415 мегабајта, док је прорачунати меморијски буџет приказан у табели 24. Множењем крајње вредности из прорачунатог меморијског буџета са 1024, како би се добили мегабајти, добија се вредност 2.415,7184 изражена у мегабајтима што је у сагласности са вредношћу додељеној виртуелној машини. Мање одступање се јавља из разлога што се виртуелној машини може искључиво доделити целобројна вредност, па кад извршимо одбацивање децимала видимо да постоји слагање вредности. Тиме се може рећи да је процедура стартовања виртуелне машине успешно реализовања.

Табела 24. Прорачунати меморијски буџет виртуелне машине током реализације теста

Величина	Вредност	Опис
M_{ht}	7,7764 GB	укупна радна меморија рачунара
k_I	0,1670	коефицијент иницијализације
m_{MLpr}	5,4507 GB	прогнозирана тренутна вредност доступне радне меморије рачунара
m_{MLMpr}	4,9280 GB	прогнозирана средња вредност доступне радне меморије рачунара
m_{hap}	4,6893 GB	прогнозирана вредност доступне радне меморије рачунара
m_{gt}	4,0813 GB	количина виртуелне радне меморије која се може доделити виртуелној машини
M'_{gt}	3 GB	вредност додељене количине виртуелне радне меморије виртуелној машини током претходне сесије
m_{gamin}	2,3 GB	минимална вредност забележених стања доступне количине виртуелне радне меморије у претходној сесији виртуелне машине
M_{gt}	2,3591 GB	количина виртуелне радне меморије која ће се доделити виртуелној машини за наступајућу сесију

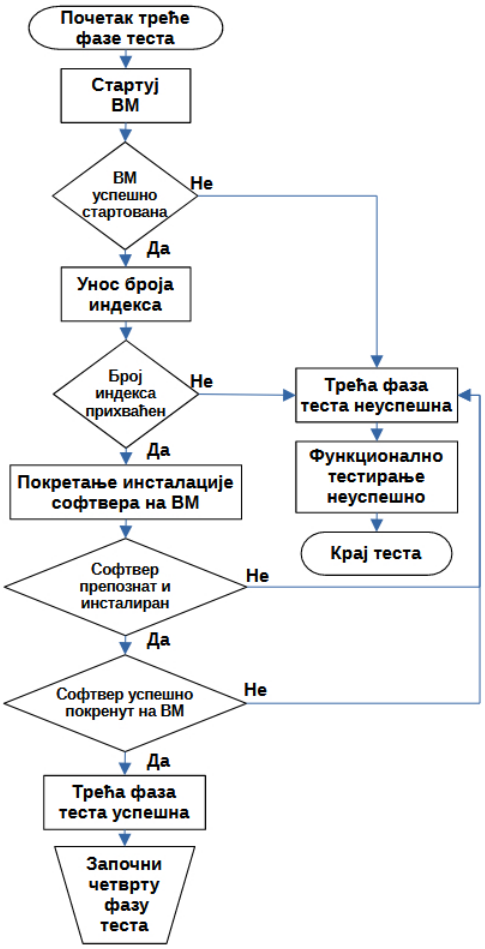
Сада се приступило гашењу виртуелне машине. Будући да је време реализације гашења виртуелне машине у појединим случајевима исувише кратко, како би се могла остварити провера да ли је дошло до привременог заустављања сервиса, приликом реализације теста уведено је одређено стање чекања које не постоји у нормалном извршењу програма, али је овде било неопходно. Стање чекања ни на један начин не ремети функционалности програма и користи се само у овом тесту, тако да нема ефеката на даља тестирања где би могло испољити одређене утицаје. Увидом у стање сервиса потврђено је да је дошло до заустављања рада истог. Провером садржаја датотеке *MemLogVM* на физичком рачунару и времена реализације записа, утврђено је да је трансфер података са виртуелне машине на физички рачунар реализован кроз одговарајућу процедуру гашења виртуелне машине. Увидом у стање виртуелне машине у самом хипервизору потврђено је да је статус виртуелне машине дефинисан као „угашена“ (powered off), чиме је потврђено да и заустављање рада виртуелне машине и њено гашење од стране хипервизора, такође, ради. Потом је проверено стање сервиса на физичком рачунару. Утврђено је да је сервис из заустављеног, прешао у радни статус, а такође је потврђено да се подаци снимају у одговорајућу датотеку, чиме је верификовано да све процедуре у оквиру гашења виртуелне машине испуњавају задатке у целости, па је и овај корак тестирања успешно окончан.

Будући да је показано да процедуре стартовања и гашења испуњавају своје функционалности у потпуности, тестирање рестартовања виртуелне машине се свело на потврђивање стања, односно да ли ће се виртуелна машина угасити и поново стартовати без икаквих додатних корективних радњи од стране корисника. И овај део теста је успешно реализован пошто нису примећени никакви проблеми приликом рестартовања машине.

Како би све функционалности биле у потпуности испитане, након провере стартовања и провере гашења виртуелне машине, спроведена је и провера да ли виртуелна машина када се затражи њен статус одговара позитивно или долази до грешака у очитавању тренутног статуса. Нарочито је било од интереса проверити да ли при промени статуса виртуелне машине долази и до промене у очитавању тог статуса. И овај сегмент тестирања завршен је са позитивним исходом.

Сходно претходно изнетим показатељима тестирања, друга фаза функционалног тестирања је успешно окончана, па се прешло на даље тестирање обухваћено процедурама треће фазе функционалног тестирања.

Трећа фаза тестирања обухвата проверу процедуре инсталирања неопходног софтвера на виртуелној машини за реализацију задатака у оквиру наставног процеса. Овај корак подразумева проверу исправности процеса који се реализују коришћењем опција садржаних у подменију *Софтвер* корисничког графичког интерфејса предметне апликације и заснива се на примени одговарајућих принципа онтолошког инжењерства који су већ разматрани у оквиру докторске дисертације. Сажета процедура тестирања у оквиру ове фазе приказана је графичким приказом датом на слици 78.



Слика 78. Процедура треће фазе функционалног тестирања

У овој фази тестирања користиће се симулирано решење информационог система високошколске установе како је већ претходно описано. Како не би дошло до ситуације да лоше конципирани симулирани део, намењен искључиво потребама тестирања, утиче на резултат теста и изазове да тест буде неуспешан, прво се приступило независном тестирању тог дела. Тестирање симулираног дела извршено је ван предвиђене фазе

тестирања реализацијом низа упита над подацима унетим у релациону базу података и снимањем резултата тих упита у одговарајуће елементе листе идентичним елементима какви ће се користити у целокупној фази тестирања. Добијени су очекивани резултати и жељено понашање симулираног дела, а при томе нису уочена никаква понашања ван предвиђених, тако да је оцењено да симулирани део неће компромитовати целокупну фазу тестирања и да се може користити у даљем процесу. Сходно томе приступило се реализацији тестирања кроз целокупну трећу фазу тестирања.

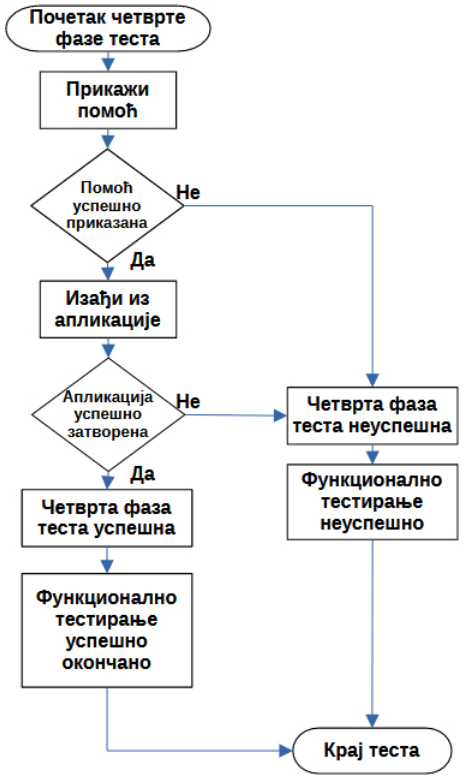
Након успешно стартоване виртуелне машине, унешен је број индекса путем процедуре која је објашњена у претходном поглављу докторске дисертације, а потом извршен увид у прихватање тог податка од стране одговарајућег софтвера на виртуелној машини (*ontoinstall*). Софтвер на виртуелној машини је прихватио податак непромењен по типу и по вредности, чиме је доказано да је податак пренешен у изворном облику са физичког рачунара у апликацију на виртуелној машини чиме је овај део теста успешно окончан. Потом је издата наредба да се изврши инсталирање софтвера. За потребе тестирања кроз апликацију је укључена опција терминалског приказивања (генерисања приказа кроз shell оперативног система) како би се могао испратити целокупан процес инсталирања софтвера. У регуларним условима коришћења апликације од стране корисника ова опција је искључена. Увидом у податке генерисане кроз поменути приказ, за унети број индекса и упоређивањем истих са подацима унетим у онтологију и у релациону базу података која симулира рад базе података информационог система високошколске установе, дошло се до сазнања да је апликација успешно и у целости пронашла неопходан софтвер за инсталирање, правилно дефинисала поступке инсталације и да је инсталација извршена исправно у целости за целокупан софтвер. Овде се желела још једна конфирмација успешности инсталирања софтвера, па је по окончању инсталационог поступка сав неопходан софтвер стартован у оквиру предметне виртуелне машине. Целокупан софтвер је стартован без појава грешака и у складу са очекиваним понашањем. Овим је трећа фаза функционалног тестирања успешно окончана, па се може прећи на наредну фазу у оквиру тестирања.

Пре него што се уђе у разматрање даљег тестирања, овде се мора дати једна напомена. Поменути терминалски приказ ипак је приказивао одређене грешке приликом реализације поступка инсталације. Те грешке су анализиране и установљено је да нису условљене директним понашањем апликације ни са једног аспекта њеног коришћења. Ради се о стандардним грешкама Linux система које се повремено могу сусретати током инсталације софтвера које систем генерише када информише корисника да одређеном серверу предвиђеном за преузимање датотека током инсталације (mirror) није могуће приступити и да се систем аутоматски пребацује на други предвиђени сервер за остварење задатог циља. Дакле, такве грешке су искључиво резултат саме конструкције оперативног система и начина инсталирања у оквиру истог, па се сходно томе такве грешке не могу узети као релевантне за само тестирање и стога су одбачене, а сама фаза тестирања проглашена успешном у складу са реалним стањем. Током стандардне употребе апликације корисник није свестан поменутих дешавања јер се све операције реализују као позадински процес где се пребацивање између сервера (mirrora) изводи аутоматски без потребе за икаквом интервенцијом корисника.

У завршној, четвртој фази тестирања извршено је тестирање опција *Излаз* и *Помоћ* које се налазе у главном менију апликације. Иако су, за корисника, ово можда мање битне опције апликације, одлучено је, како би се функционално тестирање обавило у целости да се обави краће тестирање и ових опција будући да су доступне у предметној апликацији. Приказ процедуре тестирања дат је шематски на слици 79.

Прво се приступило тестирању опције за пружање кратке помоћи кориснику о начину коришћења програма. Апликација је избацила одређени искачући прозор са кратким

информацијама релеватним за корисника апликације. Овим је констатовано да се помоћ приказује на предвиђени начин и овај део теста оцењеним је успешним. Потом је активирана опција изласка из апликације и прозор апликације је затворен. Међутим како би се утврдило да ли је заиста апликација затворена или је само затворен графички интерфејс апликације, извршена је претрага листе активних процеса на рачунару. У поменутој листи није био присутан процес који би означавао апликацију, па је сходно томе констатовано да је апликација у потпуности затворена и тиме је потврђена успешност овог дела фазе тестирања. Узимајући све у обзир. четврта фаза функционалног тестирања је оцењена успешном.



Слика 79. Процедура четврте фазе функционалног тестирања

Пре доношења коначног закључка о целокупном процесу тестирања овде треба напоменути још две чињенице у вези самог тестирања. Током реализације свих делова фаза тестирања адекватно су приказиване пратеће поруке саме апликације и није уочен ниједан проблем у начину, садржини и моменту приказивања истих. Корисник се уредно обавештава о току употребе апликације, конкретним стањима и предузетим радњама, тако да апликација и овде испољава предвиђено понашање. Приликом тестирања нису анализиране радње које се тичу самог прозора апликације (минимизација, промена величине и максимизација прозора, као и излазак коришћењем команде прозора) будући да се испољавање ових радњи могу благо разликовати међу оперативним системима, па би се стандардно понашање предвиђено оперативним системом могло протумачити као нестандардно понашање апликације. Команде прозора не утичу на функционалност која се остварује самом апликацијом, тако да су изузете од тестирања на овом месту, а увршене у тестирање које се бави само анализом графичког корисничког интерфејса.

Узимајући све у обзир наведено и резултате остварене кроз све фазе тестирања закључује се да је функционално тестирање изведено успешно у целости. Апликација реализује све функционалности на одговарајући и предвиђен начин, без уочених изузетака и нестандардног понашања ван дефинисаног.

6.3. Тестирање оптерећења

Перформансе једног система се могу сагледати са много аспеката и могу зависити од много фактора. По некада фактори за које се на почетку чини да немају великог утицаја на укупне перформансе система касније могу вршити значајан утицај на исте и значајно их деградирати што ствара низ проблема за систем у целини као и за корисника који користи дати систем. Немогуће је сагледати све факторе релевантне за оцењивање перформанси, а такође је немогуће сагледати и настанак свих евентуалних поремећаја који могу редуковати исте. Зато се из мноштва фактора, врши издвајање утицајних величина које су релевантне за остваривање жељеног увида у целокупне перформансе система. У случају решења описаног у докторској дисертацији издвојене су четири величине:

- време потребно за преузимање базне виртуелне машине,
- време потребно за увоз базне виртуелне машине у хипервизор,
- време потребно за стартовање виртуелне машине од стране хипервизора и
- време потребно за прорачун прогнозираних вредности.

Остале величине се неће разматрати будући да не утичу значајније на перформансе система. На овом месту даће се само краће разматрање у вези дела предвиђеног за инсталацију софтвера намењеног остваривању задатака у оквиру наставних процеса и зашто поменути део није ушао у разматрања будући да може значајније утицати на перформансе целокупног решења које се посматра. Овај сегмент је у многоме зависан од екстерних фактора, па би овакве перформансе најбоље било процењивати када се решење конкретизује у одређеном окружењу. У овом тренутку не познају се довољно перформансе информационог система високошколске установе, његових сервиса, као и осталих пратећих елемената са којим ће се остварити повезаност решења које се описује у докторској дисертацији. Зато су ова разматрања у овом моменту искључена из тестова оптерећења будући да би се морало о истим нагађати и не би било могуће добити конкретне податке. Наравно, оног момента када се дато решење стави у домен одређеног окружења треба спровести и ову врсту тестова.

Тестирање времена потребно за преузимање базне виртуелне машине вршено је са базном виртуелном машином величине 3,67 гигабајта. Тестирање је вршено на различитим симулираним брзинама преузимања, узимајући те брзине као најчешћи репрезент брзине комуникације коју корисници имају приликом приступа интернету од куће. Брзине су одређене као најчешће заступљене брзине преузимања код пружаоца Интернет услуга МТС у Републици Србији. При томе у разматрање су узета оба заступљена вида приступа Интернету, приступ заснован на коришћењу оптичке инфраструктуре (FTTH - Fiber To The Home) и асиметрична дигитална преплатничка линија (ADSL - Asymmetric Digital Subscriber Line) заснована на коришћењу класичних телефонских линија заснованих на бакарним парицама. Како би се обезбедила непроменљивост услова теста, базна виртуелна машина је за преузимање постављена на серверу који се налази у сертификованом дата центру у складу са ISO/IEC 27001 стандардом, који поседује редудантност интернет везе, напајања, хлађења и који обезбеђује доступност свих капацитета минимално од 99,99 % времена. Са друге стране, рачунар на коме је вршено преузимање и мерени резултати за приступ интернету користио је стабилну оптичку везу, као и гигабитну мрежу на локалном нивоу.

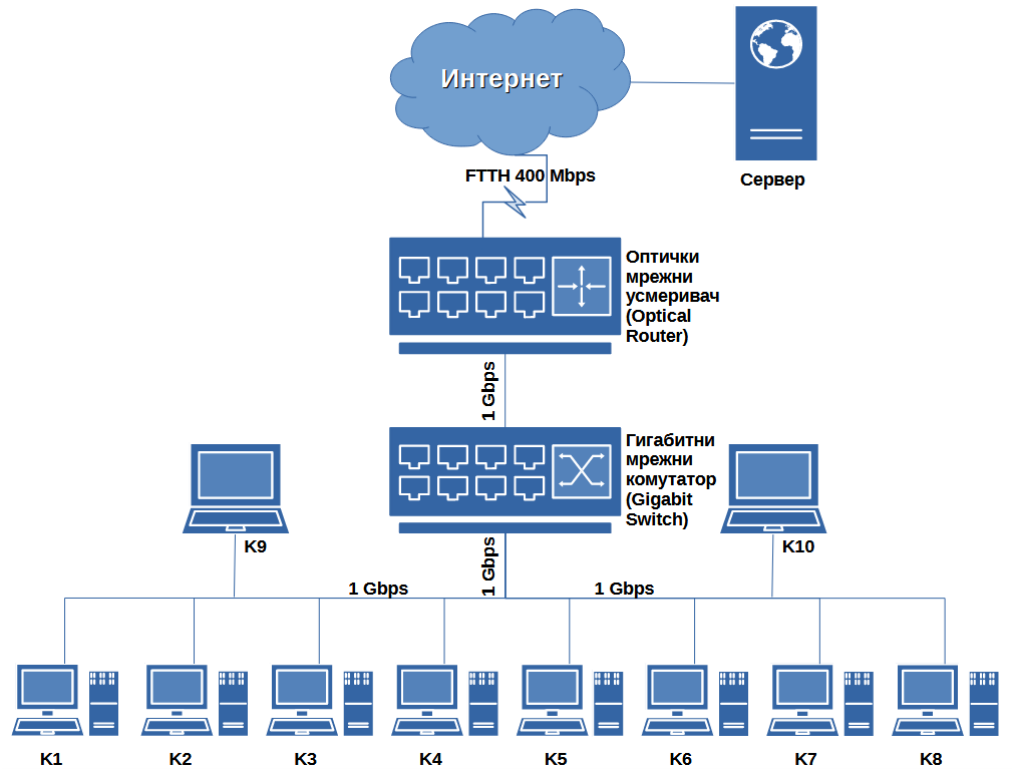
Постигнута времена преузимања за различите симулиране брзине преузимања дате су у табели 25.

Табела 25. Приказ постинутих времена преузимања базне виртуелне машине

Брзина преузимања	50 Mbps	100 Mbps	200 Mbps	400 Mbps
Време преузимања	12 min. 50 sec.	7 min. 3 sec.	4 min. 12 sec.	3 min. 3 sec.

Као што се из приложене табеле види, чак и на нижим брзинама, које представљају еквивалент брзина остваривих углавном коришћењем ADSL приступа Интернету, време преузимања је испод 15 минута што представља изузетно добар резултат. Најбоље време, свакако, остварено је при највећој брзини преузимања (еквивалент једне од могућих брзина остваривих коришћењем оптичког приступа Интернету) и оно износи око 3 минута. Међутим, ови резултати су добијени приликом приступа 1-1 где је један рачунар искључиво приступао преузимању, што представља идеалан случај који ће се у пракси врло ретко реализовати (није немогућ, али врло редак). Зато је од велике важности реализовати и другачију форму теста где ће се испитати случај преузимања када се преузимање врши од више корисника истовремено.

Тестирање преузимања базне виртуелне машине од више корисника истовремено реализовано је коришћењем принципа шематски приказаних на слици 80.



Слика 80. Шематски приказ начина тестирања времена преузимања базне виртуелне машине

Као што се са шематског приказа види тестно окружење обухвата десет рачунара (осам десктоп и два лаптоп рачунара) повезаних гигабитном мрежом коришћењем гигабитног свича. Гигабитни свич повезан је на гигабитни порт оптичког рутера који обезбеђује приступ Интернету коришћењем оптичке мреже брзине 400 Mbps у режиму преузимања и 200 Mbps у режиму слања чиме је обезбеђен довољни пропусни опсег за све рачунаре.

Сви рачунари преузимају истовремено базну виртуелну машину са исте локације, односно са истог сервера који се налази на Интернету и чија су својства идентична у односу на сервер коришћен у претходном појединачном тесту преузимања. Постигнута времена преузимања дата су у табели 26.

Табела 26. Приказ постинутих времена преузимања базне виртуелне машине на различитим рачунарима током истовременог преузимања

Ознака рачунара	К1	К2	К3	К4
Време преузимања	11 min. 43 sec.	11 min. 44 sec.	12 min. 10 sec.	11 min. 25 sec.
Ознака рачунара	К5	К6	К7	К8
Време преузимања	12 min. 37 sec.	12 min. 30 sec.	11 min. 33 sec.	14 min. 41 sec.
Ознака рачунара	К9	К10		
Време преузимања	18 min. 10 sec.	17 min. 44 sec.		

Као што се види из табеле, истовремена преузимања значајно су погоршала постигнута времена преузимања појединачних рачунара. Најбоље постигнуто време у овом случају износи 11 минута и 25 секунди, док је најспорије преузимање извршено за 18 мин и 10 секунди. Без обзира на значајније повећање времена преузимања, остварени резултати су и даље добри, будући да су сва преузимања завршена у временском оквиру мањем од 20 минута, а чак осам рачунара и даље постиже време преузимања испод 15 минута. Овим тестом се показује да се и у случајевима истовременог преузимања базне виртуелне машине од већег броја корисника и даље могу постићи добра времена преузимања. Будући да је први корак, у раду са системом приказаним у докторској дисертацији, преузимање базне виртуелне машине, у практичној имплементацији система то значи да ће се моћи остварити почетни услови рада система на адекватан начин без обзира на оптерећење које може наступити услед већег броја корисника.

Следећи тест који је реализован јесте анализа времена потребног да се изврши увоз базне виртуелне машине у предметни хипервизор. За овај тест одабрана су четири рачунара са различитим дисковима. Рачунари означени бројем 1 и 2 имају SSD (Solid State Drive), од којих рачунар 1 има SSD са брзином читања од 520 MB/s и брзином уписа података од 460 MB/s, док рачунар 2 има SSD карактеристика са брзином читања од 500 MB/s и брзином уписа података од 450 MB/s. Друга два рачунара имају HDD (Hard Disk Drive), од којих рачунар 3 има HDD карактеристике 7200 rpm, док рачунар 4 има HDD карактеристике 5400 rpm. Код свих рачунара повезаност дискова реализована је коришћењем SATA III стандарда. Остварена времена увоза виртуелне машине у предметни хипервизор приказана су у табели 27.

Табела 27. Приказ постинутих времена увоза базне виртуелне машине у хипервизор

Ознака рачунара	Рачунар 1	Рачунар 2	Рачунар 3	Рачунар 4
Време увоза	40 sec.	53 sec.	1 min. 33 sec.	2 min. 24 sec.

Резултати су очекивани и у складу са перформансама дискова који су коришћени за реализацију теста. Оно на шта је овај тест требао указати јесте очекивано време увоза базне виртуелне машине у хипервизор, а као што се из теста види то време ће бити и у случају слабијих перформанси диска свега неколико минута. Овде треба напоменути,

везано за овај тест, да су извршена и испитивања да ли сама попуњеност капацитета диска утиче на време потребно за реализацију поменутог увоза виртуелне машине. У реализованим испитивањима на идентичним рачунарима са идентичним дисковима извршене су пробе увоза виртуелне машине на 50, 75 и 90 % попуњености диска. У свим случајевима времена увоза базне виртуелне машине се нису значајније мењала, тако да се може рећи да ће увоз виртуелне машине у било ком случају, независно од конфигурације система, захтевати време краће од пет минута.

Примарна иницијализација и иницијализација мреже нису временски захтевне операције. Сходно томе није се приступило временској карактеризацији ових процеса будући да није било потребе за тим. У појединим случајевима остварује се време и краће од једне секунде па је јако тешко добити прецизне временске интервале.

У складу са свим претходним резултатима може се рећи да су очекивања да ће и у најекстремнијим ситуацијама за целокупне припремне операције бити потребно мање од 30 минута за завршетак истих, почев од момента када је започето преузимање базне виртуелне машине, преко увоза виртуелне машине у хипервизор и примарне иницијализације исте, закључно са завршетком иницијализације мреже у оквиру хипервизора.

За корисника решења које се разматра у оквиру предметне докторске дисертације једна од кључних временских карактеристика јесте она везана за време потребно да се изврши стартовање виртуелне машине од стране хипервизора. Ово тестирање обављено је над истим рачунарима над којим је спровођено и тестирање везано за време потребно да хипервизор изврши операцију увоза виртуелне машине. Када се хипервизору да команда за покретање виртуелне машине он спроводи мноштво припремних радњи везаних за иницијализацију виртуелног хардвера, па се тек након спроведене иницијализације од стране хипервизора врши покретање оперативног система виртуелне машине. Време иницијализације виртуелног хардвера је врло кратко и у већини случајева тешко мерљиво, тако да ће се овде разматрати време покретања оперативног система виртуелне машине као добра апроксимација времена стартовања виртуелне машине у целости, будући да су разлике у овим временима занемарљиво мале у већини случајева. Виртуелна машина која се користи у оквиру разматрања датих у докторској дисертацији заснована је на одговарајућој Linux дистрибуцији. Коришћена Linux дистрибуција у себи поседује механизме који могу јасно прецизирати протекло укупно време од издавања команде за покретање оперативног система до момента када је оперативни систем у потпуности спреман за даљу употребу. У овом случају тражено време добијено је коришћењем команде *systemd-analyze*, која приказује временску карактеристику покретања оперативног система. Добијени резултати приказани су у оквиру табеле 28.

Табела 28. Приказ времена потребног за покретање одговарајуће Linux дистрибуције инсталиране у оквиру виртуелне машине

Ознака рачунара	Рачунар 1	Рачунар 2	Рачунар 3	Рачунар 4
Време стартовања	14,302 s	13,167 s	24,660 s	37,733 s

На основу приложених података види се да је време далеко испод 60 секунди што се најчешће сматра неким граничним временом стартовања оперативног система на физичким системима. У случају виртуелних машина ово време је преполовљено, а у неким случајевима и 3-4 пута мање у односу на физичке рачунаре. На пример рачунар 2 користи идентичну Linux дистрибуцију која се користи и у оквиру виртуелне машине. Оперативни систем рачунара покреће се за 42,634 секунде, док се оперативни систем виртуелне машине покренуте на том рачунару покреће за 13,167 секунди што је скоро три пута краће време. Сходно добијеним резултатима, може се очекивати да ће се на

рачунарима корисника, виртуелне машине које се користе за реализацију решења представљеног у докторској дисертацији покретати са изузетном добрим временским карактеристикама и да ће и у најекстремнијим случајевима та времена бити прихватљива за кориснике.

Међутим, као што је већ претходно разматрано, у наведеном решењу се приликом стартовања виртуелне машине не врши само стартовање исте из хипервизора, већ је то један већи процес који обухвата више радњи које се спроводе пре и после самог стартовања виртуелне машине од стране хипервизора. Осим предиктивних радњи које се обављају у самом процесу, ниједна друга предвиђена радња није временски захтевна па се неће узимати у разматрање (стартовање и заустављање сервиса, разни други прорачуни у меморијском буџету и слично не представљају временски захтевне радње). Овде се поставило питање како извршити временску карактеризацију предиктивних радњи које се користе у оквиру прорачуна везаних за меморијски буџет будући да се не може унапред знати трајање сесија. Оне могу бити врло кратке, а у неким случајевима могу бити и неуобичајено дуге. На пример студент може користити рачунар свега неколико минута пре покретања виртуелне машине, а некада може рачунар користити и неколико часова па тек онда покренути виртуелну машину, што представља врло велике разлике у погледу количине података који ће се користити у самом предиктивном процесу. У складу са овим непознаницама тест је спроведен тако што је реализовано неколико сесија различитог временског трајања, а онда вршена предикција на основу података генерисаних током тих сесија. Како би подаци били упоредиви и смислени, коришћен је идентични рачунар током реализације свих сесија и спровођења тестова над њима. Тест се састојао у томе да се након одређеног временског интервала на основу података генерисаних у току тог временског интервала покрене предиктивни процес и да се измери време потребно за његову реализацију, од момента генерисања модела, до момента добијања одговарајућих прогнозираних вредности. Подаци о трајању сесија и временима потребним за генерисање прогнозираних вредности дати су у оквиру табеле 29.

Табела 29. Приказ времена потребних за генерисање предиктивних вредности

Време прикупљања података	1 h	3 h	6 h	9 h	12 h	24 h	7 дана	15 дана
Време генерисања прогнозираних вредности	<1 s	<1 s	<1 s	≈ 1 s	≈ 3 s	≈ 4 s	≈ 19 s	≈ 40 s

Као што се види из презентованих података, чак и у екстремним случајевима где је рачунар коришћен више дана без гашења или рестартовања истог и након тога покретања предиктивног процеса, времена потребна за генерисање прогнозираних вредности су остала у домену секунди (мања од једног минута). При томе треба имати на уму да се генерисање података током сесије врши на сваких 15 секунди, што за 15 дана представља датотеку са $15 \cdot 24 \cdot 60 \cdot 4 = 86400$ уписа коју током различитих предиктивних процеса треба процесирати, а при томе се прогнозирана вредност добије за 40 секунди. Наравно у пракси овако екстремни случајеви су врло ретко оствариви, у већини случајева сесија ће трајати неколико сати при чему ће се прогнозирана вредност добити за време до једне секунде. Овако добри резултати иду у прилог корисницима система, али нису зачуђујући или подложни сумњи. Остваривост овако добрих резултата лежи у чињеници да је модел коришћен у докторској дисертацији годинама усавршаван над великим скуповима

података и да је првобитна намена модела била да ради са врло великим временским серијама, па сходно томе подаци генерисани у оквиру представљеног решења за дати модел не представљају велики скуп улазних величина и неки посебан напор за обраду истих.

У складу са свим претходно представљеним тестирањима и на основу добијених резултата, може се рећи да решење успешно савладава тест оптерећења и да остаје у границама прихватљивих вредности за редовну употребу. Овде се даје и једна напомена. Сви тестови спроведени су у условима комплетне исправности хардвера и софтвера (оперативног система и његових софтверских компоненти) на којима је тестирање извршено и добијени резултати су очекивани у реализацији решења у окружењу које поседује одређену исправност. Сваки дефект у околини система може деградирати перформансе, међутим то је последица саме околине у којој систем остварује своје задатке и не може се ни на који начин третирати као недостатак самог система. Систем ће испољавати своје перформансе у најбољој могућој мери у складу са самим условима околине, уколико околина има адекватан степен исправности. могу се очекивати и добре временске карактеристике система, у противном систем може постати непредвидив по питању временских карактеристика.

6.4. Тестирање конфигурације

Решење развијено и приказано у оквиру предметне докторске дисертације намењено је раду у изразито хетерогеном окружењу. У складу са окружењем у коме ће се остваривати задаци, приликом тестирања конфигурације учињени су напори да се обезбеде такви услови тестирања који ће омогућити покривање једно ширег спектра могућих рачунарских конфигурација посмтрано са хардверског становишта. Будући да окосницу подешавања виртуелне машине чини подешавање параметара виртуелне процесорске јединице и количине додељене виртуелне радне меморије, фокус у избору тестних конфигурација био је управо на обезбеђивању разноликости ових параметара. Тако су обезбеђене рачунарске конфигурације базиране како на различитим генерацијама Intel процесора, тако и на различитим генерацијама AMD процесора. Поменуте рачунарске конфигурације поседују и разноликост у погледу укупне инсталиране меморије, па се количина радне меморије креће у распону од 8 гигабајта до 32 гигабајта. Будући да се виртуелни диск који се користи за рад виртуелне машине смешта у оквиру капацитета обезбеђених од стране саме рачунарске конфигурације и овде се водило рачуна да се обезбеди довољна разноликост како би се извршило једно шире тестирање конфигурације. Обезбеђени су дискови различитих перформанси како у писању, тако и у читању, а такође, обезбеђени су и дискови засновани на различитим технологијама израде (SSD и HDD) и различитих генерација. У претходним поглављима дисертације приказано је да је решење развијено коришћењем мултиплатформског принципа како би било свеобухватно и како би се обезбедила максимална функционалност истог. Како је решење применљиво на Windows, Linux и macOS платформама, приликом реализације теста, обезбеђено ја да тестне рачунарске конфигурације не буду засноване само на једној врсти оперативног система, већ да буду покривени сви циљани оперативни системи. Приказ карактеристика коришћених рачунарских конфигурација дат је у оквиру табеле 30.

Принцип тестирања огледао се у следећем. На свакој од рачунарских конфигурација инсталирано је предметно решење, а потом су пробане све функционалности у оквиру поменутог решења. У складу са захтевима решења, очекивано је да се на свим одабраним конфигурацијама може успешно извршити инсталација и да на свим одабраним

конфигурацијама, такође, раде све задате функционалности разматраног решења. У оквиру поменутог теста не врши се разматрање перформанси рада поменутог решења.

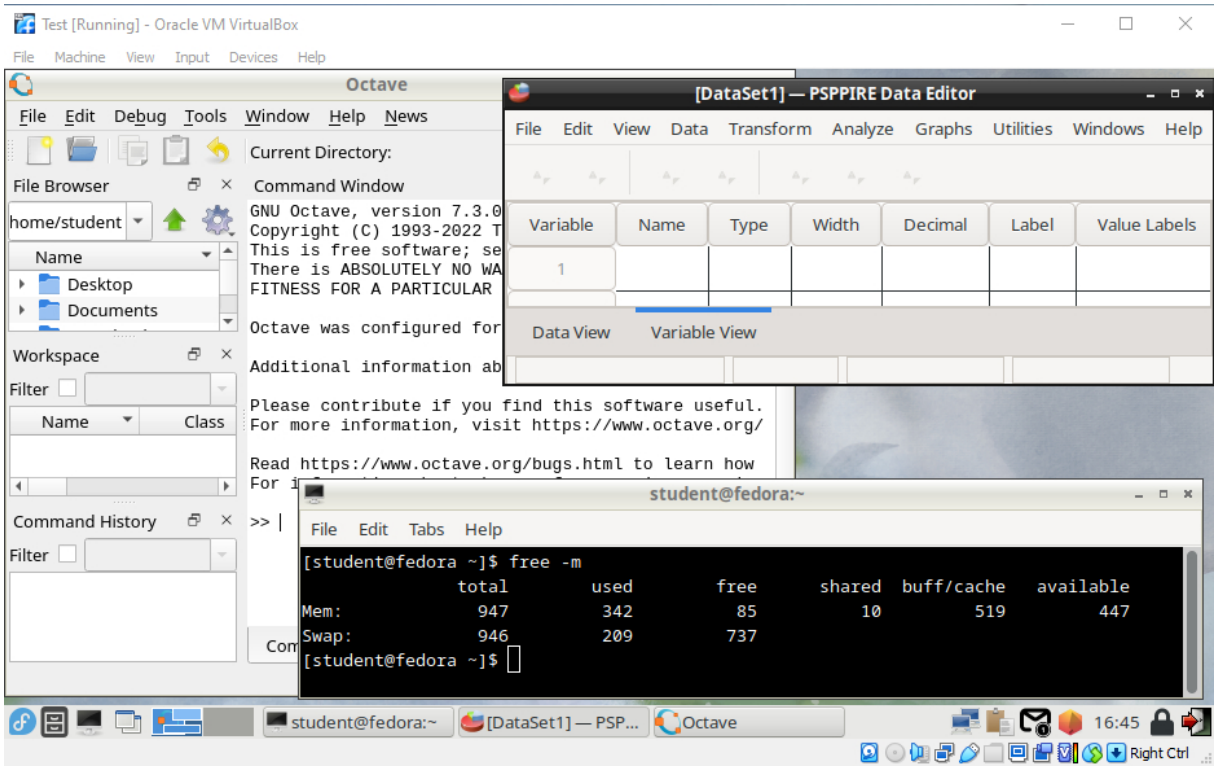
Табела 30. Основне карактеристике рачунара коришћених у тест процесу

Редни број	**1.**	**2.**	**3.**	**4.**
Централна процесорска јединица	AMD A8-7680	Intel i5-7400	AMD Pro A8-8600B	Intel i3-2350M
Количина радне меморије	10 GB	32 GB	16 GB	8 GB
Складиштење података	SATA III 256 GB SSD 520/470 Mb/s + SATA III 250 GB HDD 7200 rpm	SATA III 120 GB SSD 500/320 Mb/s + SATA III 1 TB HDD 7200 rpm	SATA III 128 GB SSD 520/460 Mb/s	SATA III 120 GB SSD 560/540 Mb/s + SATA III 500 GB HDD 7200 rpm
Мрежна повезаност	Gigabit LAN	Gigabit LAN	Gigabit LAN+WLAN	Gigabit LAN+WLAN
Оперативни систем	Windows 10 Enterprise 22H2	Fedora Workstation 37	Windows 10 Professional 22H2	Windows 10 Enterprise 22H2
Тип рачунара	Desktop	Desktop	Mini PC	Laptop
Редни број	**5.**	**6.**	**7.**	**8.**
Централна процесорска јединица	Intel i3-540	Intel Celeron G540	AMD A10-7400P	Intel i7-5775R
Количина радне меморије	8 GB	8 GB	16 GB	8 GB
Складиштење података	SATA III 120 GB SSD 535/410 Mb/s + SATA III 1TB HDD 7200 rpm	SATA III 500 GB HDD 5400 rpm	SATA III 480 GB SSD 500/450 Mb/s	SATA III 256 GB SSD
Мрежна повезаност	Gigabit LAN	Gigabit LAN	Gigabit LAN+WLAN	Gigabit LAN+WLAN
Оперативни систем	Fedora Workstation 36	Windows 10 Education 22H2	Fedora Workstation 37	macOS Catalina 10.15.7
Тип рачунара	Desktop	Desktop	Laptop	iMac All-in-one

Тест је успешно реализован будући да је на свим рачунарским конфигурацијама успешно и у целости извршен инсталациони процес, а потом успешно и у целости извршено и покретање свих функционалности у оквиру подручја рада решења које се разматра.

Овде се мора дати и неколико напомена у вези самог тестирања. За потребе теста је обезбеђен само један рачунар заснован на macOS систему јер није било могућности у моменту тестирања обезбедити више различитих рачунара тог типа услед њихове хардверске и софтверске специфичности. У складу са тим, мора се напоменути да је тестирање конфигурације везано за macOS системе извршено у прилично лимитираним условима. Друго, као што се види из претходне табеле минимална количина меморије која је била присутна код рачунарских конфигурација износи 8 гигабајта. Рачунарске конфигурације су коришћене у изворном стању и нису вршене никакве хардверске или софтверске промене над истим како би се обезбедила веродостојност тестирања конфигурације. За потребе тестирања у датом моменту није било могуће обезбедити изворну рачунарску конфигурацију са мањом количином радне меморије. Рачунарске конфигурације које су биле доступне у моменту тестирања са мањом количином радне меморије нажалост нису поседовале одговарајуће виртуелизационе могућности јер су биле засноване на старијим хардверским платформама.

Ван уобичајених протокола тестирања, одлучено је да се направи још један тест који не улази у свеобухватно тестирање, али је спроведен како би се утврдило да ли је могуће остварити циљане функционалности и у крајње екстремним условима. Из конфигурације означене редним бројем 1, уклоњен је један меморијски модул од 8 гигабајта, тако да је рачунарска конфигурација поседовала 2 гигабајта радне меморије. У таквим сложеним условима рада покренута је виртуелна машина која се адаптирала на нове услове користећи око једног гигабајта виртуелне радне меморије. У оквиру виртуелне машине било је могуће покренути апликације потребне за извођење наставног процеса без икаквих уочених проблема, како је приказано на слици 81, што доказује примењивост решења и у сложенијим условима рада.



Слика 81. Пример покретања апликација GNU PSPP и GNU Octave у оквиру виртуелне машине којој је додељено само 1 гигабајт радне меморије

Како би тест конфигурације био потпун, поред тестирања конфигурација са корисничког апсекта, потребно је дати и разматрање са серверског аспекта. У поступку тестирања конфигурације коришћени су сервери различитих спецификација приказаних у табели 31.

Табела 31. Основне карактеристике сервера коришћених у тест процесу

Редни број	**1.**	**2.**	**3.**	**4.**
Централна процесорска јединица	Intel Xeon E5410	Intel Xeon X3330	Intel Xeon E3120	Intel Xeon E3120
Количина радне меморије	8 GB	8 GB	4 GB	4 GB
Складиштење података	2 x SAS 73 GB SSD 15000 rpm + 2 x SAS 146 GB SSD 15000 rpm	SATA II 500 GB HDD 7200 rpm	SATA II 250 GB HDD 7200 rpm	SATA II 250 GB HDD 5400 rpm
Мрежна повезаност	Gigabit LAN	Gigabit LAN	Gigabit LAN	Gigabit LAN
Оперативни систем	Fedora Server Edition 36	Fedora Server Edition 36	Fedora Server Edition 36	Fedora Server Edition 36
Тип сервера	Rack	Desktop	Desktop	Desktop
Редни број	**5.**	**6.**	**7.**	
Централна процесорска јединица	Intel Xeon E3120	Intel Xeon	Intel i7-2600	
Количина радне меморије	4 GB	2 GB	16 GB	
Складиштење података	SATA II 500 GB HDD 5400 rpm	2 x Ultra320 SCSI 36,4 GB HDD 15000 rpm	2 x SATA III 3 TB HDD 7200 rpm	
Мрежна повезаност	Gigabit LAN	Gigabit LAN	Gigabit LAN	
Оперативни систем	Fedora Server Edition 36	Fedora Server Edition 36	CentOS 7.9.2009	
Тип сервера	Desktop	Rack	Rack	

У циљу што објективнијег приступа самом процесу тестирања, поред различитости серверских конфигурација одлучено је да се понуди и разноликост у погледу начина приступа Интернету. У том смислу, у складу са подацима приказаним у табели 32,

коришћен је један комерцијални пружалац услуга на територији Србије, затим пружалац Интернет услуга академским институцијама у Србији (AMPEC - Академска Мрежа РЕпублике Србије) и један пружалац Интернет услуга ван територије Републике Србије.

Табела 32. Интернет повезаност сервера коришћених у тест процесу

Редни број сервера из табеле 31	1, 2, 3 и 4.	5. и 6.	7.
Брзина преузимања	400 Mbps	100 Mbps	1 Gbps
Брзина слања	200 Mbps	100 Mbps	1 Gbps
Пружалац услуге	MTC, Београд, Република Србија	AMPEC, Београд, Република Србија	Hetzner Online Gmbh, Gunzenhausen, Germany

Приликом тестирања коришћени су следећи случајеви коришћења (СК):

- СК1: базна виртуелна машина се преузима са сервера са редним бројем 1, инсталационе датотеке се преузимају са сервера са редним бројем 2, онтологија којој се приступа налази се у оквиру сервера са редним бројем 3 и релациона база података за симулирање рада информационог система високошколске институције налази се у оквиру система за управљање релационим базама података на серверу са редним бројем 4,
- СК2: сви потребни елементи се налазе у оквиру сервера са редним бројем 5 осим базне виртуелне машине која се преузима са сервера са редним бројем 6 и
- СК3: сви потребни елементи налазе у оквиру једног сервера са редним бројем 7 смештеним у иностраном дата центру.

Насумичним избором рачунара из табеле 30, остварена су по три приступа (инсталациони процес и провера свих функционалности решења) сваком од дефинисаних случајева коришћења. У сваком приступу остварена је пуна функционалност, како самог инсталационог процеса, тако и коришћења апликације. Ово је било и очекивано будући да је решење реализовано тако да буде независно од начина серверске имплементације, али како би процес тестирања био потпун, било је потребно реализовати и овакав начин тестирања.

Независност решења од серверске имплементације значи да ће се рад поменутог решења описаног у докторској дисертацији остварити без обзира на различите начине реализације са серверске стране. Али то никако не значи да избор, број сервера, начин конфигурације и остали серверски параметри немају никаквих импликација на поменуто решење. Рецимо, поступци имплементације сервера, приказани у СК1 и СК2, допринеће бољим перформансама приликом реализације припремних радњи услед постојања сервера опремљених контролерима и дисковима високих перформанси, док на пример поступак приказан у СК3 нуди гарантовану доступност од 99,99 % времена услед редундантности и параметара који морају бити испуњени самим прописаним условима дата центра. Серверска инфраструктура није разматрана у оквиру докторске дисертације детаљније будући да обухвата стандардне поступке пројектовања серверских решења и систем администрације, али то не значи да је занемарљива приликом коришћења предметног решења.

У складу са свим изложеним може се констатовати да је предложено решење описано у докторској дисертацији успешно савладало тестирање конфигурације уз одређена ограничења на које је указано у опису тестног процеса.

6.5. Тестирање корисничког интерфејса

Тестирање корисничког интерфејса извршено је непосредним опажањем. У рачунарској лабораторији припремљено је тест окружење које се састојало од три рачунара, односно по једним рачунаром са Windows оперативним системом, Linux оперативним системом и macOS оперативним системом. На сваком рачунару стартована је апликација са одговарајућим корисничким интерфејсом. За потребе тестирања формиране су три групе од по три студента. Прво је одржан састанак у учионици где је свим студентима одржано кратко упознавање са апликацијом, њеним могућностима и циљевима коришћења, а потом је свака група уведена засебно у рачунарску лабораторију како би реализовала одређене задатке коришћењем апликације. У свакој групи за сваким рачунаром био је по један студент.

У првој групи студенти су добили задатак који треба да реализују у оквиру апликације коришћењем упутства за апликацију које им је било доступно у папирној форми. Друга група студената добила је задатак за реализацију у оквиру апликације али им је упутство саопштено усменим путем. Трећа група добила је задатак који треба реализовати коришћењем апликације при чему је за део реализације било доступно упутство у папирној форми, док је за остатак упутство саопштено усменим путем.

Након реализације добијених задатака код свих студената оцењивала се комплетност извршеног задатка од лица које је надгледало одвијање тестирања, а потом су сами студенти оцењивали интерфејс кроз три категорије: интуитивност, прегледност и једноставност. За сваку од категорија понуђена су само два одговора ЈЕСТЕ и НИЈЕ. Такође, студенти су могли у слободној форми да изнесу и додатна запажања у вези корисничког интерфејса.

Задатак је успешно комплетиран од стране свих студената, при чему су сви студенти изјавили да је могуће испратити реализацију на основу задатих упутстава без обзира на начин задавања. Овим је потврђено да се начин коришћења опција корисничког интерфејса може документовати на једноставан и разумљив начин. Интуитивност корисничког интерфејса позитивно је оценило 7 од 9 студената, прегледност су позитивно оценили сви студенти, док је једноставност позитивно оценило 8 од 9 студената. Додатно запажање је изнео само један студент.

Оба студента која су интуитивност оценила негативно, своју негативну оцену формирали су на основу тога што је целокупан кориснички интерфејс реализован на српском уместо на енглеском језику, па одређене појмове нису сматрали адекватним (посебно су истакли да би, по њима, било боље користити одредницу Download уместо одреднице Преузимање). Овакав коментар је усвојен као могуће побољшање у наредним верзијама апликације, где се може увести избор језика приликом коришћења корисничког интерфејса. Студент који је негативно оценио прегледност, образложио је своју оцену тиме да недостају неки класични елементи менија у виду скраћеница за приступ појединим опцијама (на пример истакао је недостатак позива помоћи притиском на тастер F1). И овај коментар усвојен је као оправдан за будуће верзије апликације где се кориснички интерфејс може додатно побољшати увођењем скраћеница за одређене опције садржане у менију. Додатно запажање једног од студената односило се на избор боја које су коришћене приликом реализације корисничког интерфејса. Студент сматра да је боље користити неку другу шему боја уместо стандардне шеме боја. Овај коментар је указао на непостојање подешавања изгледа корисничког интерфејса од стране самих

корисника, па је и ово запажање потребно размотрити приликом реализације неке од будућих верзија апликације, односно корисничког интерфејса.

Овим се може констатовати да је и кориснички интерфејс успешно прошао тестирање будући да постоји већи број позитивних од негативних оцена. У складу са тим кориснички интерфејс се може сматрати једноставним, употребљивим и сврсисходним, при чему на основу исказаних коментара простор за додатна побољшања постоји. Као што се види сваки од изнетих коментара није указао на одсуство неке од функционалности корисничког интерфејса, већ је исказан у форми могућих будућих унапређења која би могла бити реализована у циљу повећања задовољства корисника приликом коришћења апликације.

7. ПРИМЕНЉИВОСТ У НАСТАВНОМ ПРОЦЕСУ

Претходно поглавље показало је да конципирано решење пролази серију неопходних тестова којим се доказује његова применљивост посматрана са различитих аспеката. Функционално тестирање потврдило је да све имплементиране функционалности решења раде на предвиђени начин. Тестирање оптерећења показало је да решење поседује изузетно добре перформансе у разним режимима рада. Тестирање конфигурације је показало значајну карактеристику решења, а то је применљивост истог у изразито хетерогеном окружењу будући да је софтверски развој учињен за Windows, Linux и macOS фамилије оперативних система, уз постизање високог степена независности рада решења од хардверских карактеристика рачунарске конфигурације корисника, као и сервера који се користе у имплементацији. Тестирање корисничког интерфејса показало је висок степен применљивости од стране студената као битна спона која повезује самог корисника са функционалностима које се нуде у датом решењу. Тест прихватљивости изостао је, услед неопходности учешћа конкретне високообразовне институције у тестирању, што није могло бити омогућено у датом тренутку.

Међутим, иако се не може тврдити да је тест прихватљивости извршен, будући да заиста није, ипак се неке његове компоненте могу објаснити на основу стечених вишегодишњих искустава у реализацији наставних процеса у високошколском образовању. У складу са тиме ће се ова разматрања назвати применљивошћу у наставном процесу уместо коришћења термина прихватљивости како би се указало на јасно постојање разлике и како не би дошло до погрешних тумачења термина и њиховог међусобног изједначавања.

Сва разматрања базирана су на искуствима везаним за извођење практичне наставе из предмета Процесна мерна техника и Аутоматизација технолошких процеса на Техничком факултету у Бору [200-202], Универзитета у Београду, али су лако применљива и на друге предмете у оквиру других високошколских институција. Предмет Процесна мерна техника слуша се у оквиру основних академских студија на Техничком факултету у Бору, у осмом семестру (четврта година студија) студијског програма Рударско инжењерство, модула Припрема минералних сировина и Рециклажне технологије и одрживи развој [203]. Предмет Аутоматизација технолошких процеса слуша се у оквиру мастер академских студија на Техничком факултету у Бору, у првом семестру (једногодишње студије) студијског програма Рударско инжењерство, модула Експлоатација лежишта минералних сировина, Припрема минералних сировина и Рециклажне технологије и одрживи развој [204]. У оквиру практичне наставе из оба предмета изводе се лабораторијске вежбе уз примену рачунара, а касније студенти користећи стечено знање у оквиру својих предиспитних обавеза реализују два пројектна задатка из предмета Процесна мерна техника, односно један пројектни задатак из предмета Аутоматизација технолошких процеса. За извођење практичне наставе у оквиру предмета Процесна мерна техника користе се софтвери GNU PSPP и [205] GNU Octave [206], док се за извођење практичне наставе у оквиру предмета Аутоматизација технолошких процеса користи искључиво софтвер GNU Octave.

Током извођења наставног процеса на поменутим предметима коришћењем наведеног софтвера у периоду од 2019. до 2023. године уочено је испољавање неких заједничких карактеристика и ставова студената:

- студенти не поседују довољан ниво рачунарских знања без обзира на постојање основне рачунарске писмености,
- студенти поседују различите рачунарске конфигурације код куће,

- студенти често немају представу о карактеристикама рачунарске конфигурације које користе код куће,
- студенти често преузимају погрешне инсталационе датотеке,
- студенти често спроводе инсталациони процес на погрешан начин,
- студенти често врше погрешно конфигурисање софтвера,
- студенти желе да софтверско окружење које користе у реализацији наставног процеса код куће буде идентично окружењу у оквиру рачунарске лабораторије,
- студенти сматрају да је дужност и обавеза наставника и сарадника на предмету свака врста помоћи при инсталирању софтвера на личном рачунару студента,
- студенти сматрају да је дужност и обавеза високошколске институције обезбеђивање неопходног софтвера за инсталацију на личном рачунару студента.

До интезивирања горе наведених карактеристика и ставова нарочито долази у периоду када је, услед пандемијских услова изазваних појавом COVID-19 болести, током 2020. и 2021. године дошло до привремене обуставе наставних процеса у оквиру простора високошколских институција и целокупна наставна активност почела да се реализује коришћењем принципа учења код куће без доласка у просторе високошколске институције. У том периоду испољавају се и још неки додатни проблеми који додатно врше усложњавање наставних активности:

- лоша димензионисаност појединих система,
- проблеми са комуникационим везама услед додатне оптерећености,
- недоступност појединих сервиса услед оптерећености,
- додатна оптерећеност студената,
- додатна оптерећеност наставника.

У складу са претходно изложеним, види се да традиционални приступи коришћења софтвера у наставним процесима који се реализују у оквиру високошколског образовања, поред разних погодности, у сам образовни процес уносе и одређену додатну комплексност и одређене проблеме. Како би се задржао квалитет образовног процеса и радило на унапређењу истог, мора се извршити осмишљавање нових приступа у реализацији наставних активности у складу са све доминантнијим принципима дигиталне трансформације традиционалних наставних процеса на глобалном нивоу. Представљено решење у докторској дисертацији се уклапа у наведене норме мењањем приступа у начину коришћења софтвера било да се ради о реализацији наставног процеса који је у потпуности ослоњен на учење од куће, или о хибридном моделу који подразумева синергију учења од куће са традиционалним начином извођења наставе у оквиру високошколских институција. Наведено решење умањује комплексност и даје адекватну акцију у смислу минимизације горе наведених испољених карактеристика које могу представљати сметње приликом реализације наставе.

Студенти рударског инжењерства нису уско усмерени на рачунарске науке и добијају нека основна знања из рачунарства само на првој години основних академских студија. Међутим, та основна знања не обухватају знања потребна за инсталирање и конфигурисање софтвера и уколико их студент није стекао самосталним радом ван наставног процеса, неће их поседовати, а како је уочено, постоји велики број управо таквих студената. Наведено решење овај проблем минимизује тако што целокупан процес аутоматизује и ниво интеракције студента са самим процесом своди на један неопходан и прихватљив минимум активности. Студент више не мора да води рачуна о хардверу и сотфверу који поседује на свом личном рачунару будући да ће се решење аутоматски прилагодити сваком хардверу и инсталираном оперативном систему на

локалном рачунару. У том смислу постоје две врло значајне активности које се остварују коришћењем решења о коме се дискутује. Прво, решење превазилази проблем хетерогености рачунарског окружења, тиме што на локалном нивоу детектује окружење у коме ће вршити своје активности и прилагођава се на исто. Друго, решење увек тражи оптималну конфигурацију за извршење задатака и прилагођава се тренутном затеченом стању рачунарског хардвера у моменту покретања, захваљујући одговарајућим уграђеним статистичким и предиктивним решењима. Тиме се постиже да ће решење увек пружати свој максимум у раду без потребе да нити студент, нити предавач имају представу о коришћеном хардверском и софтверском окружењу.

На предметима Процесна мерна техника и Аутоматизација технолошких процеса често је долазило до преузимања погрешних верзија софтвера и спровођења погрешних поступака инсталационих процеса од стране студената на својим личним рачунарима. Ова појава дешавала се услед непажње студената, неразумевања концепата инсталације, погрешним одабиром инсталације са званичне странице произвођача намењеној преузимању софтверског производа. Забележене су и ситуације да у међувремену произвођач избаци новију верзију производа, па студенти преузму инсталацију која је намењена новијој генерацији хардвера и рачунара и онда имају потешкоћа приликом инсталације или су у немогућности да спроведу инсталацију у целости. Предложеним решењем и овакве ситуације се успешно превазилазе захваљујући уграђеним механизмима заснованим на онтолошком инжењерству. На основу броја индекса студента препознаје се које предмете студент слуша и који софтвер је потребан за савладавање наставних активности у склопу тог предмета, па се креирају одговарајуће инсталационе процедуре. Целокупан процес од преузимања инсталације до завршетка инсталационих процеса се обавља аутоматски без икакве захтеване интеракције са студентом. Оно што је значајно споменути јесте да се по потреби обављају и аутоматизовани прединсталациони и постинталациони процеси, односно по потреби ће се извршити подешавање окружења у прединсталационом процесу, као и конфигурисање или реконфигурисање софтвера у постинсталационом процесу.

Као што је претходно споменуто предмети Процесна мерна техника и Аутоматизација технолошких процеса део својих активности реализују коришћењем истог софтвера, GNU Octave. У до сада коришћеном, традиционалном, приступу руковању софтвером за потребе наставног процеса, а будући да се ради о различитим предметима који се уз то слушају и у различитим семестрима (један у пролећном, један у јесењем), предметни наставник је морао два пута у току године да реализује исту активност везану за упознавање са поступком инсталације и подешавања софтвера. Сада тих активности нема и оставља се простор да се време које је коришћено за реализацију тих активности додели неким другим, битнијим, активностима у реализацији исхода учења. Такође, онтолошки приниципи подразумевају поновну примену стечених знања, што у овом случају значи да ће се једном стечено знање о поступку инсталације и конфигурисања софтвера успешно примењивати у оба предмета, без стварања посебног знања за сваки од њих. Дакле, сам процес се убрзава, постаје оптимизован, лакши за имплементацију, праћење и чини да фокус и студената и наставника остане на остварењу позитивних исхода процеса учења, уместо да се фокус повремено измешта ка пратећим активностима.

Наведено решење није применљиво само на рачунарима који су у власништву студената. Решење је општег типа и везује се за применљивост на било којој рачунарској конфигурацији, а не за власништво, намену рачунара, као и његову локацију. Овакав приступ може се користити и од стране наставног особља једнако успешно као и у случајевима коришћења од стране студената, чиме ће се наставном особљу олакшати рад приликом њиховог учешћа у целокупном наставном процесу. Применљивост овог

решења код наставног особља може бити у ситуацијама где један предмет држи више наставника, па се жели конзистентност окружења које се користи приликом наставе, односно жели се да сваки од наставника има идентично окружење за рад. Такође, исто се може постићи уколико се жели идентичност окружења које користи наставник, сарадник и помоћно особље у настави, или се жели да се на пример поседује идентично окружење у канцеларији, лабораторији и код куће. Дакле принцип универзалности окружења које се користи применом разматраног решења важи увек без обзира ко и где дато решење примењује.

У складу са претходним, не постоје препреке за коришћење датог решења и у рачунарским лабораторијама које се налазе у оквиру високошколске институције. Можда ће се чинити да примена оваквог решења у рачунарским лабораторијама нема смисла и да се не остварују погодности у односу на традиционалну инсталацију софтвера у оквиру истих, али то није тако. Применом у рачунарској лабораторији могу се остварити вишеструке погодности. Ако посматрамо време потребно за инсталацију софтвера у рачунарским лабораторијама оно се драстично редукује. На пример Процесна мерна техника слуша се у једној рачунарској лабораторији са 13 рачунара, док се Аутоматизација технолошких процеса слуша у другој рачунарској лабораторији са 10 рачунара. То значи да ако посматрамо софтвер GNU Octave, особље високошколске институције ће извршити укупно 23 инсталација. Применом овог решења број инсталација које ће извршити особље се редукује на нулу, будући да ће се остварити без директног учешћа особља захваљујући уграђеним аутоматским механизмима. Такође, применом овог решења у рачунарским лабораторијама, на рачунарима студената и рачунарима наставног особља долазимо до потпуно примењеног принципа универзалности и јединствености окружења јер ће на сваком месту где је решење примењено бити идентично окружење, идентична верзија софтвера, идентична подешавања. Овим се елеминишу у потпуности потенцијални проблеми који могу настати услед различитости окружења а који су често присутни у виду различитости приказа или недостатка појединих опција, различито дефинисаних путања и сличног.

Анализирајући комуникацију путем електронске поште предметног асистента са студентима који слушају Процесну мерну технику утврдило се да је скоро половина порука било везано за неки од проблема са софтвером, док је код Аутоматизације технолошких процеса то износи негде око четвртине укупно размењене комуникације са студентима, што опет представља значајну количину. Овде треба нагласити и да наставно особље не мора нужно поседовати одређена знања из области рачунарства, па трагање за решењима може захтевати изузетне додатне напоре и временске ресурсе. Применом наведеног решења се врши значајно растерећење како наставног особља, тако и самих студената, будући да ће се умањити број потенцијалних проблема везаних за софтвер. Сходно томе студенти неће трошити значајно време у отклањању недоумица или конкретних проблема везаних за софтвер, док наставно особље неће бити оптерећено додатним софтверским питањима и тражењем решења за превазилажење истих.

Ако наведена разматрања подигнемо на највиши ниво високошколске институције и ту се остварује додатни низ погодности за саму институцију. Високошколска институција на једноставан начин може извршити обезбеђивање софтверских окружења и њихову дистрибуцију свим заинтересованим странама. Редукују се и трошкови високошколске институције, будући да се целокупно решење заснива на примени бесплатних софтверских решења. Остварује се и боља организованост наставних процеса, бољи увид у софтверске аспекте реализације наставних процеса, а самим тим примена решења може омогућити и један информисанији приступ у доношењу одлука везано за употребу софтвера у реализацији наставних процеса. Можда најбитније од свега јесте да се применом приказаног решења подиже квалитет целокупног наставног

процеса у оквиру високошколске установе, јача се њен углед и остварује се већи степен позитивних исхода везаних за остваривање задатих циљева учења. Ово су само неки од потенцијално остваривих предности увођења представљеног решења у наставне процесе високошколског образовања, а постоји могућност да ће даљим коришћењем настати и нове директне и индиректне предности.

8. УОЧЕНА ОГРАНИЧЕЊА И МОГУЋИ БУДУЋИ ПРАВЦИ РАЗВОЈА

8.1. Уочена ограничења

Решење приказано у докторској дисертацији прошло је успешно неколико фаза и врсти тестирања. Тиме је доказано да је решење стварно употребљиво при реалним сценаријима коришћења. Током спровођења тестова јавиле су се и неке непредвидиве ситуације које могу утицати на перформансе приликом примене, изазвати органичену примену решења, или онемогућити примену решења у целости. Ови поремећаји нису недостаци који су се јавили у самој реализацији решења, већ се ради о појединим околностима које се могу јавити у околини у којој поменуто решење делује.

Да би решење било могуће покренути, рачунарска конфигурација мора поседовати виртуелизационе могућности. Овде се могу испољити неколико отежавајућих околности. Ретко се може десити да рачунарска конфигурација уопште не подржава виртуелизационе могућности и да се решење, у складу са тиме, ни на који начин не може покренути на задатој конфигурацији. То може бити случај са неким врло старим и за данашње услове, превазиђеним хардвером. Чешћи случај је да виртуелизационе могућности нису укључене у оквиру задате конфигурације. BIOS (Basic Input/Ouput System) најчешће у подразумеваним вредностима ове опције држи искљученим, па их је потребно укључити. У зависности од BIOS верзије, произвођача, типа хардвера ове опције могу имати назив VT-x, AMD-V, SVM или Vanderpool, а могу бити присутне још неке додатне опције које могу имати утицаја на спровођење виртуелизационих процеса. Нажалост, крајњи корисник мора сам предузети потребне активности на укључењу ових опција, будући да не постоји адекватан начин да се управљање овим опцијама и контрола стања ових опција изврши у оквиру одговарајуће апликације представљеног решења. Потенцијални проблем се може јавити и у случају слабљења батерије која напаја CMOS (Complementary metal-oxide seimconductor) будући да CMOS представља малу количину меморије у којој су записана сва тренутна BIOS подешавања. Падом напона испод одговарајуће граничне вредности, сва тренутна подешавања се губе и BIOS тада ради са подразумеваним вредностима, а као што је већ напоменуто, у оквиру подразумеваних вредности сва подешавања везана за виртуелизацију су искључена, па покретање виртуелизационих капацитета у оквиру представљеног решења неће бити могуће.

Рад виртуелне машине заснива се на коришћењу виртуелног диска који је додељен виртуелној машини. Више пута је напоменуто да се смештање виртуелног диска остварује у оквиру смештајних капацитета самог физичког рачунара. У складу са тим може се закључити да ће физичко складиште имати директни утицај на рад виртуелног диска. Овде је идентификовано неколико кључних момената који могу имати изузетан утицај на целокупно решење. Ако разматрамо хардверски приступ овом разматрању, сваки проблем у виду неисправности, оштећења, пада перформанси физичког складишта потенцијално се може директно манифестовати на сам рад виртуелног диска, а самим тим и виртуелне машине која се користи у оквиру предвиђеног решења. При томе та манифестација може бити у благом паду перформанси рада виртуелне машине, у значајном деградирању перформанси, или у тренутној или трајној немогућности употребе виртуелне машине. Овакав проблем се не мора јавити само у виду директне последице неке хардверске неисправности, слични ефекти могу се манифестовати и у појединим софтверским случајевима с тим што код софтверског испољавања нису примећени знаци трајног онемогућавања виртуелне машине. У случају покретања неког софтверског процеса који је изузетно захтеван у погледу коришћења ресурса на ком се

налази виртуелни диск који користи виртуелна машина, може доћи до благе или потпуне деградације перформанси виртуелне машине, као и до привремене немогућности коришћења исте. У тестирањима спроведеним у оквиру предметне докторске дисертације, овакве појаве дешавале су се искључиво на рачунарским конфигурацијама заснованим на Windows оперативном систему и том приликом су идентификоване две категорије случаја. У првом случају потпуну тренутну немогућност рада са виртуелном машином изазвало је аутоматско преузимање и покретање инсталације једног већег ажурирања Windows оперативног система које је интезивно приступало диску рачунара и произвело комплетно заузеће ресурса намењених писању и читању са диска. Самим тим виртуелни диск није могао користити те ресурсе па је био онемогућен у остваривању својих активности, а такво понашање довело је до тренутне немогућности коришћења целе виртуелне машине. Овакво стање није било краткотрајне природе и окончано је завршетком предузетих активности Windows Update сервиса након рестартовања рачунара. Слична ситуација десила се и у другом случају, али је извор тренутне немогућности коришћења виртуелне машине била другачија врста софтвера. Проблем се јавио услед покретања обимне провере садржаја диска рачунара на присуство вируса од стране сервиса антивирусног софтвера. Након извршене провере и прекида реализованих активности антивирусног софтвера могло се наставити са редовним коришћењем виртуелне машине без потребе за њеним рестартовањем. Овакве и сличне проблеме могуће је превазићи редуковањем броја аутоматских процеса који приликом реализације својих операција подразумевају интезивно коришћење ресурса намењених читању и писању на диск. Ово редуковање се може остварити на два начина, превођењем аутоматских процеса у процесе за које је предвиђен мануелни начин покретања, или ограничавањем покретања аутоматских процеса на сатнице у којима је вероватноћа да ће се користити виртуелна машина изузетно ниска (углавном касни ноћни и рани јутарњи часови).

Различита безбедоносна подешавања такође могу утицати на онемогућавање функционисања делова предвиђеног решења или решења у целости и ово је једнако заступљења појава на свим платформама (Windows, Linux, macOS). Код Windows система уочени су проблеми када су подешавања UAC на врло високом нивоу сигурности, код Linux система уочена су блокирања одређених могућности од стране врло рестриктивних правила SELinux заштитног система, док је код macOS система уочено да извор потенцијалних проблема приликом рада са предметним решењем представља комбинација различитих безбедоносних правила (проблем покретања апликација које се не налазе на AppStore и нису развијене од стране сертификованих програмера, проблем приступа апликација датотекама и директоријумима и слично). Велика већина безбедоносних правила би могла бити измењена реализацијом одговарајуће апликације у оквиру представљеног решења, али то ипак није учињено из разлога приступа рачунарима који представљају приватно власништво студената и потенцијалних проблема, нарочито оних који спадају у домен правне регулативе, који могу проистећи променом неког од безбедоносних правила.

Уочено је да поједини антивируси у оквиру Windows оперативног система могу спречити покретање одређених делова предложеног решења услед лажне позитивне идентификације постојања малициозног кода у оквиру апликације. Детаљнијом анализом утврђено је да се ово дешава из неколико разлога. Примаран разлог је тај што апликација у моменту тестирања није била дигитално потписана, па самим тим није могло бити утврђено ко је произвео дату апликацију. Као секундарни разлози јављају се ти што сама апликација снима велики број хардверских параметара на самој физичкој машини, а потом врши одређене промене параметара на виртуелној машини. Када се све узме у обзир, антивирус такво понашање детектује као сумњиво понашање и дешава се

лажно позитивно идентификовање као потенцијалне претње. Овакво понашање се не испољава код свих антивируса, а дубљу анализу над великим бројем антивируса није било могуће спровести, будући да већина представља комерцијалне производе за које је потребна одређена лиценца. Зато ћемо само овде кратко размотрити случај лажне позитивне идентификације у оквиру бесплатног Avast антивируса. Avast нуди могућност пријављивања лажне позитивне идентификације и уклањање исте након извршених одређених анализа. Нажалост, њихове аутоматизоване процедуре нису дале адекватне резултате ни у једном од покушаја. Апликација је и даље пријављивана као потенцијална претња, односно вршена је лажна позитивна идентификација. Тек након директног контакта са представницима Avast компаније, личног представљања, објашњавања сврхе апликације и слања дела потенцијално сумњивог кода који је анализиран у оквиру њихових дигиталних лабораторија, апликација је трајно уклоњена са листе сумњивих апликација и више ни у једном случају није дошло до пријављивања потенцијалне опасности. Поступак уклањања је вероватно сличан код већине антивирусних компанија, али, као што је већ речено, није било могућности за веће и свеобухватније анализе услед недостатка лиценци.

Још један уочени проблем тиче се компајлирања кода. Како је већ напоменуто, за Windows 7 оперативни систем приликом реализације извршних датотека коришћена је старија верзија компајлера, односно последња верзија компајлера за коју је документовано да се може користити у оквиру Windows 7 система. Међутим уочени су и одређени проблеми када је извршено компалирање са најновијом верзијом Python компајлера за најновије верзије Windows оперативног система. Приликом компајлирања задњом верзијом компајлера више није постојала могућност стартовања одговарајућег сервиса у оквиру Windows оперативног система. Поновним компајлирањем кода нешто старијом верзијом компајлера ови проблеми су отклоњени и руковање сервисом је поново било могуће. Претпоставља се да овде долази до проблема да развој одређених библиотека не може увек адекватно и довољно брзо да испрати развој компајлера па долази у појединим случајевима до одређених неслагања и појаве грешака. Зато је у складу са овим сазнањима целокупно решење компајлирано са нешто старијом верзијом компајлера будући да постоји одређена зрелост како компајлера тако и целокупног екосистема који га окружује, па је вероватноћа појаве грешака услед неисправности и неусаглашености библиотека тиме сведена на прихватљив минимум.

8.2. Могући будући правци развоја

Постоје многи будући правци развоја приказаног решења у зависности од тога да ли посматрамо парцијално делове решења или посматрамо решење у целости. Уколико посматрамо парцијално сваки део, постоји потенцијално неограничен број могућих будућих праваца развоја у зависности где ће се одређени део интегрисати, односно у оквиру ког решења ће бити даље употребљаван.

Навешћемо само неке од могућности. Развијени сервис који документује одређена стања радне меморије могао би се искористити у оквиру готово сваког решења где је потребно испратити неку зависност од тренутног стања меморије. Предикција се може уз мање измене адаптирати и искористити за различита прогнозирања вредности које су представљене у оквиру временских серија. Одређени делови онтолошког решења се могу користити за рад са различитим сегментима високошколског образовања, док се други делови могу искористити приликом развоја разних решења намењених управљању софтвером. Овде треба додати битну чињеницу да се спектар могућих будућих употреба парцијалних решења утростручује будући да је свако решење развијено за Windows, Linux и macOS системе.

Међутим, овде ће ипак окосницу разматрања чинити могући будући правци развоја решења у целости, односно размотрићемо много извесније даље правце развоја у односу на малопре поменуте могућности над одређеним деловима решења. Окосница даљег развоја може се формулисати кроз три битна сегмента: унапређење предикције као основа постизања још бољег приступа у предвиђеном решењу, могућност даље интеграције решења у LMS (Learning Management System) и постизање једног глобалнијег приступа у коришћењу развијеног решења.

Када се говори о унапређењу предикције, у приказаном решењу предикција је остварена коришћењем принципа 1-1 (један према један). То значи да ће се за једну виртуелну машину све предиктивне радње базирати искључиво на историјским подацима прикупљеним унутар те виртуелне машине и историјским подацима прикупљеним са физичког рачунара у оквиру кога је покренута виртуелна машина. Као што је показано овакав начин прогнозирања вредности даје изузетно добре резултате. Међутим, не постоји развијена предикција која није подложна даљем усавршавању и остварењу још бољих прогнозираних резултата. У складу са тим тврђењем и предиктивне радње развијене у оквиру овог решења имају огроман простор за даље унапређивање и усавршавање. Оно о чему треба размишљати јесте проширење скупа вредности на основу кога ће се доносити одређена прогноза. Овде су на располагању два принципа: 1-n (један према више) и m-n (више према више). Принцип један према више користио би податке са предметне виртуелне машине али би приликом доношења одговарајућих прогноза узимао у обзир стечена сазнања са више физичких рачунара. Аналогно томе, принцип више према више, поред узимања у обзир стечених сазнања са више физичких рачунара, у прогнозу уноси и стечена сазнања са више виртуелних машина. Овако развијени приступи били би далеко сложенији и захтевнији али би могли да омогуће боље предиктивне радње у изразито хетерогеном рачунарском окружењу. Наравно, посебна пажња би се морала обратити на перформансе, јер се радње у оквиру решења морају извршавати у разумном времену, односно не сме бити процеса који врше велику конзумацију времена јер то удаљава корисника од коришћења самог решења, а смањује и ефикасност самог решења. Претходно изречено захтева и комплекснији развој и ангажовање већег тима на развоју, али свакако треба и оваквим покушајима дати шансу. Не треба напустити ни унапређење приступа по принципу 1-1 јер постоји још много модела за предикцију који се могу испробати и видети да ли се можда могу боље уклопити у предвиђено решење од постојећег модела.

Ако се разматра интеграција развијеног решења у неке друге сродне системе, превасходно треба размотрити какве су могућности интеграције решења у LMS и сродне системе који имају дугу традицију коришћења у високошколском образовању и високу применљивост на глобалном нивоу. Иако на први поглед делују као различита решења и различити приступи, ипак се може остварити одређена повезивост концепата разматраних у докторској дисертацији са оним који се користе у LMS. У оба случаја тежи се да учење буде усмерено на ученика и у оба случаја се, такође, тежи да се изврши одређена промена традиционалне улоге коју има наставник. Оба ова концепта су значајна за дигиталну трансформацију високошколског образовања. Веће повезивање ова два приступа може се остварити увођењем нових функционалности разматраног решења. Развијене функционалности више су биле усмерене ка бољем остваривању неких техничких услова везано за учење чиме се спречава појава да се огромни ресурси троше за радње које нису примарно везане за остварење наставног процеса, већ су радње које подржавају тај процес. У даљем развоју треба разматрати увођење функционалности које се баве добијањем повратних информација о начину коришћења решења, као и одговарајућих процена приликом коришћења у високошколском образовном систему. Тиме би се сагледали и неки аспекти који овог момента нису у потпуности сагледани у

овом решењу попут ангажовања студента и мотивације. Овако добијени подаци у комбинацији са подацима са LMS омогућавају стварање једне шире слике и реализацију бољих анализа, бољих сагледавања стилова учења и унапређење целокупног система на много вишем нивоу. Такође, треба увести и одговарајуће сервисе за једноставну размену података између самог LMS и виртуелне машине, односно размену података директним путем, уместо до сада традиционалног приступа кроз увоз података. Тиме би студенти на виртуелној машини користили несметано податке који су у већини случајева садржани у оквиру LMS високошколске установе (примери, вежбања, задаци и остало). Наравно, неопходно је комуникацију остварити двосмерно, односно треба увести и могућности да на једноставан начин садржаји виртуелне машине буду пренети и доступни кроз LMS (попут реализованих вежбања, пројектних задатака и сличних материјала). Оно што је значајно поменути јесте чињеница да се повезивањем LMS са разматраним решењем остварује обострана корист будући да се проширују функционалности LMS, али се истовремено врши проширивање и функционалности самог решења заснованог на употреби виртуелне машине.

Током свих разматрања датих у докторској дисертацији фокус је био на коришћењу решења у оквиру једне високошколске установе. Решење је тако конципирано да омогућава једно шире синтетисање знања и могућност примене на једном вишем нивоу. Дато решење би се могло адаптирати и на неке друге моделе примена који обухватају истовремено коришћење једне истанце над више образовних установа. На пример уз мање адаптације могла би се извршити и централизација решења на различитим нивоима, универзитетском, државном, регионалном. Овим се отварају неке нове могућности за високошколски образовни систем. На пример успостављањем оваквог централизованог решења на републичком нивоу дала би се шанса да студенти различитих институција добију једнаке шансе за учење јер би користили исте принципе. Сродни предмети на различитим високошкослким институцијама би могли да користе исте концепте и иста решења. Ако посматрамо финансијску карактеристику предложеног решења, овакав приступ би пружио велики потенцијал високообразовним институцијама које поседују скромније буџете. Постоји примењивост решења и на светском нивоу. Захваљујући онтолошким принципима уграђеним у поменуто решење отвара се могућност адекватне размене информација из области коришћења софтвера у наставним процесима различитих високообразовних институција без обзира на географску лоцираност, чиме се отвара простор за једну ширу сарадњу. На пример уколико се покаже успешност коришћења неког софтвера на неком предмету у оквиру предложеног решења на некој високошколској институцији и идентичне концепте жели применити нека друга високошколска институција, она то једноставно може учинити коришћењем знања садржаних у онтологији будући да онтологије у потпуности подржавају размену и поновну употребу стечених знања. Тиме се лакше креира простор за реализацију заједничких курикулума и сличних активности, посматрано у сегменту употребе рачунара и одговарајућег софтвера у наставном процесу.

Као што се види из претходних разматрања постоји много потенцијалних будућих праваца развоја датог решења. Међутим, мора се напоменути да ће се могући будући правци моћи сагледати тек након што дато решење буде примењено у стварном наставном процесу. У том моменту ће се реално испољити праве карактеристике предложеног решења и тек тада ће се моћи остварити прави увид и донети одлука на основу реалних параметара где постоји простор за даље усавршавање предложеног решења, где поједина разматрања треба додатно уобличити, а где поједина разматрања треба у потпуности одбацити.

9. ЗАКЉУЧАК

Напредак дигиталних технологија учинио је да у модерном систему високошколског образовања данас већина предмета подразумева употребу рачунара у реализацији наставних активности, а самим тим и одређеног софтвера, без обзира на ниво студија који се посматра. Поред тога дигиталном трансформацијом високошколског образовања традиционални облик реализације наставног процеса све више простора уступа методама заснованим на учењу код куће, било да се ради о хибридном моделу где учење код куће постаје надоградња традиционалног држања наставе у просторијама високошколске институције, или о настави заснованој примарно на самом учењу код куће кроз примену неког одговарајућег модела попут модела обрнуте учионице.

Оно на шта је указано у докторској дисертацији јесте да наведеном дигиталном трансформацијом не трансформише се само наставни процес по горе поменутим принципима, већ долази и до одређене трансформације рачунарског окружења у коме се наставни процес одвија. Рачунарске лабораторије високообразовних институција, у којима су реализоване наставне целине које захтевају употребу рачунара и одговарајућег софтвера исказују велики степен хомогености рачунарског окружења. Насупрот томе, реализацијом наставних процеса коришћењем принципа учења код куће, студенти у наставни процес уносе своје личне рачунарске ресурсе који међусобно исказују велику разноликост како у хардверском, тако и у софтверском погледу, што доводи до траснформације рачунарског окружења у коме се одвија наставни процес из хомогеног у изразито хетерогено окружење.

Хетерогеност рачунарског окружења у наставни процес заснованом на том окружењу уноси одређену комплексност коју треба превазићи на одговарајући начин. Начин превазилажења хетерогености треба извести тако да се минимизује учешће како студената, тако и наставног особља у целокупном процесу. Минимизација учешћа студента у поменутом процесу неопходна је из потребе задржавања фокуса студента на радњама везаним за остварење одговарајућих исхода учења, уместо измештања фокуса на неке споредне радње које су ту као подршка самом наставном процесу. Слично, минимизација учешћа наставног особља у поменутом процесу настаје из потребе задржавања примарних улога саветника, тренера и водича студента кроз неопходне радње везане за остварење поменутих исхода учења. Другачије речено, фокус и студената и наставног особља треба остати на самом наставном процесу, а не делом извршити његово измештање на пратеће активности наставног процеса.

У складу са претходним за савладавање настале хетерогености рачунарског окружења предложена је употреба концепата виртуелизације остварених кроз коришћење виртуелне машине. Употребом идентичне виртуелне машине код свих актера у наставном процесу реализује се одређена униформност окружења чиме се, коришћењем одређеног нивоа апстракције, врши превођење хетерогеног рачунарског окружења у хомогено. Међутим, виртуелизационе технике иако омогућавају тражену хомогеност окружења, поново пред студенте и наставно особље постављају задатке које учесници наставног процеса морају испунити, а нису директно везани за сам процес. Зато се мора развити такво виртуелно окружење које ће моћи своје задатке испуњавати у хардверском и софтверском смислу уз велику дозу аутономије коришћењем одговарајућих аутоматизационих механизама без директног учешћа студената и наставног особља, или свођењем њиховог учешћа на прихватљиви минимум. Другачије речено, неопходно је развити хардверску и софтверску самоадаптивност виртуелног окружења имплементираног одговрајућом виртуелном машином.

Хардверска самоадаптивност реализована је кроз неколико корака. У првом кораку машина се адаптира на рад на постојећем рачунару кроз примарну иницијализацију параметара виртуелног хардвера у односу на стварне хардверске компоненте рачунара коришћењем реализованих механизама адаптације. У другом кораку реализују се сервиси који прате тренутна стања стварне радне меморије и тренутна стања виртуелне радне меморије. На основу ових историјских података коришћењем одређених предиктивних модела генеришу се прогнозиране вредности будућих стања радне меморије, како самог стварног рачунара, тако и виртуелне машине покренуте у оквиру његовог хардвера. На основу прогнозираних вредности, коришћењем одговарајућег математичког прорачуна налази се нова вредност количине виртуелне радне меморије која се додељује виртуелној машини приликом покретања. Ови поступци се понављају приликом сваког новог стартовања виртуелне машине чиме се обезбеђује самоадаптивност виртуелне машине на тренутно стање реалног хардвера без учешћа крајњег корисника.

Софтверска самоадаптивност реализована је употребом елемената онтолошког инжењерства. Развијеном онтологијом повезана су неопходна знања из два различита домена, домена високошколског образовања и домена управљања софтвером. На основу знања садржаних у развијеној онтологији целокупан софтвер се сам инсталира у оквиру виртуелне машине без икакве интеракције са студентом или наставним особљем високошколске институције. Виртуелна машина кроз одговарајуће механизме у стању је да препозна који софтвер је потребно инсталирати и на који начин га треба инсталирати. Такође, уколико постоји потреба за тим, на основу уграђених механизама за рад са онтологијом, извршавају се и одређене прединсталационе и постинсталационе радње, односно врши се припрема окружења за сам процес инсталације и након извршене инсталације, врши се конфигурисање или реконфигурисање софтвера. Уколико постоји потреба за тим ови поступци се могу позивати више пута, али у већини случајева довољна је само једна реализација поступка. Овим процедурама обезбеђена је самодаптивност виртуелне машине и у софтверском погледу.

Као што се види употребом самоадаптирајуће виртуелне машине, учешће студената и наставног особља у целокупном процесу сведено је на минимум. То значи да фокус актера наставног процеса остаје у целости на наставном процесу у сваком тренутку. Студенти и наставно особље не морају да познају рачунарско окружење и принципе рада истог, односно уместо традиционалног приступа у ком се актери ангажују око различитих аспеката рачунарског окружења, рачунарско окружење се у целости ангажује за остварење њихових задатака у наставном процесу. Овим се подиже вероватноћа постизања позитивних исхода учења на значајно виши ниво у односу на досадашњу праксу.

Употреба самоадаптирајућих виртуелних машина не испољава превелику захтевност у погледу потребне рачунарске конфигурације за покретање истих. Не постоји ни посебна захтевност у погледу серверске инфраструктуре која прати реализацију решења разматраног у оквиру докторске дисертације. У складу са тим можемо рећи да овакво решење испољава минималну захтевност у инфраструктурном, економском и енергетском погледу. У прилог овоме иде и чињеница да је реализација решења у потпуности ослоњена на употребу бесплатног софтвера и софтвера отвореног кода чиме се минимизују потребна финансијска улагања приликом изградње целокупног екосистема које се захтева од разматраног решења. На одређеном нивоу апстракције може се рећи да ово решење спада у категорију „low cost" решења, односно решења која за своју имплементацију захтевају изразито минимална улагања.

На основу свега претходно наведеног, може се рећи да су посебне хипотезе формулисане у оквиру ове докторске дисертације у потпуности доказане. Самим тим

доказана је и општа хипотеза, односно, развијен је скуп одговарајућих метода које омогућавају реализацију самоадаптирајућих виртуелних машина, како са хардверског, тако и са софтверског аспекта, уз помоћ којих се проблем хетерогености рачунарског окружења адекватно савладава и тиме омогућава боље остварење предвиђених исхода учења у наставном процесу.

На крају треба напоменути да је предвиђено решење прошло и обимна тестирања чиме су доказане захтеване функционалности разматраног решења. Наравно, простора за додатна побољшања има, као и могућности за надоградњу и остваривање додатних функционалности предложеног решења чиме би се још додатно унапредили наставни процеси у домену високошколског образовања.

Наведено решење има и ширу примењивост. Поред примарне оријентисаности на високошколско образовање, увођењем одређених блажих модификација, решење се може адаптирати за употребу и у осталим нивоима образовања (основно, средњешколско). Такође, решење може бити примењиво уз одређене модификације и ван система образовања, на пример у индустрији где може бити коришћено за обуку радника, за потребе рада радника од куће и сличних активности.

Као што се види предложено решење остварује највиши потенцијал у високошколском образовању, али поседује велики потенцијал и за остале видове примене.

ЛИТЕРАТУРА

[1] Du, X., Tang, S., Lu, Z., Wet, J., Gai, K., Hung, P. C. K. A Novel Data Placement Strategy for Data-Sharing Scientific Workflows in Heterogeneous Edge-Cloud Computing Environments. In proceedings of 2020 IEEE International Conference on Web Services (ICWS), Beijing, China, 2020, pp. 498-507, DOI: 10.1109/ICWS49710.2020.00073.
[2] Verba, N., Chao, K. M., Lewandowski, J., Shah, N., James, A., Tian, F. Modeling industry 4.0 based fog computing environments for application analysis and deployment. Future Generation Computer Systems, 2019, 91, 48-60, DOI: 10.1016/j.future.2018.08.043.
[3] Javadzadeh, G., Rahmani, A.M. Fog Computing Applications in Smart Cities: A Systematic Survey. Wireless Netw, 2020, 26, 1433–1457. DOI: 10.1007/s11276-019-02208-y
[4] Kumar, V., Laghari, A. A., Karim, S., Shakir, M., Brohi, A. A. (2019). Comparison of fog computing & cloud computing. Int. J. Math. Sci. Comput, 2019, 1, 31-41. DOI: 10.5815/IJMSC.2019.01.03
[5] Ghobaei-Arani, M., Souri, A., Rahmanian, A.A. Resource Management Approaches in Fog Computing: a Comprehensive Review. J Grid Computing, 2020, 18, 1–42. DOI: 10.1007/s10723-019-09491-1
[6] Li, Q., Kumar, P., Alazab, M. IoT-assisted physical education training network virtualization and resource management using a deep reinforcement learning system. Complex Intell. Syst., 2022, 8, 1229–1242. DOI: 10.1007/s40747-021-00584-7
[7] Lattner, C., Amini, M., Bondhugula, U., Cohen, A., Davis, A., Pienaar, J., Riddle, R., Shpeisman, T., Vasilache, N., Zinenko, O. MLIR: A compiler infrastructure for the end of Moore's law. arXiv:2002.11054, 2020. DOI: 10.48550/arXiv.2002.11054
[8] Lattner, C., Amini, M., Bondhugula, U., Cohen, A., Davis, A., Pienaar, J., Riddle, R., Shpeisman, T., Vasilache, N., Zinenko, O. MLIR: Scaling Compiler Infrastructure for Domain Specific Computation. In proceedings of the 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), Seoul, Korea (South), 2021, pp. 2-14, DOI: 10.1109/CGO51591.2021.9370308.
[9] Bogdandy, B., Tamas, J., Toth, Z. Digital Transformation in Education during COVID-19: a Case Study. In proceedings of the 11th IEEE International Conference on Cognitive Infocommunications (CogInfoCom), Mariehamn, Finland, 2020, pp. 000173-000178. DOI: 10.1109/CogInfoCom50765.2020.9237840.
[10] Alashhab, Z. R., Anbar, M., Singh, M. M., Leau, Y. B., Al-Sai, Z. A., Alhayja'a, S. A. Impact of coronavirus pandemic crisis on technologies and cloud computing applications, Journal of Electronic Science and Technology, 2021, 19 (1), 100059. DOI: 10.1016/j.jnlest.2020.100059.
[11] Qasem, Y. A. M., Abdullah, R., Jusoh, Y. Y., Atan, R., Asadi, S. Cloud Computing Adoption in Higher Education Institutions: A Systematic Review. IEEE Access, 2019, 7, 63722-63744. DOI: 10.1109/ACCESS.2019.2916234.
[12] Luchini, G., Alegre-Requena, J. V., Funes-ArDOIz, I., Paton, R. S. GoodVibes: automated thermochemistry for heterogeneous computational chemistry. F1000Research 2020, 9, 291. DOI: 10.12688/f1000research.22758.1
[13] Rocha, A. R., Travassos, G. H., de Oliveira, K. M., Villela, K., Santos, G., de Menezes, C. S. Ontologies in Software Development Environments. In Engineering Ontologies & Ontologies for Engineering, Nemo, Helsinki, Finland, 2020, 23-35. ISBN 978-1393963035
[14] Hao, Y., Yu, X. Ontology-based Software Trustworthy Requirements and Behavior Modeling. In proceddings of the 2021 International Conference on Networking,

Communications and Information Technology (NetCIT), Manchester, United Kingdom, 2021, pp. 460-466, DOI: 10.1109/NetCIT54147.2021.00097.

[15] Eito-Brun, R., Gómez-Berbís, J. M., de Amescua Seco, A. Knowledge tools to organise software engineering Data: Development and validation of an ontology based on ECSS standard. Advances in Space Research, 2022, 70 (2), 485-495. DOI: 10.1016/j.asr.2022.04.052.

[16] Alsanad, A.A., Chikh, A., Mirza, A. A Domain Ontology for Software Requirements Change Management in Global Software Development Environment. IEEE Access, 2019, 7, 49352-49361. DOI: 10.1109/ACCESS.2019.2909839.

[17] Bhatia, M. P. S., Kumar, A., Beniwal, R., Malik, T. Ontology driven software development for automatic detection and updation of software requirement specifications. Journal of Discrete Mathematical Sciences and Cryptography, 2020, 23(1), 197-208. DOI: 10.1080/09720529.2020.1721884

[18] Rocha, R., Bion, D., Azevedo, R., Gomes, A., Cordeiro, D., Leandro, R., Silva, R., Freitas, F. A Syntactic and Semantic Assessment of a Global Software Engineering Domain Ontology. In Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services (iiWAS '20). Association for Computing Machinery, New York, NY, USA, 2020, pp. 253–262. DOI: 10.1145/3428757.3429143

[19] Mkhinini, M. M., Labbani-Narsis, O., Nicolle, C. Combining UML and ontology: An exploratory survey. Computer Science Review, 2020, 35, 100223. DOI: 10.1016/j.cosrev.2019.100223.

[20] Yang, L., Cormican, K., Yu, M. Ontology-based systems engineering: A state-of-the-art review. Computers in Industry, 2019, 111, 148-171. DOI: 10.1016/j.compind.2019.05.003.

[21] Tebes, G., Peppino, D., Becker, P., Matturro, G., Solari, M., Olsina, L. A Systematic Review on Software Testing Ontologies. In Quality of Information and Communications Technology. QUATIC 2019. Communications in Computer and Information Science, 2019, 1010. Springer, Cham. DOI: 10.1007/978-3-030-29238-6_11

[22] Triandini, E., Kristyanto, M., Rishika, R., & Rawung, F. A Systematic Literature Review of The Role of Ontology in Modeling Knowledge in Software Development Processes. IPTEK The Journal For Technology And Science, 2021, 32(3), 159-175. DOI:10.12962/j20882033.v32i3.12998

[23] Sikos, L. F. AI in digital forensics: Ontology engineering for cybercrime investigations. WIREs Forensic Sci., 2021, 3, e1394. DOI: 10.1002/wfs2.1394

[24] Sharma, S., Raja, L., Bhatt, D. P. Role of ontology in software testing, Journal of Information and Optimization Sciences, 2020, 41(2), 641-649. DOI: 10.1080/02522667.2020.1733196

[25] Stancin, K., Poscic, P., Jaksic, D. Ontologies in education – state of the art. Educ Inf Technol, 2020, 25, 5301–5320. DOI: 10.1007/s10639-020-10226-z

[26] OWL 2 Web Ontology Language Quick Reference Guide (Second Edition), W3C Recommendation 11 December 2012, World Wide Web Consortium. Доступно на http://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/. Задњи пут приступано 20. јула 2023. године

[27] Khair, A. M. M., Meziane, F. The Use Of Ontologies In Software Elicitation. HNSJ, 2021, 2(9). DOI: 10.53796/hnsj2923

[28] Qaswar, F., Rahmah, M., Raza, M. A., Noraziah, A., Alkazemi, B., Fauziah, Z., Hassan, M. K. A., Sharaf, A. Applications of Ontology in the Internet of Things: A Systematic Analysis. Electronics, 2023, 12, 111. DOI: 10.3390/electronics12010111

[29] Popereshnyak, S., Vecherkovskaya, A. Modeling Ontologies in Software Testing. In the proceedings of the 2019 IEEE 14th International Conference on Computer Sciences and

Information Technologies (CSIT), Lviv, Ukraine, 2019, pp. 236-239, DOI: 10.1109/STC-CSIT.2019.8929785.

[30] Protégé, Доступно на https://protege.stanford.edu/. Задњи пут приступано 20. јула 2023. године

[31] Musen, M. A. The Protégé project: A look back and a look forward. AI Matters. Association of Computing Machinery Specific Interest Group in Artificial Intelligence, 2015, 1(4). DOI: 10.1145/2557001.25757003.

[32] Ageed, Z. S., Ibrahim, R. K., Sadeeq, M. A. Unified Ontology Implementation of Cloud Computing for Distributed Systems. Current Journal of Applied Science and Technology, 2020, 39(34), 82-97. DOI: 10.9734/cjast/2020/v39i3431039

[33] Rahayu, N. W., Ferdiana, R., Kusumawardani, S. S. A systematic review of ontology use in E-Learning recommender system, Computers and Education: Artificial Intelligence, 2022, 3, 100047, DOI: 10.1016/j.caeai.2022.100047.

[34] OWLViz, Доступно на https://protegewiki.stanford.edu/wiki/OWLViz# Documentation. Задњи пут приступано 20. јула 2023. године

[35] OntoGraph, Доступно на https://protegewiki.stanford.edu/wiki/OntoGraf. Задњи пут приступано 20. јула 2023. године

[36] Husáková, M., Bureš, V. Formal Ontologies in Information Systems Development: A Systematic Review. Information, 2020, 11, 66. DOI: 10.3390/info11020066

[37] FaCT+ Reasoner, Доступно на http://owl.cs.manchester.ac.uk/tools/fact/. Задњи пут приступано 20. јула 2023. године

[38] HermiT OWL Reasoner, Доступно на http://www.hermit-reasoner.com/. Задњи пут приступано 20. јула 2023. године

[39] Agbaegbu, J., Arogundade, O. T., Misra, S., Damaševičius, R. Ontologies in Cloud Computing—Review and Future Directions. Future Internet, 2021, 13, 302. DOI: 10.3390/fi13120302

[40] Nesterenko, O. Technological Trends & Software Engineering Education: A Systematic Review Study. Problems in Programming, 2022, 3-4, 107-116. DOI: 10.15407/pp2022.03-04.107

[41] Prosyukova, K. O., Shigapova, F. F. Virtualization and digitalization in higher education. In Proceedings of the III International Scientific and Practical Conference (DEFIN '20). Association for Computing Machinery, New York, NY, USA, 2020, Article 6, 1–3. DOI: 10.1145/3388984.3389063

[42] Vodenko, K. V., Chernykh, S. S., Borovaya, L. V. Interactive Development Practices of Modern Russian Education and the Risks of its Virtualization. Bulletin Social-Economic and Humanitarian Research, 2020, 8(10), 16-25. DOI: 10.5281/zenodo.3985877

[43] Liu, J., Li, Z., Zhou, T. Research on the Practical Application of Server Virtualization Technology in Computer Laboratory. J. Phys.: Conf. Ser., 2021, 1744, 022042, DOI: 10.1088/1742-6596/1744/2/022042

[44] Shishkova, A.V., Kozhevnikova, L.V., Starovoytova, I.E. Virtualization of Educational Environment in a Modern Tertiary School. In: Smart Technologies for Society, State and Economy. ISC 2020., Lecture Notes in Networks and Systems, 2021, 155. Springer, Cham. DOI: 10.1007/978-3-030-59126-7_104

[45] Kuyumdzhiev, I. Technical feasibility of a software project for virtualization of tasks performed by students in the higher education course - problems and solutions in evolutionary development. Economics and computer science, 2019, 2, 39-46. ISSN 2367-7791

[46] Huang, A. Teaching, Learning, and Assessment With Virtualization Technology. Journal of Educational Technology Systems, 2019, 47(4), 523–538. DOI: 10.1177/0047239518812707

[47] Wazan, A.S., Kuhail, M. A., Hayawi, K., Venant, R. Which Virtualization Technology is Right for My Online IT Educational Labs?. In proceedings of the 2021 IEEE Global Engineering Education Conference (EDUCON), Vienna, Austria, 2021, pp. 1254-1261, DOI: 10.1109/EDUCON46332.2021.9454048.

[48] Karkalashev, B., Gushev, M. Virtualization technology and possibilities of its use in education. In the Proceedings of The 19th International Conference on Informatics and Information Technologies – CIIT 2022, Skopje, North Macedonia, 2022, pp. 108-110, ISBN 978-608-4699-15-6

[49] Segeč, P., Moravčík, M., Kontšek, M., Papán, J., Uramová, J., Yeremenko, O. Network virtualization tools – analysis and application in higher education, In the proceedings of the 17th International Conference on Emerging eLearning Technologies and Applications (ICETA), Starý Smokovec, Slovakia, 2019, pp. 699-708, DOI: 10.1109/ICETA48886.2019.9040148.

[50] Panum, T.K., Hageman, K., Pedersen, J. M., Hansen, R. R. Haaukins: A Highly Accessible and Automated Virtualization Platform for Security Education. In the proceedins of the IEEE 19th International Conference on Advanced Learning Technologies (ICALT), Maceio, Brazil, 2019, pp. 236-238, DOI: 10.1109/ICALT.2019.00073.

[51] Shevchuk, M., Shevchenko, V., Chukalovskaya, E., Gramakov, D. Cloud platforms and virtualization technologies in education. In the proceedings of the E3S Web Conf., 2020, 210, 22034. DOI: 10.1051/e3sconf/202021022034

[52] Garlisi-Torales, L. D., Adrián Gonzalez, J., Herman-Kaspari, C. A., Aveiro-Róbalo, T. R., Valladares-Garrido, M. J. Impact of virtualization of medical education on academic performance in 2020, Revista Cubana de Medicina Militar, 2022, 51(2). ISSN 1561-3046

[53] Nascimento, K. A. S., Castro Neto, D. N. O., Telles, J. C. C. B. S. The Virtualization Of Health Education In Times Of Covid-19. Reflexão e Ação, Santa Cruz do Sul, 2021, 29(1), 8-19. DOI: 10.17058/rea.v29i1.15748

[54] Affouneh, S., Khlaif, Z. N., Burgos, D., Salha, S. Virtualization of Higher Education during COVID-19: A Successful Case Study in Palestine. Sustainability, 2021, 13, 6583. DOI: 10.3390/su13126583

[55] Intriago, C .Z., Posligua, T. I. Q. Telecommunications and Virtualization in Times of Pandemic: Impact on the Electrical Engineering Career. International Journal of Physical Sciences and Engineering, 2020, 4 (3), 38-44, DOI: 10.29332/ijpse.v4n3.630.

[56] Orejarena, B. O., Murillo, C. R. M., Vicente, J. S. Y. Burnout Syndrome In The Covid-19 Pandemic And The Virtualization Of Education, Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12 (3). DOI: 10.17762/turcomat.v12i3.1937

[57] Ray, S., Srivastava, S. Virtualization of science education: a lesson from the COVID-19 pandemic. J Proteins Proteom, 2020, 11, 77–80. DOI: 10.1007/s42485-020-00038-7

[58] Kim, Y. A Technical Trend and Prospect of I/O Virtualization. The Transactions P of the Korean Institute of Electrical Engineers, 2019, 68P(3), 147-155. DOI: 10.5370/KIEEP.2019.68.3.147

[59] Asvija, B., Eswari, R., Bijoy, M. B. Security in hardware assisted virtualization for cloud computing - State of the art issues and challenges. Computer Networks, 2019, 151, 68-92, DOI: 10.1016/j.comnet.2019.01.013.

[60] Ahmed, B. T. Virtualization Mechanisms and Tools: A Comprehensive Survey. International Journal of Computer Science Engineering (IJCSE), 2020, 9 (4). DOI: 10.21817/ijcsenet/2020/v9i4/200904022

[61] Verma, R., Rane, D., Jha, R. S., Ibrahim, W. Next-Generation Optimization Models and Algorithms in Cloud and Fog Computing Virtualization Security: The Present State and Future. Scientific Programming, 2022, 2419291. DOI: 10.1155/2022/2419291

[62] Oracle VirtualBox, Доступно на https://www.virtualbox.org. Задњи пут приступано 22. јула 2023. године
[63] VirtualBox Virtual Machines, Доступно на https://www.virtualbox.org/wiki/Virtualization. Задњи пут приступано 22. јула 2023. године
[64] Mansouri, Y., Babar, M.A. A review of edge computing: Features and resource virtualization, Journal of Parallel and Distributed Computing, 2021, 150, 155-183. DOI: 10.1016/j.jpdc.2020.12.015.
[65] Alser, M., Waymost, S., Ayyala, R., Lawlor, B., Abdill, R. J., Rajkumar, N. LaPierre, N., Brito, J., Ribeiro-dos-Santos, A. M., Firtina, C., Almadhoun, N., Sarwal, V., Eskin, E., Hu, Q., Strong, D., Kim, B. D., Abedalthagafi, M. S., Mutlu, O., Mangul, S. Packaging, containerization, and virtualization of computational omics methods: Advances, challenges, and opportunities, 2022, arXiv:2203.16261. DOI: 10.48550/arXiv.2203.16261
[66] Kumar, A. Research Issues in Virtualization in Cloud Computing, International Journal of New Innovations in Engineering and Technology, 2020, 12 (4). ISSN: 2319-6319
[67] Tanwar, G. P., Bansal, V., Sharma, S. The challenges and Issues with Virtualization n Cloud Computing. In the proceedings of the 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2021, pp. 1334-1338, DOI: 10.1109/ICOEI51242.2021.9452848.
[68] Ajdari, R. J., Zenuni, X. Cloud Computing Virtualization: A Comprehensive Survey. In the proceedings of the 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2020, pp. 462-472, DOI: 10.23919/MIPRO48935.2020.9245124.
[69] Tiwari, S. K., Neogi, S. G., Mishra, A., Singh, S., Khan, H., Kispotta, S., Purohit, C. Trusted Infrastructure Design for Secure Virtualization in Cloud Computing: A Review. Journal of University of Shanghai for Science and Technology, 2022, 24(9) 1-13. DOI: 10.51201/JUSST/22/0934
[70] Bermejo, B., Juiz, C., Guerrero, C. Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance. J Supercomput 2019, 75, 808–836. DOI: 10.1007/s11227-018-2613-1
[71] Sierra-Arriaga, F., Branco, R. Lee, B. Security Issues and Challenges for Virtualization Technologies. ACM Comput. Surv. , 2020, 53 (2), Article 45. DOI: 10.1145/3382190
[72] Lee, M., Park, S., Song, Y., Eom, Y.I. Introspection of Virtual Machine Memory Resource in the Virtualized Systems. In the proceedings of the 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), Kyoto, Japan, 2019, pp. 1-4, DOI: 10.1109/BIGCOMP.2019.8679210.
[73] Bielski, M., Rigo, A., Pacalet, R. Dynamic Guest Memory Resizing - Paravirtualized Approach. In the proceedings of the 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pavia, Italy, 2019, pp. 181-186, DOI: 10.1109/EMPDP.2019.8671611.
[74] Thyagaturu, S., Shantharama, P., Nasrallah, A., Reisslein, M. Operating Systems and Hypervisors for Network Functions: A Survey of Enabling Technologies and Research Studies. IEEE Access, 2022, 10, 79825-79873. DOI: 10.1109/ACCESS.2022.3194913.
[75] Paraskevas, K., Attwood, A., Luján, M., Goodacre, J. Scaling the capacity of memory systems; evolution and key approaches. In Proceedings of the International Symposium on Memory Systems (MEMSYS '19). Association for Computing Machinery, New York, NY, USA, 2019, pp. 235–249. DOI: 10.1145/3357526.3357555
[76] Deshmukh, P. P., Amdani, S. Y. Virtual Memory Management using Memory Ballooning in OpenStack Cloud Platform. In the proceedings of the 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2020, pp. 1-5, DOI: 10.1109/ICCCNT49239.2020.9225435.

[77] Park, Y., Gu, M., Park, S. Ballooning Graphics Memory Space in Full GPU Virtualization Environments. Scientific Programming, 2019, Article 5240956. DOI: 10.1155/2019/5240956

[78] Li, X., Chen, R., Zhang, B., Li, C. A Dynamic Memory Allocation Approach Based on Balloon Technology on Virtualization Platform. J. Phys.: Conf. Ser., 2019, 1169, 012038. DOI: 10.1088/1742-6596/1169/1/012038

[79] Patil, R., Modi, C. An Exhaustive Survey on Security Concerns and Solutions at Different Components of Virtualization. ACM Comput. Surv. 2019, 52(1), Article 12. DOI: 10.1145/3287306

[80] Niraja, J., Balraj, R. D. Computing Frameworks for VM Migration in Cloud. In the Proceedings of the International Conference on Innovative Computing & Communication (ICICC) 2021. DOI: 10.2139/ssrn.3884449

[81] Pereira, R. F., Silva, R. M., Orvalho, J. P. Virtualization and Security Aspects: An Overview, International Journal of Computer Science and Security (IJCSS), 2020, 14(5). ISSN 1985-1553

[82] Илустровани енглески речник Oxford, Младинска књига Нова, 2002. ISBN 86-84213-00-9

[83] Barkley, J.F., Olsen, K. Introduction to Heterogeneous Computing Environments – NIST Special Publication 500-176, National Institute of Standards & Technology, 1989.

[84] Khokhar, A. A., Prasanna, V. K., Shaaban, M. E., Wang, C. L. Heterogeneous computing: challenges and opportunities. Computer, 1993, 26 (6), 18-27. DOI: 10.1109/2.214439.

[85] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S. H., Skadron, K. Rodinia: A benchmark suite for heterogeneous computing. In the proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC), Austin, TX, USA, 2009, pp. 44-54, DOI: 10.1109/IISWC.2009.5306797.

[86] Fumero, J., Kotselidis, C., Zakkak, F., Papadimitriou, M., Akrivopoulos, O., Tselios, C., Kanakis, N., Doka, K., Konstantinou, I., Mytilinis, I., Bitsakos, C. Programming and Architecture Models. In Heterogeneous Computing Architectures, Challenges and Vision, CRC Press, 2019. DOI: 10.1201/9780429399602-3

[87] Hagleitner, C., Diamantopoulos, D., Ringlein, B., Evangelinos, C., Johns, C., Chang, R. N., D'Amora, B., Kahle, J. A., Sexton, J., Johnston, M., Pyzer-Knapp, E., Ward, C. Heterogeneous Computing Systems for Complex Scientific Discovery Workflows. In the proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2021, pp. 13-18, DOI: 10.23919/DATE51398.2021.9474061.

[88] Benslama, M., Mokhtari, H. Compressed Sensing in Interconnections Covering WiMAX, UMTS and MANET Satellite Networks. In Compressed Sensing in Li-Fi and Wi-Fi Networks, Elsevier, 2017, Pages 137-161. ISBN 9781785482007, DOI: 10.1016/B978-1-78548-200-7.50009-9.

[89] Byers, C., Heterogeneous Computing in the Edge. Industrial Internet Consortium, 2021.

[90] Rasheed, Z., Ashraf, S., Ibupoto, N. A., Butt, P. K., Sadiq, E. H. SDS: Scrumptious Dataflow Strategy for IoT Devices in Heterogeneous Network Environment. Smart Cities, 2022, 5, 1115-1128. DOI: 10.3390/smartcities5030056

[91] Sodhro, A. H., Sangaiah, A. K., Sodhro, G. H., Lodro, M. M., Sekhari, A., Ouzrout, Y., Pirbhulal, S., Fatima, K. Medical Quality of Service Optimization Over Internet of Multimedia Things, In Intelligent Data-Centric Systems, Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Academic Press, 2018, Pages 271-295, ISBN 9780128133149, DOIL 10.1016/B978-0-12-813314-9.00013-X.

[92] Њутн, Х. Њутнов речник, Компјутер библиотека, Београд, Србија, 2005. ISBN 86-7310-319-3

[93] Yang, C., Xiao, Y., Zhang, Y., Sun, Y., Han, J. Heterogeneous Network Representation Learning: A Unified Framework With Survey and Benchmark. IEEE Transactions on Knowledge & Data Engineering, 2022, 34 (10), 4854-4873. DOI: 10.1109/TKDE.2020.3045924

[94] Kärner, T., Warwas, J., Schumann, S. A Learning Analytics Approach to Address Heterogeneity in the Classroom: The Teachers' Diagnostic Support System. Tech Know Learn, 2021, 26, 31–52. DOI: 10.1007/s10758-020-09448-4

[95] Fernández-Gutiérrezdelalamo, L., Bolonio, D., Izquierdo-Díaz, M., Barrio-Parra, F., Mazadiego, L. F., Fidalgo-Blanco, Á. The Use of Heterogeneity to Improve the Learning Process of Large Groups of Students. In Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'18). Association for Computing Machinery, New York, NY, USA, 2018, 794–798. DOI: 10.1145/3284179.3284297

[96] Zhang, S., Wang, S. Modeling Learner Heterogeneity: A Mixture Learning Model With Responses and Response Times. Frontiers in psychology, 2018, 9, 2339. DOI: 10.3389/fpsyg.2018.02339

[97] Cabrera-Lozoya, A., Cerdan, F., Cano, M. D., Garcia-Sanchez, D., Lujan, S. Unifying heterogeneous e-learning modalities in a single platform: CADI, a case study. Computers&Education, 2012, 58 (1), 617-630. DOI: 10.1016/j.compedu.2011.09.014.

[98] Cristescu, I., Balog, A. Heterogeneity of Students' Perceptions of e-Learning Platform Quality: a Latent Profile Analysis. In Proceedings of the 15th International Scientific Conference "eLearning and Software for Education", Bucharest, April 11 - 12, 2019, 195-202. DOI: 10.12753/2066-026X-19-097.

[99] Masud, M. Collaborative e-learning systems using semantic data interoperability, Computers in Human Behavior, 2016, 61, 127-135. DOI: 10.1016/j.chb.2016.02.094

[100] Tang, Z., Zhang, Y., Shi, S., He, X., Han, B., Chu, X. Virtual Homogeneity Learning: Defending against Data Heterogeneity in Federated Learning. In Proceedings of the 39 th International Conference on Machine Learning,17-23 July, 2022, Baltimore, Maryland, USA, PMLR 162, DOI: 10.48550/arXiv.2206.02465

[101] Farcas, A. J., Chen, X., Wang, Z., Marculescu, R. Model elasticity for hardware heterogeneity in federated learning systems. In Proceedings of the 1st ACM Workshop on Data Privacy and Federated Learning Technologies for Mobile Edge Network (FedEdge '22). Association for Computing Machinery, New York, NY, USA, 2022, 19–24. DOI: 10.1145/3556557.3557954

[102] MariaDB Server: The open source relational database. Доступно на https://mariadb.org. Задњи пут приступано 31. јула 2023. године.

[103] MariaDB versus MySQL: Compatibility. Доступно на https://mariadb.com/kb/en/mariadb-vs-mysql-compatibility/. Задњи пут приступано 31. јула 2023. године.

[104] MySQL Workbench. Доступно на https://www.mysql.com/products/workbench/. Задњи пут приступано 31. јула 2023. године.

[105] MySQL Workbench Reference Manual, Oracle, 2023. Доступно на https://downloads.mysql.com/docs/workbench-en.a4.pdf. Задњи пут приступано 31. јула 2023. године.

[106] platform – Access to underlying platform's identifying data. Доступно на https://docs.python.org/3.9/library/platform.html. Задњи пут приступано 1. августа 2023. године.

[107] Distro – an OS platform information API. Доступно на https://pypi.org/project/distro/. Задњи пут приступано 1. августа 2023. године.

[108] subprocess – Subprocess management. Доступно на https://docs.python.org/3.9/library/ subprocess.html. Задњи пут приступано 1. августа 2023. године.
[109] Daniel J. Barrett, Macintosh Terminal Pocket Guide, O'Reilly Media Inc., 2012. ISBN: 9781449328344
[110] py-cpuinfo. Доступно на https://github.com/workhorsy/py-cpuinfo. Задњи пут приступано 1. августа 2023. године.
[111] psutil documentation. Доступно на https://psutil.readthedocs.io/en/latest/. Задњи пут приступано 1. августа 2023. године.
[112] Group of authors, Deployment Guide for Red Hat Enterprise Linux 6 - Deployment, Configuratiion and Administration of Red Hat Enterprise Linux 6, Red Hat Inc. Доступно на https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/deployment_guide/index. Задњи пут приступано 1. августа 2023. године.
[113] Windows Management Instrumentation, Microsoft, 2023. Доступно на https://learn.microsoft.com/en-us/windows/win32/wmisdk/wmi-start-page. Задњи пут приступано 1. августа 2023. године.
[114] lspci – Linux manual page. Доступно на https://man7.org/linux/man-pages/man8/lspci.8.html . Задњи пут приступано 1. августа 2023. године.
[115] system_profiler Man Page macOS. Доступно на https://ss64.com/osx/system_profiler.html. Задњи пут приступано 1. августа 2023. године.
[116] speedtest-cli Python API. Доступно на https://github.com/sivel/speedtest-cli/wiki. Задњи пут приступано 1. августа 2023. године.
[117] Speedtest CLI – Internet connection measurement for developers. Доступно на https://www.speedtest.net/apps/cli. Задњи пут приступано 1. августа 2023. године.
[118] csv – CSV File Reading and Writing. Доступно на https://docs.python.org/3.9/library/csv.html. Задњи пут приступано 1. августа 2023. године.
[119] MySQL Connector/Python Developer Guide, Oracle, 2023. Доступно на https://downloads.mysql.com/docs/connector-python-en.a4.pdf. Задњи пут приступано 1. августа 2023. године.
[120] PyInstaller Manual. Доступно на https://pyinstaller.org/en/stable/. Задњи пут приступано 2. августа 2023. године.
[121] Осмокровић, П., Станковић, К., Вујисић, М. Мерна несигурност, Академска мисао и Електротехнички факултет у Београду, Београд, Србија, 2009. ISBN 978-86-7466-376-9
[122] Howarth, P., Redgrave, F., Метрологија-укратко, треће издање, Дирекција за мере и драгоцене метале, Београд, Србија, 2008. ISBN 978-86-7287-036-7
[123] pandas – Python Data Analysis Library. Доступно на https://pandas.pydata.org/. Задњи пут приступано 2. августа 2023. године.
[124] Matplotlib: Visualization with Python. Доступно на https://matplotlib.org/. Задњи пут приступано 2. августа 2023. године.
[125] Stolic, P., Milosevic, D., Stevic, Z., Radovanovic, I. Ontology Development for Creating Identical Software Environments to Improve Learning Outcomes in Higher Education Institutions. Electronics, 2023, 12, 3057. DOI: 10.3390/electronics12143057
[126] Sengupta, K., Hitzler, P. Web Ontology Language (OWL). Encyclopedia of Social Network Analysis and Mining. 2014. Доступно на https://corescholar.libraries.wright.edu/cse/184. Задњи пут приступано 10. августа 2023. године.

[127] Република Србија. Закон о високом образовању. 2021. Доступно на https://www.paragraf.rs/propisi/zakon_o_visokom_obrazovanju.html. Задњи пут приступано 11. августа 2023. године.
[128] Република Србија. Министарство просвете. 2023. Доступно на https://prosveta.gov.rs/prosveta/visoko-obrazovanje/vrste-i-obim-studija/. Задњи пут приступано 11. августа 2023. године.
[129] What is Linux. Understanding Linux. Red Hat Inc. 2023. Доступно на https://www.redhat.com/en/topics/linux/what-is-linux. Задњи пут приступано 11. августа 2023. године.
[130] Repositories. Ubuntu Documentation. Canonical Ltd. 2022. Доступно на https://help.ubuntu.com/community/Repositories. Задњи пут приступано 11. августа 2023. године.
[131] Graphviz. Доступно на https://graphviz.org/. Задњи пут приступано 11. августа 2023. године.
[132] XML Schema Part 2: Datatypes, Second Edition. World Wide Web Consortium. 2004. Доступно на https://www.w3.org/TR/xmlschema-2/. Задњи пут приступано 12. августа 2023. године.
[133] Stepan, G., Pascal, H., Andreas, A. Knowledge Representation and Ontologies. In Semantic Web Services, Springer: Berlin, Germany. 2007. DOI: 10.1007/3-540-70894-4_3
[134] Tenorth, M., Beetz, M. Representations for robot knowledge in the KnowRob framework. Artificial Intelligence, 2017, 247, 151-169. DOI: 10.1016/j.artint.2015.05.010
[135] Gayathri, R., Uma V. Ontology based knowledge representation technique, domain modeling languages and planners for robotic path planning: A survey. ICT Express, 2018, 4 (2), 69-74. DOI: 10.1016/j.icte.2018.04.008
[136] Robinson, P., Haendel, M. Ontologies, Knowledge Representation, and Machine Learning for Translational Research: recent Contributions. Yearb Med Inform, 2020, 29 (01), 159-162. DOI: 10.1055/s-0040-1701991
[137] SPARQL 1.1 Overview. World Wide Web Consortium. 2013. Доступно на https://www.w3.org/TR/sparql11-overview/. Задњи пут приступано 12. августа 2023. године.
[138] DuCharme, B. Learning SPARQL; O'Reilly Media Inc.: Sebastopol, California, USA, 2011.
[139] Almendros-Jiménez, J. M., Becerra-Terón, A. Type Checking and Testing of SPARQL Queries. In Actas de las XVII Jornadas de Programación y Lenguajes (PROLE 2017), Sistedes, 2017. https://hdl.handle.net/11705/PROLE/2017/002
[140] Motik, B., Shearer, R., Horrocks, I. Hypertableau reasoning for description logics. J. Artif. Int. Res., 2009, 36 (1), 165–228.
[141] Horrocks, I., Motik, B., Wang, Z. The HermiT OWL Reasoner. In Proceedings of the 1st International Workshop on OWL Reasoner Evaluation ORE 2012, Manchester, UK, 1 July 2012, ISSN 1613-0073
[142] Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z. HermiT: An OWL 2 Reasoner, J Autom Reasoning, 2014, 53, 245-269. DOI: 10.1007/s10817-014-9305-1
[143] Almendros-Jiménez, J. M., Becerra-Terón, A., Cuzzocrea, A. Detecting and Diagnosing Syntactic and Semantic Errors in SPARQL Queries. In Proceedings of the Workshops of the EDBT/ICDT 2017 Joint Conference, Venice, Italy, 21-24 March 2012, ISSN 1613-0073
[144] Lamy, J. B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. Artificial Intelligence In Medicine, 2017, 80, 11-28. DOI: 10.1016/j.artmed.2017.07.002

[145] Lamy, J.B. Ontologies with Python: Programming OWL 2.0 Ontologies with Python and Owlready2; Apress: Berkeley, California, USA: 2021. DOI: 10.1007/978-1-4842-6552-9

[146] Lamy, J.B. Reasoning. Owlready2 Documentation. 2019. Доступно на https://owlready2.readthedocs.io/en/latest/reasoning.html. Задњи пут приступано 12. августа 2023. године.

[147] os – Miscellaneous operating system interfaces. Python Software Foundation. 2023. Доступно на https://docs.python.org/3/library/os.html. Задњи пут приступано 12. августа 2023. године.

[148] Rosenberg, J. Security in embedded systems. In Rugged Embedded Systems, Morgan Kaufmann, 2017, Pages 149-205, ISBN 9780128024591, DOI: 10.1016/B978-0-12-802459-1.00006-3.

[149] Tan, L., Jiang, J. Hardware and Software for Digital Signal Processors. In Digital Signal Processing (Third Edition), Academic Press, 2019, Pages 727-784, ISBN 9780128150719, DOI: 10.1016/B978-0-12-815071-9.00014-2.

[150] Sterling, T., Anderson, M., Brodowicz, M. HPC Architecture 1: Systems and Technologies, In High Performance Computing, Morgan Kaufmann, 2018, Pages 43-82, ISBN 9780124201583, DOI: 10.1016/B978-0-12-420158-3.00002-2.

[151] Buchanan, W. Busses, Interrupts and PC Systems, In Computer Busses, Butterworth-Heinemann, 2000, Pages 49-83, ISBN 9780340740767, DOI: 10.1016/B978-034074076-7/50002-X.

[152] User Manual for Oracle VM VirtualBox, Version 7.0.10, Oracle and/or its affiliates, 2023.

[153] PyWin32 – Python Wiki. Доступно на https://wiki.python.org/moin/PyWin32. Задњи пут приступано 22. августа 2023. године.

[154] pywin32 – PyPi. Доступно на https://pypi.org/project/pywin32/#description. Задњи пут приступано 22. августа 2023. године.

[155] win32serviceutil. Доступно на https://github.com/mhammond/pywin32/blob/main/win32/Lib/win32serviceutil.py. Задњи пут приступано 22. августа 2023. године.

[156] Red Hat Enterprise Linux 7 System Administrator's Guide, Red Hat, Inc., 2023. Доступно на https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/pdf/system_administrators_guide/red_hat_enterprise_linux-7-system_administrators_guide-en-us.pdf. Задњи пут приступано 22. августа 2023. године.

[157] Engelhard, C., Páral, K., McKee, C. Understanding and administering systemd, Fedora Quick Docs, Fedora Project, 2020. Доступно на https://docs.fedoraproject.org/en-US/quick-docs/systemd-understanding-and-administering/. Задњи пут приступано 22. августа 2023. године.

[158] Pählig, R. A Launchd Tutorial, Soma-zone, 2022. Доступно на https://www.launchd.info/. Задњи пут приступано 23. августа 2023. године.

[159] Script management with launchd in Terminal on Mac, Terminal User Guide, Apple Inc., 2023. Доступно на https://support.apple.com/en-gb/guide/terminal/apdc6c1077b-5d5d-4d35-9c19-60f2397b2369/mac. Задњи пут приступано 23. августа 2023. године.

[160] Memory Baloon Driver, vSphere Resource Management, VMWare vSphere, VMWare Inc., 2019. Доступно на https://docs.vmware.com/en/VMware-vSphere/7.0/com.vmware.vsphere.resmgmt.doc/GUID-5B45CEFA-6CC6-49F4-A3C7-776AAA22C2A2.html?hWord=N4IghgNiBcIEaQgeyQOxAXyA. Задњи пут приступано 26. августа 2023. године.

[161] Moniruzzaman, A.B. Analysis of Memory Ballooning Technique for Dynamic Memory Management of Virtual Machines (VMs). *Inernational Journal of Grid Distribution Computing, 2014, 7 (6), 81-90. DOI: 10.14257/ijgdc.2014.7.6.07*
[162] *Virtualization Best Practices, SUSE Linux Enterprise Server 15 SP3, SUSE S.A., 2023. Доступно на* https://documentation.suse.com/sles/15-SP3/pdf/article-virtualization-best-practices_en.pdf. Задњи пут приступано 26. августа 2023. године.
[163] Memory Overcommitment, Oracle VM VirtualBox User Guide for Release 7.0, F43655-07 from August 2023, Oracle and/or its affiliates, 2023. Доступно на https://docs.oracle.com/en/virtualization/virtualbox/7.0/user/guestadditions.html#guestadd-memory-usage. Задњи пут приступано 26. августа 2023. године.
[164] Hurbans, R. Алгоритми вештачке интелигенције, Компјутер библиотека, Чачак, Србија, 2021. ISBN 978-86-7310-561-1
[165] Ahmad, I. 40 алгоритама које би сваки програмер требао да зна, Компјутер библиотека, Београд, Србија, 2020. ISBN 978-86-7310-553-6
[166] Shetty, C. Time Series Models - AR, MA, ARMA, ARIMA, Towards Data Science, 2020. Доступно на https://towardsdatascience.com/time-series-models-d9266f8ac7b0. Задњи пут приступано 27. августа 2023. године.
[167] Nagy, Z. Основе вештачке интелигенције и машинског учења, Компјутер библиотека, Београд, Србија, 2019. ISBN 978-86-7310-544-4
[168] Prophet, Meta Open Source, 2023. Доступно на https://facebook.github.io/prophet/. Задњи пут приступано 27. августа 2023. године.
[169] Hyndman, R. J., Athanasopoulos, G. Forecasting: principles and practice, 2nd edition, OTexts, Melbourne, Australia. 2018. Доступно на OTexts.com/fpp2. Задњи пут приступано 27. августа 2023. године.
[170] Matplotlib Application Interfaces (APIs), Matplotlib development team, 2023. Доступно на https://matplotlib.org/stable/users/explain/api_interfaces.html#api-interfaces. Задњи пут приступано 27. августа 2023. године.
[171] MathWorks MATLAB, 2023. Доступно на https://www.mathworks.com/products/matlab.html. Задњи пут приступано 27. августа 2023. године.
[172] Géron, A. Машинско учење: Scikit-Learn, Keras и TensorFlow, Микро књига, Београд, Србија, 2021. ISBN 978-86-7555-449-3
[173] Raschka, S., Mirjalili, V. Python машинско учење, Компјутер библиотека, Београд, Србија, 2020. ISBN 978-86-7310-549-9
[174] NumPy. Доступно на https://numpy.org/. Задњи пут приступано 28. августа 2023. године.
[175] Ensemble methods, Scikit-learn developers, 2023. Доступно на https://scikit-learn.org/stable/modules/ensemble.html. Задњи пут приступано 28. августа 2023. године.
[176] sklearn.ensembleRandomForestRegressor, Scikit-learn developers, 2023. Доступно на https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Задњи пут приступано 28. августа 2023. године.
[177] Taylor, S. J., Letham, B. Forecasting at scale. PeerJ Preprints 2017. DOI: 10.7287/peerj.preprints.3190v2
[178] Raschka, S., Liu, Y. H., Mirjalili, V. Машинско учење: PyTorch и Scikit-Learn, Компјутер библиотека, Београд, Србија, 2022. ISBN 978-86-7310-577-2
[179] Metrics and scoring: quantifying the quality of the predictions, Scikit-learn developers, 2023. Доступно на https://scikit-learn.org/stable/modules/model_evaluation.html. Задњи пут приступано 28. августа 2023. године.
[180] Fedora Linux, The Fedora Project, 2023. Доступно на https://fedoraproject.org/. Задњи пут приступано 30. августа 2023. године.

[181] Fedora Desktops - Memory Footprints, Fedora Magazine, 2019. Доступно на https://fedoramagazine.org/fedora-desktops-memory-footprints/. Задњи пут приступано 30. августа 2023. године.
[182] Fedora LXDE Spin, 2023. Доступно на https://fedoraproject.org/spins/lxde/. Задњи пут приступано 30. августа 2023. године.
[183] Morris, J. SELinux Wiki, 2017. Доступно на http://selinuxproject.org/page/Main_Page. Задњи пут приступано 2. септембра 2023. године.
[184] Kurose, J. F., Ross, K. W. Умрежавање рачунара: од врха ка дну са Интернетом у фокусу, CET, Београд, Србија, 2005. ISBN 86-7991-267-0
[185] Odom, W. CCNA 200-301: званични водич за сертификат, књига 1, Компјутер библиотека, Београд, Србија, 2020. ISBN 978-86-7310-555-0
[186] Lammle, T. CCNA: Cisco Certified Network Associate, студијски приручник, испит 640-801, Компјутер библиотека, Београд, Србија, 2008. ISBN 86-7310-374-6
[187] Naik, G. Научите Linux Shell скриптовање, Компјутер библиотека, Београд, Србија, 2018. ISBN 978-86-7310-528-4
[188] GNU Bash, Free Software Foundation, Inc., 2020. Доступно на https://www.gnu.org/software/bash/. Задњи пут приступано 2. септембра 2023. године.
[189] Open Virtualization Format Specification, Version 2.1.1, DMTF, Portland, Oregon, USA, 2015. Доступно на https://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.1.pdf. Задњи пут приступано 2. септембра 2023. године.
[190] tkinter - Python interface to Tcl/Tk, Python Software Foundation, 2023. Доступно на https://docs.python.org/3/library/tkinter.html#. Задњи пут приступано 2. септембра 2023. године.
[191] Moore, A. D. Python GUI Programming with Tkinter, Second Edition, Packt Publishing, Birmingham, UK, 2021. ISBN 978-1-80181-592-5
[192] urllib.request - Extensible library for opening URLs, Python Software Foundation, 2023. Доступно на https://docs.python.org/3/library/urllib.request.html. Задњи пут приступано 3. септембра 2023. године.
[193] ctypes - A foreign function library for Python, Python Software Foundation, 2023. Доступно на https://docs.python.org/3/library/ctypes.html. Задњи пут приступано 3. септембра 2023. године.
[194] Martin, R. C. Јасан код: приручник за писање јасних програма, друго, ревидирано издање, Микро књига, Београд , Србија, 2022. ISBN 978-86-7555-444-8
[195] Martin, R. C. Чиста архитектура: Стручни водич за структуру и дизајн софтвера, Компјутер библиотека, Београд, Србија, 2020. ISBN 978-86-7310-557-4
[196] Martin, R. C. Чисто мајсторство: Дисциплине, стандарди, етика, Компјутер библиотека, Београд, Србија, 2021. ISBN 978-86-7310-571-0
[197] Поповић, Ј. Тестирање софтвера у пракси: преглед теорије тестирања са примерима из праксе, Рачунарски факултет, Београд, Србија и CET, Београд Србија, 2012. ISBN 978-86-7991-363-0
[198] David Thomas, Andrew Hunt, Прагматични програмер, Компјутер библиотека, Београд, Србија, 2019. ISBN 978-86-7310-546-8
[199] Matthes, E. Python: интезивни курс, Београд, Србија, 2019. ISBN 978-86-7310-589-5
[200] Stolic, P., Stanimirovic, Z., Stanimirovic, I., Jaric, M., Radovanovic, I., Stevic, Z. Application of open source solutions in the realization of low-cost teaching laboratories, In Proceedings of the XXIV International scientific-practical conference "Modern information and electronic technologies", Odesa, Ukraine, 29 – 31 May 2023, 36-39, ISSN 2308-8060
[201] Stolic, P., Ivaz, J., Petrovic, D., Stevic, Z. Advantages of Mining Engineering Curriculum Realization Using Solutions Based on Free Software, In Proceedings of the

52nd International October Conference on Mining and Metallurgy - IOC 2021, Bor, Serbia, 29-30 November 2021, 221-224, ISBN 978-86-6305-119-5

[202] Stolic, P., Milosevic, D. Alternative Software Solutions for Ensuring the Continuity of the Teaching Process in Emergency Situations, Proceedings of the 8th International Scientific Conference Technics and Informatics in Education, Cacak, Serbia, 18-20 September 2020, 196-203, ISBN 978-86-7776-247-6

[203] Наставни план основних академских студија студијског програма Рударско инжењерство, Акредитација 2020. године, Технички факултет у Бору, 2019. Доступно на https://www.tfbor.bg.ac.rs/files/doc/studijski-programi/RI/2019/oas_ri_nastavni_plan.pdf. Задњи пут приступано 5. септембра 2023. године.

[204] Наставни план мастер академских студија студијског програма Рударско инжењерство, Акредитација 2020. године, Технички факултет у Бору, 2019. Доступно на https://www.tfbor.bg.ac.rs/files/doc/studijski-programi/RI/2019/mas_ri_nastavni_plan.pdf. Задњи пут приступано 5. септембра 2023. године.

[205] GNU PSPP, Free Software Foundation, Inc., 2023. Доступно на https://www.gnu.org/software/pspp/. Задњи пут приступано 5. септембра 2023. године.

[206] Eaton, J. W. GNU Octave, 2023. Доступно на https://octave.org/. Задњи пут приступано 5. септембра 2023. године.

БИОГРАФИЈА КАНДИДАТА

Предраг Столић је рођен у Бору 1980. године. Основно и средње образовање завршио је у Бору са одличним успехом. Основне студије завршио је на Универзитету у Београду – Техничком факултету у Бору у оквиру Одсека за информатику, смер Индустријска информатика, са просечном оценом 9,00 током студија. Докторске академске студије уписао је 2014. године на студијском програму Електротехничко и рачунарско инжењерство, модул Рачунарска техника на Факултету техничких наука у Чачку Универзитета у Крагујевцу.

Тренутно је запослен на Универзитету у Београду – Техничком факултету у Бору у звању асистента, ужа научна област Аутоматика и рачунарско инжењерство. У оквиру свог досадашњег ангажмана, као асистент изводи практичну наставу из општих и стручних предмета у оквиру основних академских студија и мастер академских студија.

На Техничком факултету у Бору именован је за лице за заштиту података о личности у складу са Законом о заштити података о личности Републике Србије.

Поседује изузетну склоност ка научно-истраживачком раду.

Објављивао је радове на домаћим и међународним конференцијама, као и у домаћим и међународним часописима, као аутор или коаутор. Као члан организационог одбора учествовао је у организацији више домаћих и међународних конференција. Био је учесник на једном пројекту, као и на једном техничком решењу.

Тренутно ради у подручјима дистрибуираног рачунарства, cloud computinga, анализе и имплементације безбедоносних аспеката информационих система на различитим нивоима (организационим, техничким...), науке о подацима, машинског учења, имплементације Big Data решења, Data Streaminga, примене логовања, анализе лог података, оптимизације лог датотека, Internet of Things, виртуелизације.

Члан је IEEE Computer Society, IEEE Computational Intelligence Society и IEEE Education Society.

Ожењен је и отац је двоје малолетне деце.

Образац 1

ИЗЈАВА АУТОРА О ОРИГИНАЛНОСТИ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да докторска дисертација под насловом:

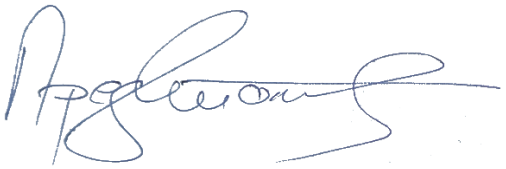
РАЗВОЈ САМОАДАПТИРАЈУЋИХ ВИРТУЕЛНИХ МАШИНА У ЦИЉУ ПОБОЉШАЊА ИСХОДА УЧЕЊА У ХЕТЕРОГЕНОМ РАЧУНАРСКОМ ОКРУЖЕЊУ

представља *оригинално ауторско дело* настало као резултат *сопственог истраживачког рада*.

Овом Изјавом такође потврђујем:

- да сам *једини аутор* наведене докторске дисертације,
- да у наведеној докторској дисертацији *нисам извршио/ла повреду* ауторског нити другог права интелектуалне својине других лица,

У Чачку, 20. септембра 2023. године,



потпис аутора

Образац 2

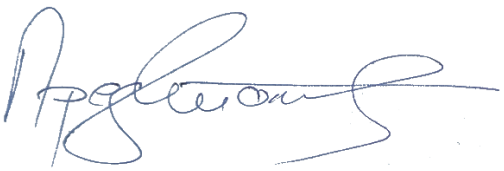
ИЗЈАВА АУТОРА О ИСТОВЕТНОСТИ ШТАМПАНЕ И ЕЛЕКТРОНСКЕ ВЕРЗИЈЕ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Изјављујем да су штампана и електронска верзија докторске дисертације под насловом:

РАЗВОЈ САМОАДАПТИРАЈУЋИХ ВИРТУЕЛНИХ МАШИНА У ЦИЉУ ПОБОЉШАЊА ИСХОДА УЧЕЊА У ХЕТЕРОГЕНОМ РАЧУНАРСКОМ ОКРУЖЕЊУ

истоветне.

У Чачку, 20. септембра 2023. године,



потпис аутора

Образац 3

ИЗЈАВА АУТОРА О ИСКОРИШЋАВАЊУ ДОКТОРСКЕ ДИСЕРТАЦИЈЕ

Ја, Предраг Р. Столић,

[X] дозвољавам

[ ] не дозвољавам

Универзитетској библиотеци у Крагујевцу да начини два трајна умножена примерка у електронској форми докторске дисертације под насловом:

РАЗВОЈ САМОАДАПТИРАЈУЋИХ ВИРТУЕЛНИХ МАШИНА У ЦИЉУ ПОБОЉШАЊА ИСХОДА УЧЕЊА У ХЕТЕРОГЕНОМ РАЧУНАРСКОМ ОКРУЖЕЊУ

и то у целини, као и да по један примерак тако умножене докторске дисертације учини трајно доступним јавности путем дигиталног репозиторијума Универзитета у Крагујевцу и централног репозиторијума надлежног министарства, тако да припадници јавности могу начинити трајне умножене примерке у електронској форми наведене докторске дисертације путем *преузимања*.

Овом Изјавом такође

[X] дозвољавам

[ ]

не дозвољавам[1]

припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од следећих *Creative Commons* лиценци:

1) Ауторство

2) Ауторство - делити под истим условима

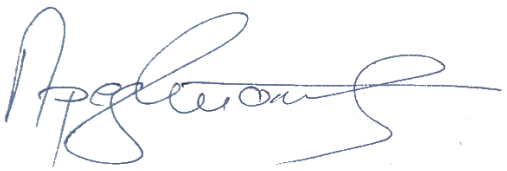
3) Ауторство - без прерада

4) Ауторство - некомерцијално

5) Ауторство - некомерцијално - делити под истим условима

6) Ауторство - некомерцијално - без прерада[2]

У Чачку, 20. септембра 2023. године,



потпис аутора

[1] Уколико аутор изабере да не дозволи припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци, то не искључује право припадника јавности да наведену докторску дисертацију користе у складу са одредбама Закона о ауторском и сродним правима.

[2] Молимо ауторе који су изабрали да дозволе припадницима јавности да тако доступну докторску дисертацију користе под условима утврђеним једном од *Creative Commons* лиценци да заокруже једну од понуђених лиценци. Детаљан садржај наведених лиценци доступан је на: http://creativecommons.org.rs/